

UNIVERSIDAD POLITÉCNICA DE MADRID
FACULTAD DE INFORMÁTICA
MÁSTER UNIVERSITARIO EN SOFTWARE
Y SISTEMAS

**Realidad Aumentada en Interiores:
posicionamiento del usuario en
dispositivos móviles y aplicaciones en
rehabilitación y guiado (GuIAR)**

Trabajo Fin de Máster

Autor: Anaolena Martínez Ospina
anaolena.martinez@alumnos.upm.es

Directora: Dra. Angélica de Antonio Jiménez
angelica@fi.upm.es

Codirector: Dr. Oscar Javier Romero López
ojrlopez@hotmail.com

Septiembre 2013



*Somos básicamente máquinas de soñar
que construyen modelos virtuales del mundo real*

Agradecimientos

Primero que todo quiero agradecer a Dios por ser mi guía y fortaleza en todo momento.

A mi prometido Mauricio, sencillamente por aparecer en mi vida y apoyarme en toda circunstancia... siempre estás ahí en todo momento

A mis padres, hermanas y demás familiares por su incondicional apoyo desde Colombia, siempre dándome ánimos para la culminación de ésta meta.

También quiero agradecer a mi directora Angélica de Antonio y codirector Oscar López por los buenos y experimentados consejos proporcionados en el seguimiento de este trabajo y por su valiosa dedicación.

A la división de ATOS SI Solutions & Data and Information Management perteneciente a Atos Spain, donde me encuentro laborando actualmente, por brindarme las facilidades de horarios que me permitieron cursar este máster.

De igual manera, gracias a la Universidad Politécnica de Madrid el cual me permitió adquirir un conjunto de conocimientos y experiencias que no solo se ven reflejadas en este trabajo sino también en otros aspectos de mi vida profesional.

Y a mis compañeros de estudio, colegas y equipo de prácticas; les agradezco su valiosa amistad y momentos compartidos durante esta etapa de mi vida

A todos gracias y los llevo siempre en mi corazón

Resumen

La amplia difusión de los dispositivos móviles en el mercado actual ha puesto al alcance de la mano, altas capacidades de cómputo y diversidad de sensores. Esto ha posibilitado el auge de la Realidad Aumentada, lo cuál busca superponer información al mundo real con el fin de enriquecerlo para un fin determinado. Este trabajo presenta un estado de la cuestión sobre esta área de investigación y se centra específicamente en la Realidad Aumentada en espacios interiores. Uno de los principales problemas de este escenario es el de obtener la posición del usuario y utilizar esta información durante el proceso de Realidad Aumentada, este problema y las soluciones propuestas han sido descritas y evaluadas en el presente documento.

Sin embargo, este trabajo da un paso fuera del ámbito teórico al construir un prototipo de Realidad Aumentada en Interiores, el cual implementa diversos aspectos tratados en el estado de la cuestión para un dominio de aplicación concreto, esta implementación permitió por una parte identificar las falencias de los elementos actualmente propuestos para posicionamiento en interiores y por otra parte detectar y afrontar otros problemas técnicos en el desarrollo de este tipo de aplicaciones. Toda esta experiencia es descrita en este documento con el fin que sirva de guía para futuros trabajos en el área.

Palabras claves: Realidad Aumentada en interiores, Android, posicionamiento en interiores.

Abstract

The diffusion of mobile devices in the market brought high computational capabilities and a set of different sensors to users hand. This has allowed the grow of Augmented Reality, which tries to mix digital data and the real world in order to provide an enriched experience to the user in a specific domain. This work presents a state of the art about this area focused on indoor augmented reality. One of the problems in this specific area is to get the exact users position in a building and use this data in the augmented reality process. This problem and the related work are described and evaluated in this document.

In addition, this work not only provides a theoretical discussion but goes a step further building a indoor augmented reality prototype, which uses different approaches described in the state of the art in a specific domain, this allows to evaluate in a practical way the ideas proposed in the state of the art and also deal with other technical issues related with this kind of software. All these experiences are described in this document in order to guide further works in this area.

Keywords: Indoor Augmented Reality, Android, indoor positioning.

Índice general

Agradecimientos	2
Resumen	4
Abstract	5
1. Introducción	12
1.1. Descripción del problema y objetivos	12
1.1.1. Objetivo General	13
1.1.2. Objetivos específicos	13
1.1.3. Estructura del documento	14
2. Estado de la Cuestión	16
2.1. Realidad Aumentada	16
2.1.1. Breve historia	17
2.1.2. Clases de Realidad Aumentada	18
2.2. Aplicaciones de Realidad Aumentada	20
2.2.1. AndAR	20
2.2.2. Mixare	21
2.2.3. Layar Reality Browser	21
2.2.4. Análisis y conclusiones	22
2.2.5. Uso de la Realidad Aumentada en otros campos	23
2.3. Sistemas operativos móviles	26
2.3.1. iOS	27
2.3.2. Android	27
2.4. Sistemas de localización	33
2.4.1. PlaceLab	34
2.4.2. Proyecto Indoor Navigation System for Handheld Devices	34
2.5. Sistemas de posicionamiento en interiores	35
2.5.1. Marcadores y marcas visibles	37
2.5.2. Etiquetas RFID	38
2.5.3. Sistemas iniciales	39
2.5.4. Cálculo del punto central a partir de nodos predefinidos .	40
2.5.5. Triangulación de la señal	41
2.5.6. Detección de movimiento	42
2.5.7. Mapas de Radio Wifi	42
2.6. Componentes Base del Proyecto	43
2.6.1. Sistema MILES	43

ÍNDICE GENERAL

2.6.2. Framework LooK	45
2.6.3. Sensor Test	46
3. Solución Adoptada	50
3.1. Arquitectura Global	50
3.2. Diseño e implementación de GuIAR	53
3.2.1. Intermediario REST	53
3.2.2. Realidad Aumentada Indoor	55
3.2.3. Modo de uso de la aplicación	79
4. Experimentación y Resultados	83
4.1. Sistema REHABILITA	83
4.1.1. Escenario para REHABILITA	84
4.2. Experimentación de ámbito Educativo MILES	91
4.2.1. Escenario I	91
4.2.2. Escenario II	93
4.2.3. Escenario III	95
5. Conclusiones y Trabajo Futuro	102
5.1. Trabajo Futuro	103
Referencias	105
Apéndices	109
Dispositivos Móviles	110
.1. Smartphone I	110
.2. Smartphone II	111
Instalación de GuIAR	114
Herramientas de Desarrollo	116
Estructura del código fuente	117
Codificaciones	120

Índice de figuras

2.1.	Realidad Aumentada en un dispositivo móvil	17
2.2.	Funcionamiento de realidad aumentada en Smartphone y ordenadores	18
2.3.	Reconocimiento de una marca	20
2.4.	Captura de Mixare	21
2.5.	Aplicación de realidad aumentada Layar	22
2.6.	IMagic Books con marcas de RA	23
2.7.	Revista de IKEA en Realidad Aumentada	24
2.8.	Reconocimiento de la estatua y aportando información característica con RA	24
2.9.	Realidad aumentada en las aulas de medicina	25
2.10.	Aprender las partes del brazo con ayuda de marcadores para la RA	25
2.11.	Obtener información de un jugador de fútbol con RA	26
2.12.	Logo de iOS	27
2.13.	Logo Android	28
2.14.	Diagrama de flujo del ciclo de vida de una aplicación Android	29
2.15.	Orientación en Android	31
2.16.	Ejecución de Indoor Navigation System for Handheld Devices	35
2.17.	Sistema de localización indoor	36
2.18.	Diversos sistemas de posicionamiento local	36
2.19.	Código QR	37
2.20.	Códigos BIDI, utilizados al momento de acceder a algún espectáculo	37
2.21.	Etiqueta RFID	38
2.22.	Fluctuación de la señal en redes Wifi	40
2.23.	Triangulación de señal	42
2.24.	Arquitectura global del sistema MILES	44
2.25.	Pantalla principal de Look Social con menús	46
2.26.	Diagrama de Flujo: Sistema de Navegacion Inercial de Look	47
2.27.	Comportamiento de la aceleración durante el ciclo de caminado	47
3.1.	Arquitectura Global	51
3.2.	Diagrama de clases del intermediario REST	54
3.3.	Diagrama de Clases de la aplicación GuIAR	57
3.4.	Diagrama de Clases del sistema de navegación inercial de Look.	60
3.5.	Diagrama de secuencia del sistema de navegación inercial	62
3.6.	Diagrama de secuencia de la interacción de la aplicación Android con el sistema MILES	65

ÍNDICE DE FIGURAS

3.7.	Diagrama de secuencia del entorno de Realidad Aumentada en Interiores GuIAR	68
3.8.	Interfaz gráfica de la sesión de inicio de GuIAR	71
3.9.	Mensaje de Calibración del smartphone	72
3.10.	Coordenadas polares	73
3.11.	Coordenadas polares y cartesianas	74
3.12.	Modo de uso de la aplicación GuIAR	80
4.1.	Fotografía A y B Marcación de la ruta	85
4.2.	(A) Ruta de 277 cms (2.77mts), (B) Ruta de 1300 cms (13mts) y (C) Ruta de 52000 cms (52mts)	85
4.3.	Captura de la pantalla del móvil cuando queremos iniciar la marcha y aún se encuentra en cero las variables	86
4.4.	Captura de la pantalla del móvil cuando finalizó la primera ruta (A)	86
4.5.	Recorrido de 250 metros aproximados. Tomada de Google Maps .	89
4.6.	Plano del interior de la sala principal	91
4.7.	Representación de cada uno de las trayectorias realizadas	94
4.8.	Correlación lineal que relaciona el crecimiento del error (empleando el cálculo de la distancia Euclídea) a medida que aumenta la distancia del recorrido	97
4.9.	Correlación lineal que relaciona el crecimiento del error (empleando el cálculo de la distancia Manhattan) a medida que aumenta la distancia del recorrido	97
4.10.	Seguimiento de las indicaciones de GuIAR, vista completa.	99
4.11.	Indicación de GuIAR orientando a la izquierda	99
4.12.	Indicación de GuIAR orientando a la derecha	100
4.13.	Indicación de GuIAR orientando a seguir recto	100
1.	Smartphone Samsung Galaxi S Plus	111
2.	Smartphone Sony Xperia S	112
3.	Estructura del código GuIAR	118

Índice de cuadros

2.1.	Tabla comparativa de las características de las aplicaciones de realidad aumentada	22
2.2.	Arquitectura Android	28
2.3.	Descripción de los tipos de sensores, de la clase Sensor	32
3.1.	Clase: MILESService, fragmento de código del método getPunto- sInteres	56
3.2.	Código fuente de la Clase MainActivity donde se incorpora el método INS	63
3.3.	Código fuente de la clase asíncrona ObtenerJSON	66
3.4.	Código fuente del método getNames	69
3.5.	Código fuente del acceso a los datos de los sensores del dispositivo móvil	75
3.6.	Código fuente del método updateSensorData	75
3.7.	Código fuente del método que permite la calibración del dispositi- vo móvil	76
3.8.	Código fuente del método que obtiene la orientación generada por los sensores del móvil	76
3.9.	Código fuente del método obtenerOrientacionDestino	78
4.1.	Estadísticas Ruta (A)	87
4.2.	Estadísticas Ruta (B)	88
4.3.	Estadísticas Ruta (C)	88
4.4.	Estadísticas tracking largo	89
4.5.	Error de orientación obtenido por cada objeto de interés	92
4.6.	Datos de posición real, la estimada y la distancia recorrida por cada trayectoria	96

Capítulo 1

Introducción

Desde el año 2009 se ha provocado un crecimiento exponencial de aplicaciones sobre Realidad Aumentada (en adelante RA) para plataformas móviles, cada vez están abriendo nuevos caminos a la innovación y a la adopción de estas tecnologías por la gente de forma intensiva, y es aquí donde se vé que ésto no ha hecho más que empezar.

Debido a la revolución que se tiene con los Smartphones, el Internet de las cosas se mete prácticamente en los bolsillos de cada uno. Los móviles con cámaras y sensores se convierten en los ojos y oídos de cada individuo; los sensores de movimiento y localización indican la ubicación geoespacial, la orientación y el desplazamiento de una persona a través del dispositivo móvil; recogiendo y mostrando estos datos en tiempo real.

Existen dos tipos de sistemas de localización, tanto el que puede darse en interiores como en exteriores; estos han tomado un gran valor al día de hoy. Para la localización en exteriores, existe un sistema bastante fiable y de buena precisión, como lo es el GPS (Global Positioning System) uno de los dispositivos más usados en todo smartphone, debido a su buen funcionamiento y facilidad de adquisición por el coste. En cuanto a la localización en interiores existen diversas ramas que se encuentran en desarrollo, debido a que se pretende que un dispositivo portable permita la ubicación de una persona dentro de un recinto cerrado y que éste sea de bajo presupuesto tanto para el usuario como para el edificio que se explora.

1.1. Descripción del problema y objetivos

Una de las formas en que las personas interactúan con la Realidad Aumentada (RA), es a través de la ubicación basada en aplicaciones móviles. Aprovechando el campo de visión de un Smartphone, los usuarios pueden visualizar de cerca puntos de interés y demás información relacionada con la ubicación, con base a la información obtenida de diversos sensores del móvil (giróscopos, acelerómetros y campo magnético). La mayoría de aplicaciones se encuentran disponibles para

1.1. DESCRIPCIÓN DEL PROBLEMA Y OBJETIVOS

la localización en espacios exteriores; en este trabajo se pretende dar un sistema de seguimiento de movimiento o Tracking System de forma personalizada al usuario en espacios interiores con ayuda de la RA; con el fin de proporcionarle al usuario servicios (cuando la señal de GPS no está disponible) e información relacionada con su ubicación, para ser orientado a su destino de forma adaptativa. A continuación se describe el objetivo general y objetivos específicos:

1.1.1. Objetivo General

Proponer e implementar una técnica de realidad aumentada para la orientación personalizada en espacios interiores a través de dispositivos móviles, denominado GuIAR, la cual será evaluada en dos casos de estudio prácticos; de ámbito médico y educativo.

1.1.2. Objetivos específicos

1. Realizar un estado de la cuestión sobre la realidad aumentada, aplicada en el posicionamiento geoespacial utilizando dispositivos móviles.
2. Desarrollar un cliente móvil de realidad aumentada que interactúe con la arquitectura de software basada en agentes MAEVIF, utilizada en el proyecto MILES, para permitir la navegación espacial en interiores y aplicarlo a un escenario educativo.
3. Visualizar información relacionada con la distancia y velocidad recorrida por el usuario, incorporando y adaptando un componente de medición basado en el giroscopio
4. Definir y representar gráficamente la trayectoria de orientación con objetos comunes en el dibujado, como flechas y formas básicas en dos dimensiones para la realidad aumentada.
5. Representar la información verbal de forma aumentada, como instrucciones de la trayectoria a seguir.
6. Proponer un modelo geométrico que permita calcular la ubicación aproximada de los elementos en el espacio en función del vector de orientación del usuario y las instrucciones de la trayectoria a seguir, la cual es calculada desde la plataforma MAEVIF.
7. Desarrollar un método de comunicación entre la plataforma MAEVIF del sistema MILES y la plataforma Android de la aplicación GuIAR, permitiendo la publicación y suscripción de mensajería, entre estos dos sistemas.

1.1. DESCRIPCIÓN DEL PROBLEMA Y OBJETIVOS

1.1.3. Estructura del documento

Para satisfacer los objetivos antes mencionados, El presente trabajo se divide de la siguiente forma: El capítulo 2 presenta un estado de la cuestión sobre la realidad aumentada, en el cual se describen los tipos de clases existentes y se analizan aplicaciones de RA relevantes, el sistema operativo a utilizar, se profundiza en el estado de los distintos tipos de técnicas de localización en interiores y se describen los componentes base a reutilizar en el desarrollo de éste trabajo. Todo esto servirá para el diseño de la estrategia propuesta en ésta memoria. Por su parte el capítulo 3, describe la solución propuesta en este trabajo, como es: la realidad aumentada en interiores (GuIAR).

Seguidamente el capítulo 4 presentará la experimentación para el caso de estudio de ámbito educativo y médico, validando de ésta forma la funcionalidad implementada.

Finalmente, el capítulo 5 presenta las conclusiones, los principales resultados de éste trabajo y también se propone el camino de futuros trabajos en ésta área de investigación. Adicionalmente éste trabajo contiene algunos apuntes finales como anexos, que ayudan a la comprensión.

Capítulo 2

Estado de la Cuestión

Este capítulo permite dar una visión global del estado actual de la Realidad Aumentada en Interiores y el posicionamiento del usuario con dispositivos móviles, desde sus comienzos hasta la actualidad, pasando por las distintas plataformas, técnicas en los que se han desarrollado y ver de cierta manera la evolución en el futuro y, el apogeo causado. Siendo ésta la forma de proporcionar la información necesaria para entender las peculiaridades de éste proyecto de fin de Máster.

2.1. Realidad Aumentada

En inglés Augmented Reality (AR), se define como la tecnología que permite la superposición en el tiempo real, de imágenes generadas por ordenador sobre el mundo real. Estos datos superpuestos puede ser información pertinente a los elementos reales que se visualizan en el momento (datos que sirven de referencia física) ó información que puede servir de guía o aprendizaje según el medio donde se encuentre y la aplicación escogida. Además el usuario puede interactuar con los objetos virtuales objetos en 2D ó 3D permitiendo añadir nuevos objetos ó tan solo modificando los existentes. De ésta forma lo que se pretende es combinar el mundo real con el virtual mediante un proceso informático, enriqueciendo la experiencia visual y la realidad cotidiana.

Un ejemplo de RA basada en la posición a través de un dispositivo móvil, se aprecia en la figura 2.1, en ella se muestra a través de la cámara los distintos barrios mas cercanos desde la ubicación actual del usuario.

Para un buen funcionamiento de una aplicación de Realidad Aumentada, se debe tener en cuenta cuatro importantes componentes mostrados en la figura 2.2¹ y explicados cada uno de ellos a continuación:

- *Cámara*: este elemento permite captar la realidad y a su vez es la fuente de información real para la aplicación (encontrándose en los ordenadores y teléfonos móviles).

¹<http://www.lacofa.es/index.php/general/realidad-aumentada-una-nueva-lente-para-ver-el-mundo-i>

2.1. REALIDAD AUMENTADA



Figura 2.1: Realidad Aumentada en un dispositivo móvil

- *Pantalla*: permite proyectar la mezcla de imágenes reales con las imágenes virtuales (como pantalla del ordenador, teléfono móvil o consola de videojuegos).
- *Hardware y software RA*: la RA necesita de estos dos componentes principales en conjunto, pues se necesita de un sistema de display ó lente visor (hardware: ejemplo la cámara) para mostrar al usuario la información visual (software: ejemplo imágenes virtuales 2D ó 3D) que se añade al mundo real.
- *Activador de realidad aumentada*: existe una gran variedad de activadores como son: una imagen que este visualizando el usuario, localización con GPS, sensores, marcadores RFID, Bidi ó cualquier otro elemento que proporcione información de lo que ve el usuario. Para luego ser proyectada la información sintetizada, de forma que el ojo del individuo sea capaz de ver en tiempo real sobre su entorno.

2.1.1. Breve historia

El término Realidad Aumentada fue introducido en 1990, por el físico e investigador Tom Caudell de la empresa Boeing. Caudell fue contratado para encontrar una alternativa a los complejos sistemas de tableros de configuración de cables que utilizaban los trabajadores; fue allí cuando salió con la idea de anteojos especiales y tableros virtuales sobre tableros reales genéricos; de esta forma fue que se le ocurrió que estaba “aumentando” la realidad del usuario, y surgió el término de Realidad Aumentada en 1992.

Luego en esa misma década, surgió la idea de resolver los problemas de seguimiento del movimiento ó también llamado tracking, que se basaban en intentar

2.1. REALIDAD AUMENTADA



Figura 2.2: Funcionamiento de realidad aumentada en Smartphone y ordenadores

capturar los movimientos y variaciones, para que todos los fotogramas estuvieran definidos en un sistema 3D de geometría euclídea. Esto requería un control muy minucioso de la posición del individuo y de la cámara; por lo que con frecuencia surgían cada vez mas desviaciones y fallos de alineamiento entre el mundo virtual y real.

En 1996, Fred Brooks del Departamento de Ciencias de la Computación de la Universidad de Carolina del Norte, aseguró que no creía en la posibilidad de que funcionase el tracking de la posición del usuario y ahora podemos ver que se está ganando popularidad en las plataformas de teléfonos móviles, como el método de combinación de los dispositivos del sistema del posicionamiento global (GPS) ú otros datos de seguimiento de localización y los datos de acelerómetro para determinar donde se encuentra el dispositivo y en qué dirección está apuntando; con esta información como base y etiquetas, anotaciones, se superponer a la escena, teniendo así el tracking en exteriores con Realidad Aumentada y otros servicios geográficos de localización. En cuanto a la localización en interiores se ha venido avanzado bastante y actualmente existen aplicaciones que hacen uso de diferentes técnicas de localización, pero aún no hay alguna en la que se obtenga una precisión exitosa en cuanto a posicionamiento.

2.1.2. Clases de Realidad Aumentada

La realidad aumentada funciona de cuatro maneras diferentes, dependiendo de la plataforma desde la que se ejecuta; a saber:

2.1. REALIDAD AUMENTADA

- **Realidad aumentada desde teléfonos inteligentes:** para llevar a cabo este tipo de experiencia el usuario debe tener un Smartphone del tipo iPhone, Blackberry, Android ó similares, que tengan cámara digital posterior, al cual se le descarga un programa que permite ejecutar la realidad aumentada. Normalmente se usan en combinación con un sistema de posicionamiento global (GPS), para realizar la geolocalización en tiempo real del aparato, y con ello poder interpretar la posición de la cámara en el momento que ésta es puesta frente al usuario en movimiento. Su funcionamiento se reduce a activar el teléfono y el GPS, cargar el programa residente en su memoria y apuntar la cámara al escenario que se desea complementar.

En ese momento, sobre la pantalla del dispositivo, aparecerá el entorno actual con objetos de información asociados al lugar donde se ubica el usuario. Sus aplicaciones van especialmente dirigidas al turismo, como identificador de sitios de interés, ámbito educativo para mejora de la enseñanza, al marketing por proximidad, para temas relacionados con el comercio, o a los juegos, entre otras.

- **Realidad aumentada con gafas especiales:** en esta posibilidad el usuario debe tener, como sensor, unas gafas translúcidas que van conectadas de manera alámbrica ó inalámbrica, a un PC o un teléfono inteligente, desde donde se ejecutará el programa que permite la experiencia. Estas gafas facilitan ambientes muy inmersivos, desde donde la experiencia en realidad aumentada abarcará todo el campo visual del individuo, propiciando así situaciones ricas en experiencias por descubrimiento. Sus aplicaciones están enfocadas a la instrucción y al entrenamiento o la medicina entre muchas otras posibilidades.

- **Realidad aumentada OFF-line desde un PC:** este tipo de realidad aumentada se lleva a cabo desde una computadora personal convencional, en asocio con una cámara web de resolución media que sirve de sensor, un programa elaborado para ser ejecutado desde el PC, y un marcador impreso que representa la manera en que el evento es invocado para la visualización sobre la pantalla del computador.

Para este propósito no se requiere ningún tipo de conectividad a Internet, y todos los procesos informáticos se llevan a cabo desde y con la computadora únicamente. Para su funcionamiento debe ejecutarse el programa de representación de los modelos virtuales en el PC, activar la cámara web y poner frente a ésta los marcadores que representan la actividad que se quiere visualizar. Estas actividades pueden ser un objeto en 3D, un video, un texto, un sonido, o la combinación de todos.

- **Realidad aumentada ON-line desde un PC:** a diferencia de la anterior, se requiere necesariamente una conexión alámbrica o inalámbrica a Internet, desde donde se ejecutan las rutinas del programa que interpreta los modelos virtuales; sólo se necesita el visor para Flash normalmente disponible en las computadoras para la respectiva visualización de este

2.2. APLICACIONES DE REALIDAD AUMENTADA



Figura 2.3: Reconocimiento de una marca

tipo de formatos. La cámara web y los marcadores serán también parte del proceso. Para su funcionamiento sólo debe cargarse desde un navegador, o browser en Internet, una dirección web o URL (Uniform Resource Locator), esperar a que se ejecute el Adobe Flash Player, autorizar su uso desde la caja de diálogo que aparece sobre la pantalla del PC, y poner frente a ella los marcadores suministrados para la experiencia.

Las aplicaciones para estas dos últimas plataformas con el PC abarcan prácticamente todas las profesiones y oficios, donde se resaltan especialmente los procesos pedagógicos y educativos, la Medicina misma, la Publicidad, el entretenimiento, etc(Enrique, 2011).

2.2. Aplicaciones de Realidad Aumentada

Hoy en día se encuentran infinitas aplicaciones de RA, algunas públicas otras privadas y otras en fase de desarrollo; por lo que en esta sección se presentará las aplicaciones más relevantes orientadas al posicionamiento (uno de los objetivos de ésta memoria) y al final de la sección se mencionan las aplicaciones más comunes en otros campos de uso como: la medicina, publicidad, cultura, entretenimiento y educación.

2.2.1. AndAR

AndAR es una aplicación que permite la representación de objetos tridimensionales sobre marcas físicas en el mundo real. Consiste en un visor de elementos 3D, que son situados sobre marcas predeterminadas. Estos elementos se verán afectados por la orientación y el punto de vista del usuario (ver figura 2.3 tomada de (AndAR, 2010)).

2.2. APLICACIONES DE REALIDAD AUMENTADA

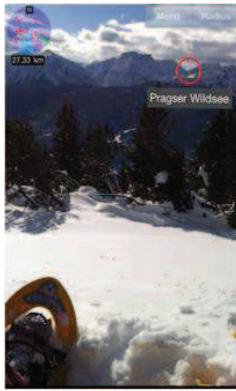


Figura 2.4: Captura de Mixare

2.2.2. Mixare

Mixare (mix Augmented Reality Engine) es un framework de código abierto para realidad aumentada, publicada bajo la licencia GPLv3. Mixare está disponible para sistemas Android y para iPhone (ver figura 2.4 tomada de(Mixare, 2012)).

Este framework permite construir aplicaciones completas y proporciona funciones para asociar puntos (coordenadas) y texto. Es decir, su funcionalidad permite asociar texto a localizaciones.

Características:

- Ofrece capacidades de representación en dos dimensiones limitadas a cajas de texto e imágenes.
- localización por GPS.
- Acceso a datos por conexión de red.

2.2.3. Layar Reality Browser

Layar es un navegador de realidad aumentada, desarrollado para plataformas móviles como Android o iPhone (Madden, 2011). Tiene una licencia privativa por lo que no se dispone de acceso al código fuente. Está basado en un sistema de capas que funcionan sobre el navegador de realidad aumentada base, y que el usuario puede decidir si mostrar o no (ver figura 2.5²).

Características:

- Localización basada en GPS.

²<https://www.layar.com/>

2.2. APLICACIONES DE REALIDAD AUMENTADA



Figura 2.5: Aplicación de realidad aumentada Layar

Aplicación	Look!	Layar	mixare	AndAR
Integración con cámara	X	X	X	X
2D	X	X	X	
3D	X	X		X
Integración con Servicios Remotos	X	X	X	
Creación de Servicios Remotos	X	X		
Localización en interiores	X			
Localización por GPS		X	X	
Sistema de Navegación Inercial	X			
Interacción con Objetos Virtuales	X	X		

Cuadro 2.1: Tabla comparativa de las características de las aplicaciones de realidad aumentada

- Capas en dos dimensiones
- Capas en tres dimensiones
- Estructura de cliente-servidor, permite la descarga de datos de las capas definidas por los usuarios en tiempo real.
- Promoción de las capas: las capas definidas por el usuario pueden ser puestas a disposición de la comunidad de manera centralizada.

2.2.4. Análisis y conclusiones

En la tabla 2.1, se puede observar que la aplicación AndAR y Mixare, ofrecen una funcionalidad muy básica y además que no son aplicables en la localización en interiores. De Layar, al ser una aplicación de código privado, no es posible tenerla en cuenta para futuros proyectos con base a ella. Queda el framework Look (que se describe en el apartado 2.6.2 como componente base al desarrollo de este trabajo), que parece tener todas las funcionalidades aceptables para que pueda funcionar la localización en interiores. Aunque una parte de su código fuente se encuentra limitada a datos de conexión con un servidor y base de datos propia de los creadores, se puede obtener una parte del módulo de localización

2.2. APLICACIONES DE REALIDAD AUMENTADA

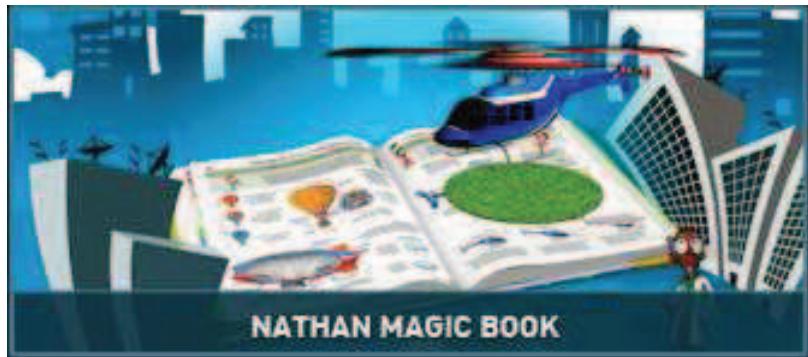


Figura 2.6: IMagic Books con marcas de RA

donde utilizan el método de navegación inercial, que es el que se requiere para este proyecto.

2.2.5. Uso de la Realidad Aumentada en otros campos

Hasta aquí se ha presentado un estado de la cuestión de las aplicaciones de RA teniendo en cuenta siempre el posicionamiento, pero la RA no sólo se basa en la posición, es decir, existen otras aplicaciones en distintos campos de uso, aunque en ésta memoria el objetivo principal es tener en cuenta el posicionamiento, no está de más mencionar estos campos de uso y un ejemplo de cada uno de ellos; realmente es innegable el gran impacto visual que logra aportar ésta tecnología y la facilidad de comprensión que se tiene con los objetos virtuales. A continuación se mencionan algunas aplicaciones comunes en diversos campos:

- **Educativo:** La RA en éste campo ha sido de gran importancia desde el nivel investigativo y universitario (como ejemplo éste trabajo de fin de máster) hasta el nivel escolar aprendizaje en los niños, ya que las imágenes pueden llegar a dar una visión y entendimiento mas claro que una cantidad de palabras tratando de explicar algo. Por ejemplo, el poder ver un volcán en erupción a centímetros del rostro de una persona, de forma tridimensional y en tiempo real; es algo que no podría verse en el mundo real pues podría llegar a causar lesiones graves.

Otro ejemplo se presenta en la figura 2.6³, corresponde a libros llamados *IMagic Books* con marcas de RA que permiten visualizar una escena completa virtual cada vez que se da vuelta a la página del libro, como si se tratara de un teatro en miniatura virtual, de ésta forma se aprecian con detalles los personajes y escenarios.

- **Publicitario:** también llamado Marketing, éste sirve para conquistar clientes (potenciales), vender más y mejor; obteniendo así más beneficios para

³<http://www.realidadaugmentada.info/nathan.html>

2.2. APLICACIONES DE REALIDAD AUMENTADA



Figura 2.7: Revista de IKEA en Realidad Aumentada



Figura 2.8: Reconocimiento de la estatua y aportando información característica con RA

el negocio. La RA permite interactuar con un producto (cambiando características como el color de un automóvil, decoración de interiores, probarse ropa) despertando de ésta forma el interés por él (ver figura 2.7). Se podría decir que algún día las grandes y costosas escenografías y utilerías pueden llegar a ser reemplazadas por esta tecnología de RA.

- **Cultural:** se emplea para mostrar información sobre objetos ó lugares de interés por un usuario. En especial en proyecto turísticos donde el usuario desea conocer los sitios de interés más representativos de un lugar.

En la figura 2.8 se aprecia como un usuario con el dispositivo móvil a través de la RA reconoce y obtiene información sobre una estatua que se encuentra ubicada dentro de un museo.

- **Medicina:** actualmente se ha visto grandes avances en el campo de la medicina, como apoyo a la visualización y entrenamiento para la cirugía, por ejemplo realizar una cirugía y que la RA permita superponer datos visuales sobre la operación a seguir; además el poder realizar programas de rehabilitación que ayuden al paciente a ir mejorando cada día.

2.2. APLICACIONES DE REALIDAD AUMENTADA



Figura 2.9: Realidad aumentada en las aulas de medicina

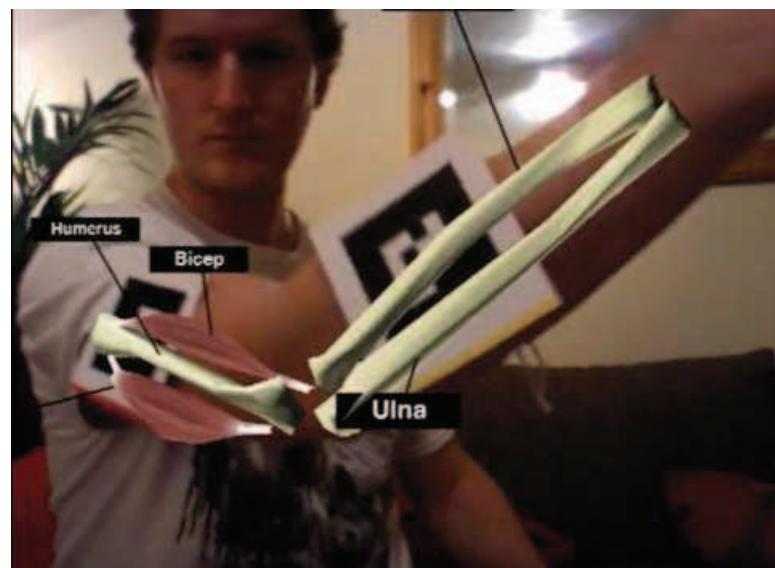


Figura 2.10: Aprender las partes del brazo con ayuda de marcadores para la RA

2.3. SISTEMAS OPERATIVOS MÓVILES

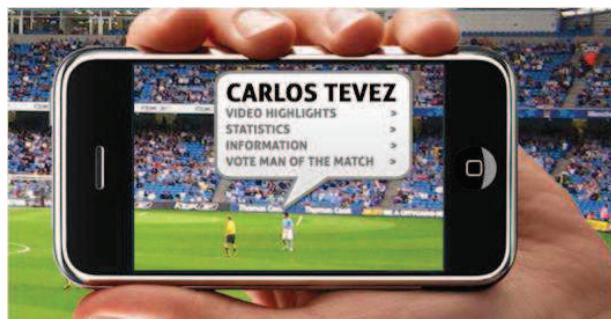


Figura 2.11: Obtener información de un jugador de fútbol con RA

También está la RA en conjunto entre la medicina y la educación, para un estudiante de medicina es vital conocer y aprender las partes del cuerpo humano y una forma tradicional para ello es mirarlo en un libro (información plana) ó mediante un muñeco de plástico (la mayoría de las veces es de coste alto ó tan sólo se consigue en bibliotecas) u otros medios de estudio; por ello uno de los grandes avances es disponer de Realidad Aumentada y poder ver las partes del ser humano cuando se reconoce a un humano. Como ejemplos se presentan algunas imágenes de anatomía con ayuda de la RA, ver la figura 2.9⁴ y la figura 2.10⁵.

- **Entretenimiento:** la RA aplica mucho en nuestro tiempo libre, dando nuevas posibilidades y maneras de ocio, como los juegos bien sea siendo parte de ellos ó presenciar un partido de fútbol y obtener información de los jugadores participantes, como el ejemplo que se presenta en la figura 2.11.

- Entre otros..

2.3. Sistemas operativos móviles

Actualmente existen muchas plataformas para móviles (iPhone, Symbian, Windows Phone, BlackBerry, Palm, Java Mobile Edition, Linux Mobile (LiMo), entre otros); sin embargo este trabajo se enfocará más en el sistema operativo Android (plataforma de desarrollo para este proyecto). También se describe de forma breve el sistema operativo IOS como comparativa de mercado y finalmente se mencionan las características, ventajas y desventajas de éstos dos sistemas operativos descritos en este apartado y como conclusión las características de la plataforma escogida para el desarrollo de éste proyecto.

⁴<http://www.somosmedicina.com/2013/08/realidad-aumentada-en-las-aulas-de.html>

⁵<http://www.learnar.org/>

2.3. SISTEMAS OPERATIVOS MÓVILES



Figura 2.12: Logo de iOS

2.3.1. iOS

Este sistema operativo ha sido desarrollado por Apple, en un principio cuando se lanzó al mercado el iPhone, tomo el nombre de iPhone OS. Posteriormente, con la aparición en el mercado de otros dispositivos basado en este sistema operativo (iPod Touch, iPad, AppleTV) fue renombrado a iOS. Una de las ventajas que tiene este sistema operativo es, como Apple fabrica el hardware como el sistema operativo del iPad, iPhone y el iPod Touch, tiene un buen nivel de integración; por lo que las aplicaciones aprovechan al máximo las prestaciones como la pantalla Retina, la interfaz Multi-Touch, el acelerómetro, el giroscopio, aceleración de gráficos y entre otros (ver figura 2.12(Inc, 2013)).

2.3.2. Android

Android es un Sistema Operativo además de una plataforma de Software basada en el núcleo de Linux. Diseñada en un principio para dispositivos móviles, Android permite controlar dispositivos por medio de bibliotecas desarrolladas o adaptados por Google mediante el lenguaje de programación Java (ver figura 2.13 tomado de (Android, 2013)). Android es una plataforma de código abierto.

Esto quiere decir, que cualquier desarrollador puede crear y desarrollar aplicaciones escritas con lenguaje C u otros lenguajes y compilarlas a código nativo de ARM (API de Android), por lo que en este caso la portabilidad está asegurada, tanto en el presente como en el futuro.

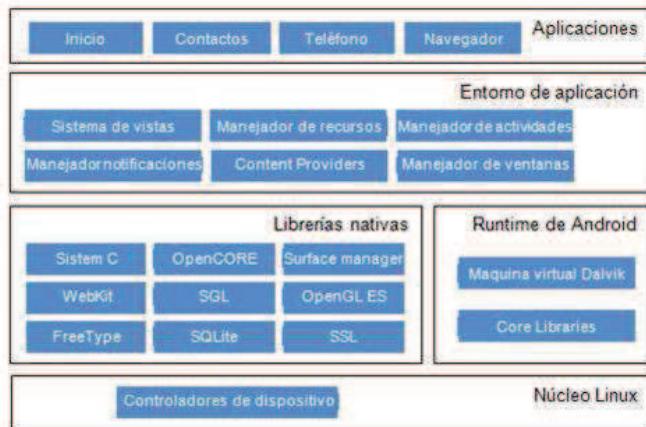
Inicialmente, Android fue desarrollada por Google Inc. aunque poco después se unió Open Handset Alliance, un consorcio de 48 compañías de Hardware, Software y telecomunicaciones, las cuales llegaron a un acuerdo para promocionar los estándares de códigos abiertos para dispositivos móviles (Gironés, 2013).

Sin embargo Android no ha sido diseñado exclusivamente para su uso en teléfonos y tabletas. Actualmente se puede encontrar relojes, cámaras, electrodomésticos y gran variedad de sistemas empotrados que se basan en este sistema operativo. Este hecho tiene sus evidentes ventajas, pero también va a suponer un esfuerzo adicional al programador. Ya que la aplicación ha de funcionar correctamente en dispositivos con gran variedad de tipos de entrada, pantalla, memoria,

2.3. SISTEMAS OPERATIVOS MÓVILES



Figura 2.13: Logo Android



Cuadro 2.2: Arquitectura Android

etc.; gracias a la arquitectura basada en componentes inspirados en internet, por ejemplo el diseño de la interfaz de usuario se hace en XML, lo que permite que una misma aplicación se ejecute en un móvil de pantalla reducida o en un TV.

La arquitectura de Android, se puede apreciar en el cuadro 2.3.2 tomado de (Gironés, 2013), donde se ve que está formada por cuatro capas; cada una de ellas basadas en software libre

- **Nucleo Linux:** formado por el sistema operativo Linux versión 2.6. esta capa proporciona servicios como la seguridad, manejo de memoria, multiproceso, pila de protocolos y soporte de drivers para dispositivos. Esta capa es la única que actúa como dependiente del hardware.
- **Runtime de Android:** basado en el concepto de máquina virtual utilizado en java. Google creó la máquina virtual dalvik que permite responder mejor a las limitaciones de poca memoria y procesador; optimizando así mejor los recursos.
- **Liberías nativas:** incluye un conjunto de librerías en C/C++ usadas en varios componentes de Android. Éstas se encuentran compiladas en código nativo del procesador y muchas de ellas utilizan proyectos de código abierto.

2.3. SISTEMAS OPERATIVOS MÓVILES

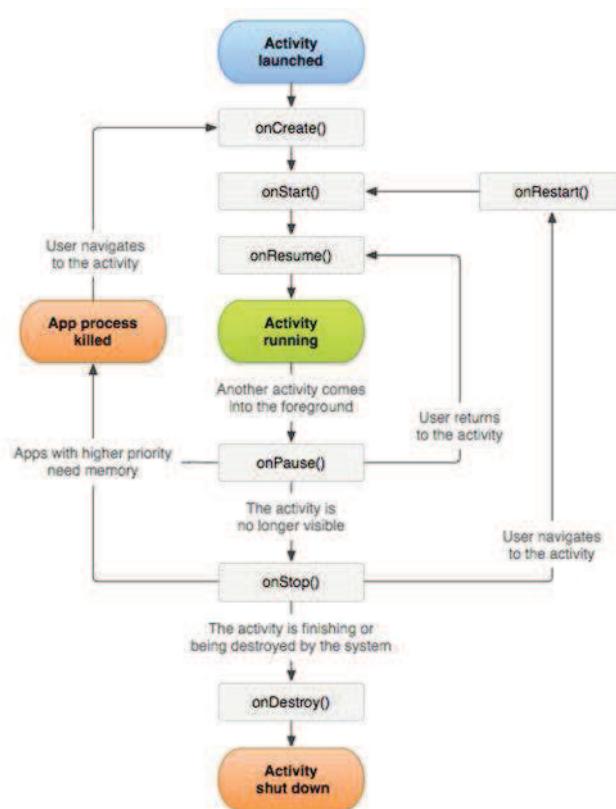


Figura 2.14: Diagrama de flujo del ciclo de vida de una aplicación Android

2.3. SISTEMAS OPERATIVOS MÓVILES

- **Entorno de aplicación:** proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones (sensores, localización, servicios, barra de notificaciones). Esta capa permite simplificar la reutilización de componentes, las aplicaciones pueden publicar sus capacidades y otras pueden hacer uso de ellas (sujetas a las restricciones de seguridad). Este mismo mecanismo permite a los usuarios reemplazar componentes.
- **Aplicaciones:** esta capa se encuentra formada por el conjunto de aplicaciones instaladas en la máquina Android. Todas las aplicaciones han de correr en la máquina virtual Dalvik para garantizar la seguridad del sistema. Normalmente las aplicaciones Android están escritas en Java. Para desarrollar aplicaciones en Java podemos utilizar el Android SDK. Existe otra opción consistente en desarrollar las aplicaciones utilizando C/C++. Para esta opción podemos utilizar el Android NDK (Native Development Kit).

En cuanto al ciclo de vida⁶ de una aplicación en Android (ver figura 2.14 tomado de (Android, 2013)); se compone de clases base Activity⁷ que a su vez definen una serie de eventos que están gobernados por el ciclo de vida de la misma Activity. Estos eventos son:

- **onCreate()**: Se ejecuta cuando la Activity es creada por primera vez
- **onStart()**: Se ejecuta cuando la Activity es visible al usuario.
- **onResume()**: Se ejecuta cuando la Activity empieza a interactuar con el usuario.
- **onPause()**: Se ejecuta cuando la actividad actual es pausada y otra actividad previa es desplegada en la pantalla.
- **onStop()**: Se ejecuta cuando la actividad ya no es visible para el usuario.
- **onRestart()**: Se ejecuta cuando la Activity ha sido detenida y esta reiniciando.

2.3.2.1. Orientación del móvil

El conocer la orientación del dispositivo móvil, permite que la realidad aumentada se muestre de forma creíble, colocando así los objetos de una manera lógica. Todo esto es posible a que la plataforma de Android define tres rotaciones para el dispositivo: pitch, azimuth y roll (ver figura 2.15⁸); denotándose así como el sistema de coordenadas cartesianas utilizado a continuación:

⁶<http://developer.android.com/reference/android/app/Activity.html>

⁷ *Activity*: Ventana que contiene elementos de interfaz de usuario con los cuales se puede interactuar con el mismo. Las aplicaciones pueden tener una o varias Activities

⁸<http://f.hatena.ne.jp/IchiRoku/20101219131236>

2.3. SISTEMAS OPERATIVOS MÓVILES

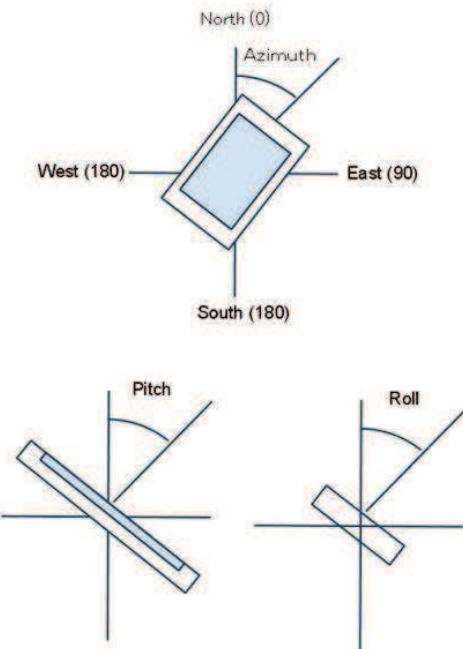


Figura 2.15: Orientación en Android

- El eje X se refiere al eje horizontal de la pantalla (pitch), este toma como valor del ángulo en grados es:

$$(-180 \leq pitch \leq 180)$$

- El eje Y apunta hacia la parte superior de la pantalla (azimuth), el valor del ángulo en grados es:

$$(0 \leq azimuth < 360), 0 = Norte, 90 = Medio, 180 = Sur, 270 = Oeste$$

- El punto del eje Z hacia el cielo cuando el dispositivo se encuentra acostado de espaldas sobre una mesa, el valor del ángulo en grados es:

$$(-90 \leq roll \leq 90)$$

2.3.2.2. Sensores de Android

La plataforma Android soporta una gran variedad de sensores, como se puede ver en la tabla 2.3 se muestra los diferentes tipos de sensores que puede tener un dispositivo móvil (los tipos de sensores varían según el dispositivo móvil utilizado)(Gironés, 2013). La clase central en el manejo de los sensores es

2.3. SISTEMAS OPERATIVOS MÓVILES

Tipo CONSTANTE	Utilidad	dim.	desde API
acelerómetro TYPE_ACCELEROMETER	medir aceleraciones por gravedad y cambios de movimiento	3	3
campo magnético TYPE_MAGNETIC_FIELD	brújula, detectar campos magnéticos	3	3
giroscopio TYPE_GYROSCOPE	detectar giros	3	3
orientación TYPE_ORIENTATION	indicar dirección a la que apunta el dispositivo	3	3
luz ambiental TYPE_LIGHT	ajustar iluminación pantalla	1	3
proximidad TYPE_PROXIMITY	si hay un objeto a menos de 5 cm (al hablar por teléfono)	booleano	3
presión atmosférica TYPE_PRESSURE	altímetro, barómetro	1	3
temperatura interna TYPE_TEMPERATURE	evitar sobrecalentamientos (obsoleto desde API14)	1	3
gravedad TYPE_GRAVITY	medir la aceleración debida a la gravedad	3	9
acelerómetro lineal TYPE_LINEAR_ACCELERATION	medir aceleraciones descontando la gravedad	3	9
vector de rotación TYPE_ROTATION_VECTOR	detectar giros	3	9
temperatura ambiental TYPE_AMBIENT_TEMPERATURE	medir la temperatura del aire	1	14
humedad relativa TYPE_RELATIVE_HUMIDITY	medir el punto de rocío, humedad absoluta y relativa.	1	14

Cuadro 2.3: Descripción de los tipos de sensores, de la clase Sensor

SensorManager, ésta clase dispone de constantes que representan los diferentes sensores posibles, así como de métodos para saber cuáles están instalados en él dispositivo actual y métodos para registrar eventos asociados a cada uno de ellos.

Los sensores se encuentran por defecto deshabilitados, siendo la habilitación y sucesiva deshabilitación responsabilidad de la aplicación. Un sensor habilitado no se deshabilita de manera automática cuando la pantalla está apagada, ya que dejar los sensores habilitados puede consumir rápidamente la batería del móvil. lo habitual es eliminar los eventos asociados a un sensor en el método *onPause* y volver a activarlos en el método *onResume*(Hristov, 2012).

Interpretar las lecturas del Sensor(Burnette, 2012)

Cada que se realiza algún movimiento con el dispositivo móvil, se genera un cambio de valor en el servicio sensor y éste llama al método *onSensorChanged()*. Todos los sensores devuelven una matriz de valores punto flotantes. El tamaño

2.4. SISTEMAS DE LOCALIZACIÓN

de la matriz depende del sensor particular, por ejemplo en el caso de necesitar simplemente un rumbo de brújula, puede emplearse el primer número devuelto por el sensor *TYPE_ORIENTATION*.

Para convertir las lecturas del sensor (especialmente del acelerómetro) en información significativa requiere de una serie de cálculos que dependen de lo que se quiere realizar. Para ello se tienen los siguientes tópicos:

- Las lecturas del acelerómetro son realmente frenéticas. Se necesita pulir los datos utilizando algún tipo de adaptación precisa, tratando de no adaptarlos en exceso ó sino la interfaz se volverá mas lenta.
- Los números del sensor aparece de forma aleatoria. Puede recibir varios seguidos, después una breve pausa y a continuación recibir otros tantos (no establecen una media).
- Se podría predecir el movimiento que realizará a continuación, si por ejemplo, las tres últimas lecturas muestran un viraje hacia la derecha, cada uno de ellos un poco más rápido que el anterior. Se puede adivinar de forma bastante precisa cuál será la próxima lectura y comenzar a responder en base a las predicciones.

Opción escogida

El desarrollar una aplicación en Android tiene la ventaja de que sirve en cualquier dispositivo, mientras que esta característica contrasta con la estrategia de Apple; En iOS se tiene que desarrollar una aplicación para iPhone y otra diferente para iPad, por lo que el tener dependencia de hardware y software de Apple, requiere un gran coste de desarrollo.

En conclusión para este proyecto se trabajará con el sistema operativo Android debido a que la plataforma es de carácter abierto, el acceso al kit de desarrollo es completamente gratuito e idóneo para un proyecto como éste, además de que se puede integrar con todos los servicios de google de ser necesario.

2.4. Sistemas de localización

Un sistema de localización permite establecer la posición de un individuo, vehículo ó cualquier otro objeto notable dentro de un escenario ó el mundo mismo; a través de coordenadas geoespaciales ó como la longitud, altitud y latitud (las generadas por el sistema de posicionamiento global GPS).

A continuación se presentan las aplicaciones más relevantes del mercado actual, que utilizan sistemas de geolocalización.

2.4. SISTEMAS DE LOCALIZACIÓN

2.4.1. PlaceLab

Es un proyecto Open Source desarrollado por Intel Research con el objetivo de crear un sistema de geolocalización basado en la técnica conocida como: Cálculo del punto central a partir de nodos predefinidos, este método no tiene la necesidad de elaborar un mapa de radio, aunque la precisión disminuye de forma drástica (LaMarca y cols., 2005).

Esta técnica es válida tanto para exteriores como para interiores de edificios.

Para ello se vale de fuentes de ondas de radiofrecuencia que pueden ser de tres tipos distintos:

- Puntos de acceso WiFi.
- Dispositivos Bluetooth.
- Antenas de radio GSM.

De esta forma, la información recibida en un momento dado se contrasta con bases de nodos fuente de ondas, conocidos como balizas, y sus coordenadas. Se trata de buscar que ondas se reciben en cada momento, donde se encuentran sus fuentes y a partir de esta información calcular el punto central de todas ellas.

En conclusión, el proyecto de PlaceLab siendo un proyecto de código libre, se puede llegar a apreciar más su uso en la localización en exteriores, cuando en situaciones carece de un sistema de posicionamiento por GPS es un poco impreciso en torno a pocos metros en la localización en interiores.

2.4.2. Proyecto Indoor Navigation System for Handheld Devices

Este proyecto se basa en el trabajo Application of Channel Modeling for Indoor Localization Using TOA and RSS, realizado por Ahmad Hatami para el Worcester Polytechnic Institute de Massachusetts (Hatami, 2006b), y trata de proporcionar un prototipo utilizando las ideas que se desarrollan en dicha investigación.

Estas ideas son principalmente la creación automática de un mapa de radio mediante procesado de mapas por raytracing (Hatami, 2006a), la detección de movimiento y la combinación de geolocalización por ondas Wifi y movimiento inercial mediante técnicas estadísticas.

El proyecto en sí está formado por dos componentes principales:

- Una serie de librerías para Matlab para el procesado de mapas y obtención de los mapas de radio.

2.5. SISTEMAS DE POSICIONAMIENTO EN INTERIORES

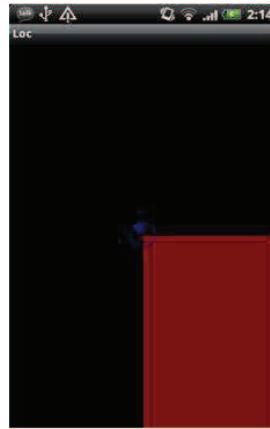


Figura 2.16: Ejecución de Indoor Navigation System for Handheld Devices

- Una aplicación para Android que implementa la localización en función de los mapas de radio, y la detección de pasos.

Como conclusión de este proyecto Indoor Navigation System con referencia al trabajo que se presenta en esta memoria, tiene la desventaja de que no fue fáctible coseguir el código fuente abierto al público sólo es posible de forma fraccionada dentro de los anexos de un documento, aunque el motivo fuerte por el que no es posible utilizarlo es el desconocimiento de la localización exacta de los puntos de acceso y mapas de radio.

2.5. Sistemas de posicionamiento en interiores

Los sistemas de posicionamiento local (LPS) son servicios basados en la localización (ejemplo figura 2.17), lo que permiten ofrecerle al usuario un servicio personalizado, en la mayoría de casos utilizan información geográfica; sólo depende si la tecnología del posicionamiento se encuentra al lado del cliente (ej. GPS, Wifi, etc) ó de lado del servidor (ej. servicios de posicionamiento suministrado por el operador de la red) y tecnología de comunicación de redes para transmitir información hacia una aplicación de localización basada en servicios (LBS) que pueda procesar y responder la solicitud.

Actualmente existen muchos tipos de sistemas de posicionamiento local, ver figura 2.18 tomada de (Hazas, 2004), estos diversos sistemas pueden variar según el tipo de diseño, tales como la tecnología física que los sustenta, la localización individual ó colectiva, la precisión que se pueda tener y métodos matemáticos de estimación

2.5. SISTEMAS DE POSICIONAMIENTO EN INTERIORES

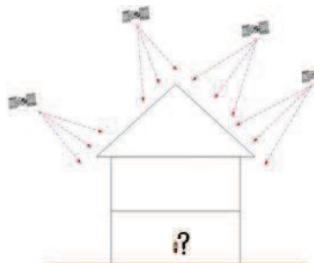
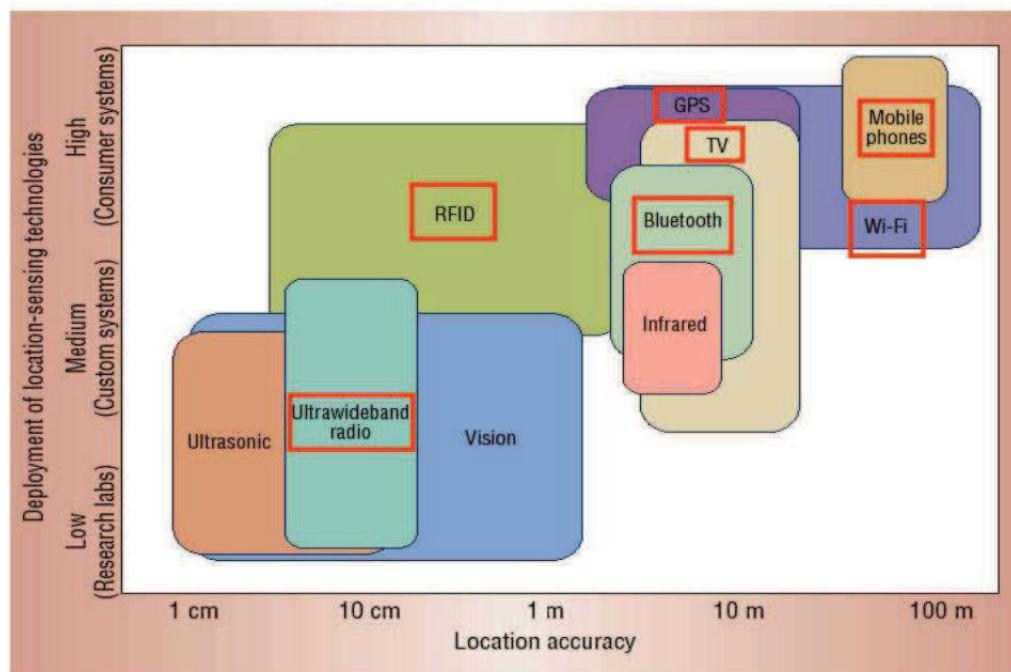


Figura 2.17: Sistema de localización indoor



Source: M. Hazas et al, "Location-Aware Computing Comes of Age", IEEE Computer, p. 95-97 (Feb. 2004)

Figura 2.18: Diversos sistemas de posicionamiento local

2.5. SISTEMAS DE POSICIONAMIENTO EN INTERIORES



Figura 2.19: Código QR



Figura 2.20: Códigos BIDI, utilizados al momento de acceder a algún espectáculo

2.5.1. Marcadores y marcas visibles

La plataforma de realidad aumentada se basa en algoritmos de reconocimiento de imágenes, qué a través de la cámara del ordenador ó móvil, son capaces de identificar rápidamente y en tiempo real cualquier tipo de imagen y objeto, a continuación se presentan algunos de los marcadores más relevantes que existen al día de hoy:

- **Código QR:** un código QR (Quick Response Barcode) es un sistema para almacenar información en una matriz de puntos o un código de barras bidimensional creado por la compañía japonesa Denso-Wave en 1994 (Froján y Lorenzo, 2011); se caracterizan por los tres cuadrados que se encuentran en las esquinas y que permiten detectar la posición del código al lector. Tiene como ventaja que estos códigos son gratuitos y existen aplicaciones en las que el usuario puede crear su propio QR. (figura 2.19).
- **Código BIDI:** los códigos BIDI son una particularidad de la compañía de Movistar creados en el 2008 (Guzmán Ullán, 2011). La aplicación Bidi de Movistar no permite usar el modo macro de la cámara en los móviles, por lo que en muchos casos el sistema no reconoce el código y da error. Telefónica Móviles y La Caixa, a través de Serviticket, han desarrollado el primer sistema que permite acceder a espectáculos con *ciberentradas*. A diferencia de los códigos QR, los BIDI no son gratuitos, el usuario tendrá que pagar por ello, ya que presta algún servicio en especial creado por la compañía (figura 2.20).
- **Marcas visibles:** consiste en la utilización de marcas visibles en las paredes que puedan ser detectadas mediante una cámara y reconocidas por un sistema de visión. Cada marca equivale a una posición específica almacenada en una base de datos. Por ejemplo en museos o la aplicación AndAR mencionada anteriormente. (Ver figura 2.3.).

2.5. SISTEMAS DE POSICIONAMIENTO EN INTERIORES

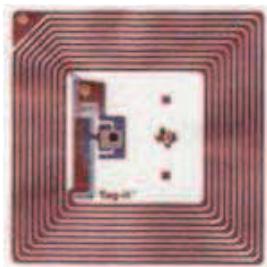


Figura 2.21: Etiqueta RFID

Ventajas:

- Al ser marcas localizadas se consigue gran precisión.

Desventajas:

- Localización no pasiva y no continua.
- Reconocimiento de marcas no trivial.
- Deja de funcionar si se interponen obstáculos entre la cámara y las marcas.

2.5.2. Etiquetas RFID

Dentro de la identificación por radio frecuencia las etiquetas RFID (Soto, Peinado, y Ortiz, s.f.), son el vehículo natural para aplicar un tag o realizar el tagging sobre los artículos, productos y embalajes que han de ser personalizados y controlados a través de radiofrecuencia. Hoy en día, la etiqueta es un requisito fundamental para indicar qué es un producto, sus características y es el vínculo de comunicación entre el fabricante y el usuario. (Ver figura 2.21).

- **Interior de un RFID tag** Tiene un circuito receptor-emisor pero reducido en un espacio de 0,3 x 0,3 mm² y una antena de dimensiones máximas de 100 x 100 mm². El interior del RFID es un dispositivo común que por sus diferentes formas de *packaging*, puede ser aplicado o insertado en objetos, agrupaciones animales ó personas con el objeto de que tengan un código que los identifique (Shepard, 2005). El tag emitirá el código cuando entre dentro del campo de acción generado por una antena RFID.
- **Clasificación de los RFID Tags** Los tags RFID se clasifican según los siguientes parámetros generales:
 - Según el sistema de alimentación del chip, así

2.5. SISTEMAS DE POSICIONAMIENTO EN INTERIORES

- tendremos tags pasivos, activos o semipasivos.
- Según su frecuencia operativa. UHF, HF o LH
- Según su aplicación
- Según la normativa ISO que deba cumplir

Estas etiquetas o tags RFID, al ser utilizadas como medio de localización interior, consistiría en la colocación a lo largo de todo el edificio de etiquetas RFID. Cada una de las etiquetas posee un identificador único que puede ser leído por radiofrecuencia, de forma que al ser detectadas por un lector RFID puede inferirse la posición a partir de una base de datos de identificadores y posiciones. Un claro ejemplo son las etiquetas antirrobo presentes en multitud de productos.

Ventajas:

- Sencillez

Desventajas:

- Necesidad de un gran número de etiquetas para que sea preciso, debido a su escaso alcance.
- Dificultad de implantación y poca escalabilidad.
- Necesidad de hardware externo al dispositivo y coste asociado.

2.5.3. Sistemas inerciales

El sistema de navegación inercial, es un sistema de navegación autónomo que no depende de medidas ni redes de localización externas. El objetivo de la navegación inercial es determinar la posición con la mayor precisión posible, a partir de las medidas de la IMU (Inertial Measurement Unit) (Prieto Tejedor y cols., 2012).

La IMU se compone de sensores inerciales como: acelerómetros, campo magnético y giróscopos (o en su defecto una brújula). Para realizar estos cálculos de la localización se miden las aceleraciones a las que el dispositivo está sujeto, mediante una doble integración respecto del tiempo se puede averiguar, en primer lugar, la velocidad del móvil en cada uno de los ejes de desplazamiento, y posteriormente el desplazamiento desde la última posición medida (Perez, por defender).

Ventajas:

- Puede ser un buen complemento a otras técnicas de localización.

2.5. SISTEMAS DE POSICIONAMIENTO EN INTERIORES

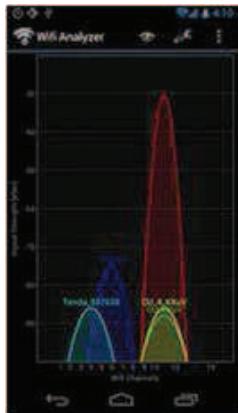


Figura 2.22: Fluctuación de la señal en redes Wifi

Desventajas:

- Necesidad de hardware de gran precisión.
- Error acumulativo.
- Insuficiente como único método de localización.

2.5.4. Cálculo del punto central a partir de nodos predefinidos

Consiste en la creación de una base de datos de puntos predefinidos, que pueden ser antenas de telefonía, puntos de acceso Wifi, dispositivos bluetooth, etc. (Figura 2.22) Estos nodos tienen asociadas en la base de datos sus coordenadas reales.

De esta forma, el dispositivo móvil comprueba periódicamente cuántos de estos puntos es capaz de detectar, y a partir de los nodos detectados infiere su posición mediante el cálculo del punto central a todos ellos, también conocido como Centroid o Centro Geométrico.

Existen diversos proyectos que utilizan esta tecnología en conjunción con bases de datos de puntos de acceso obtenidos por *wardriving* que permiten localizar un móvil con una precisión aceptable sin la necesidad de usar GPS. El ejemplo más claro es Placelab (LaMarca y cols., 2005), que se mencionó anteriormente en el apartado de los proyectos de localización.

Ventajas:

- No requiere hardware externo.
- Precisión aceptable en exteriores.

2.5. SISTEMAS DE POSICIONAMIENTO EN INTERIORES

Desventajas:

- Precisión insuficiente para interiores. Si el alcance a cada emisor es muy grande la precisión es muy baja incluso en exteriores.
- Inestabilidad.
- Necesita reconocimiento previo de los nodos.

2.5.5. Triangulación de la señal

Localización mediante Triangulación de la Señal, existen dos métodos de triangulación para sistemas basados en ondas de radio:

- **Triangulación por tiempo de llegada (TOA)** (Arias, Fuentes, y García, s.f.): Consiste en calcular el tiempo que tarda una onda desde que sale del dispositivo fuente hasta que llega al dispositivo destino. En base a dicha información, y utilizando al menos tres nodos fuente, se infiere la posición en que se encuentra el dispositivo móvil.
- **Triangulación por intensidad de señal (RSS)** (Palazón, Gozalvez, y Prieto, s.f.): Consiste en calcular la fuerza de la señal que llega de al menos tres dispositivos fuente distintos. En base a dicha información, se infiere la posición del dispositivo.

Ninguno de los anteriores métodos es aplicable a las ondas Wifi, ya que dichas ondas rebotan en el interior, de un edificio, produciendo reflexiones y refracciones que alteran el tiempo de llegada y la intensidad de la señal. Además surge el problema de la propagación multirayecto, ya que las ondas pueden llegar al dispositivo destino por múltiples caminos, produciendo tiempos de llegada y atenuaciones distintos.

Es posible aplicar estos métodos a las redes de telefonía, pero debido a la lejanía que hay entre las diferentes antenas de telefonía, la precisión que se consigue es muy baja y en cualquier otro caso es insuficiente para la localización en interiores.

Ventajas:

- Puede conseguir buenas precisiones.

Desventajas:

- No aplicable con el hardware disponible(Smartphone Android)
- Implementación costosa.

2.5. SISTEMAS DE POSICIONAMIENTO EN INTERIORES

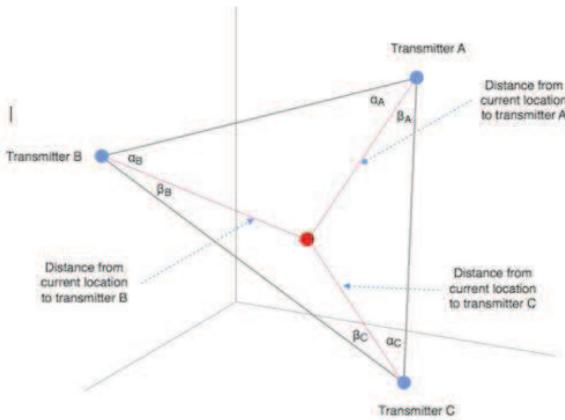


Figura 2.23: Triangulación de señal

2.5.6. Detección de movimiento

Esta técnica consiste en detectar, mediante el uso de acelerómetros, cuando la persona que porta el dispositivo está caminando, para posteriormente aplicar una velocidad de paso estándar en la dirección correcta tomado de (Bellón Alcarazo Sergio, 2011).

Ventajas:

- Efectivo como método auxiliar de localización.
- Aporta precisión extra.
- Hardware necesario disponible en teléfonos móviles actuales.

Desventajas:

- Error acumulativo, requiere ser reseteado periódicamente.
- Insuficiente como único método de localización.

2.5.7. Mapas de Radio Wifi

Consiste en la obtención, a partir de un proceso de captura de datos, de un mapa de radio (radiomap) de las ondas wifi de diferentes puntos de acceso en diferentes nodos previamente definidos (Ocana, Bergasa, Sotelo, Revenga, y Flores, s.f.). Este mapa de radio es utilizado posteriormente para inferir la posición del dispositivo mediante distintos tipos de algoritmos (Closest Neighbor y Redes Neuronales).

2.6. COMPONENTES BASE DEL PROYECTO

- Obtener lista de puntos de acceso AP Wifi: dirección MAC o SSID (Antoniades y Stephanedes, 1996).
- Definir los nodos, guardándose información de los niveles de señal de cada AP la información obtenida en las capturas conforma el radiomap

Ventajas:

- El mecanismo de entrenamiento asegura una buena precisión.
- Hardware necesario disponible en teléfonos móviles actuales.

Desventajas:

- Se requiere captura previa de datos.
- Los datos obtenidos para un dispositivo pueden no ser válidos para otro.
- Cambios en la estructura del edificio, de mobiliario, etc, pueden invalidar los datos de entrenamiento.

2.6. Componentes Base del Proyecto

A lo largo de éste capítulo se ha realizado un análisis del estado de la cuestión general sobre las aplicaciones de realidad aumentada, Android, sistemas y técnicas de localización en interiores; ahora se presenta el estado de la cuestión de los distintos componentes que se utilizarán a lo largo de la solución propuesta en ésta memoria.

2.6.1. Sistema MILES

El sistema MILES es un acrónimo de Modelos de Interacción centrados en el Lenguaje, Espacio y Semántica computacional; surge del proyecto conjunto coordinado por tres universidades españolas: Universidad Politécnica de Madrid, Universidad Complutense de Madrid y la Universidad de Sevilla. MILES tiene como propósito construir un sistema de navegación semántico en entornos virtuales y reales, orientado a recintos cerrados en el ámbito educativo, de forma adaptativa y personalizada.

Se basa en una arquitectura computacional (figura 2.24) diseñada para la representación e inferencia semántica de conocimiento sobre el espacio físico y el usuario. Este mecanismo ha sido implementado con ayuda de herramientas de ingeniería ontológica e integrado dentro de una plataforma de software multi-agente para la construcción de entornos virtuales, como lo es MAEVIF (Modelo

2.6. COMPONENTES BASE DEL PROYECTO

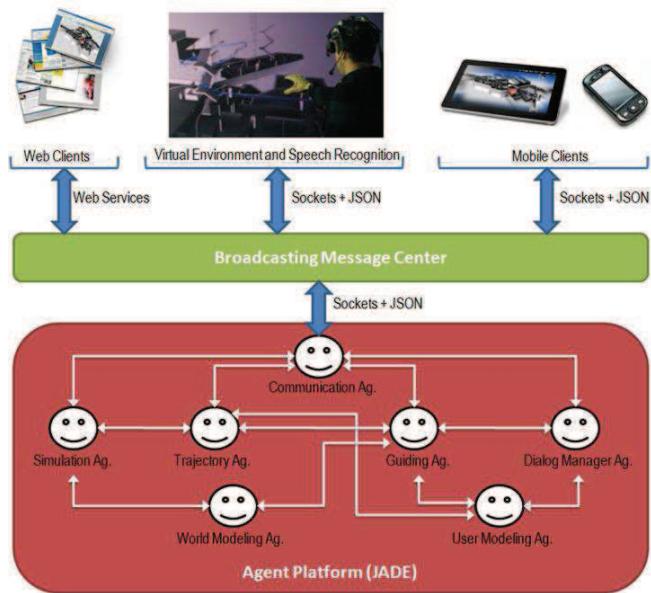


Figura 2.24: Arquitectura global del sistema MILES

para la Aplicación de Entornos Virtuales Inteligentes a la Formación) (Angelica De Antonio, 2005), comenzó en el año 2001 como proyecto de investigación financiado por el Ministerio de Educación y Cultura, en el Plan Nacional de I+D+i.

Este proyecto ha sido desarrollado en el Laboratorio Decoroso Crespo de la Facultad de Informática de la Universidad Politécnica de Madrid. La primera versión de MAEVIF se obtuvo en enero de 2004, luego se obtuvo una segunda versión de mejora a partir del curso académico 2005/06 y desde ese entonces se ha ido optimizando, con determinados elementos tecnológicos que necesitaban ser mejorados.

Sobre esta plataforma se dispone de varios agentes como por ejemplo, el de trayectoria encargado de generar distintas alternativas de camino hacia un destino utilizando un mecanismo híbrido y robusto para generar la ruta más óptima, teniendo en cuenta los datos del usuario; un agente mundo, que contiene toda la información semántica del mundo y el usuario; realiza inferencias sobre el continuo proceso de cambio de estado de ellos y un agente responsable de rastrear todas las acciones y movimientos que tiene el usuario. Además se encuentran los clientes externos como el: web, virtual y móvil; siendo este último el objetivo de este trabajo; implementar el cliente externo móvil e integrarlo a la plataforma MILES.

2.6.1.1. Ingeniería ontológica integrada en MAEVIF

La comunicación es el reto al que siempre se ha enfrentado el ser humano y se tiene diversas formas de abordarlo para obtener una mejor comunicación,

2.6. COMPONENTES BASE DEL PROYECTO

existen sistemas que pueden ayudar a resolver dichas diferencias, y la ontología es una parte importante del desarrollo del entendimiento compartido a través de un proyecto, en este caso MILES. Una ontología ⁹ representa el esfuerzo de formular un exhaustivo y riguroso esquema conceptual dentro de un dominio dado, típicamente una estructura de datos jerárquica que contiene todos los elementos relevantes y sus relaciones y reglas (reglamentos) en el dominio (G, 2004).

Existen varios lenguajes de ontologías disponibles, tales como: Resource Description Framework (RDF) (Klyne G, 2004), Web Ontology Language (OWL) (W3Schools, 2006), DARPA Agent Markup Language (DAML), Ontology Interchange Language (OIL) etc. Para el proyecto MILES el lenguaje ontológico que se utilizó fue OWL -DL (McGuinness y Van Harmelen, 2003)¹⁰ de la World Wide Web Consortium (W3C), se utilizó la herramienta visual Protégé (Noy y cols., 2001)¹¹ para su creación y mantenimiento.

El proyecto MILES (A, 2012), tiene como objetivo tomar ventaja de estos avances para apoyar la representación y la inferencia del conocimiento semántico sobre los mundos virtuales; el diseño de la representación e inferencia del conocimiento sobre el espacio físico y el usuario. Y es este el mecanismo que ha sido implementado con ayuda de herramientas de la ingeniería ontológica y a su vez a sido integrado dentro de la plataforma de software multiagente para la construcción de entornos virtuales, como lo es MAEVIF (Modelo para la Aplicación de Entornos Virtuales Inteligentes a la Formación).

2.6.2. Framework LooK

Look es un framework de Realidad Aumentada para Android creado para resolver los problemas comunes encontrados en el desarrollo de aplicaciones de este tipo. Éste ha sido un proyecto final de carrera de la Universidad Computense de Madrid, de la Facultad de Informática (Bellón Alcarazo Sergio, 2011).

Look es de código abierto (Licencia GPL v3) y con capacidad de extensión. Integra las siguientes características:

- Dibujado de gráficos en dos y tres dimensiones
- Construcción de Entidades representables en Realidad Aumentada
- Localización en Interiores de Edificios
- entre otros.

Actualmente Look ha sido usado en los siguientes desarrollos:

- Una galería de imágenes en 3D

⁹<http://www.artsci.wustl.edu/~philos/MindDict/ontology.html>

¹⁰<http://www.w3.org/2001/sw/WebOnt/>

¹¹<http://protege.stanford.edu/>

2.6. COMPONENTES BASE DEL PROYECTO



Figura 2.25: Pantalla principal de Look Social con menús

- Un mundo virtual
- Un juego interactivo
- Una aplicación para la creación de redes sociales con soporte para geolocalización llamada Look Social (ver figura 2.25)

Look emplea un módulo de localización donde combina, a su vez, los módulos de localización por wifi y un subsistema de navegación inercial (diagrama de flujo figura 2.26). Para ello, en primer lugar se inicializa el sistema de localización mediante la API *LocationManager*. Se utilizan tanto el subsistema de localización por wifi como el subsistema de navegación inercial de ésta forma se obtiene la máxima precisión en el posicionamiento del usuario dentro de un edificio.

2.6.3. Sensor Test

Sensorstest, es un trabajo de fin de carrera por (Perez, por defender) de la Universidad Politécnica de Madrid, se centra en la construcción de un sistema de navegación inercial que va a hacer uso de los sensores del dispositivo, en este caso una tablet Android; para reconstituir la ruta de una persona dentro de un edificio, conocida su localización y orientación inercial. Para la realización de éste método inercial utiliza el filtro de Kalman explicado en (Kalman, 1960), principalmente para estimar el estados del sistema, como es la eliminación del ruido de la señal. El método de navegación inercial consta de:

- Uso de sensores inerciales: acelerómetros, campo magnético y giróscopos (o en su defecto una brújula)
- Cálculo de la localización: medida de las aceleraciones del dispositivo
- Velocidad: doble integración respecto del tiempo (en cada uno de los ejes de desplazamiento)
- Desplazamiento: desde la última posición medida

2.6. COMPONENTES BASE DEL PROYECTO

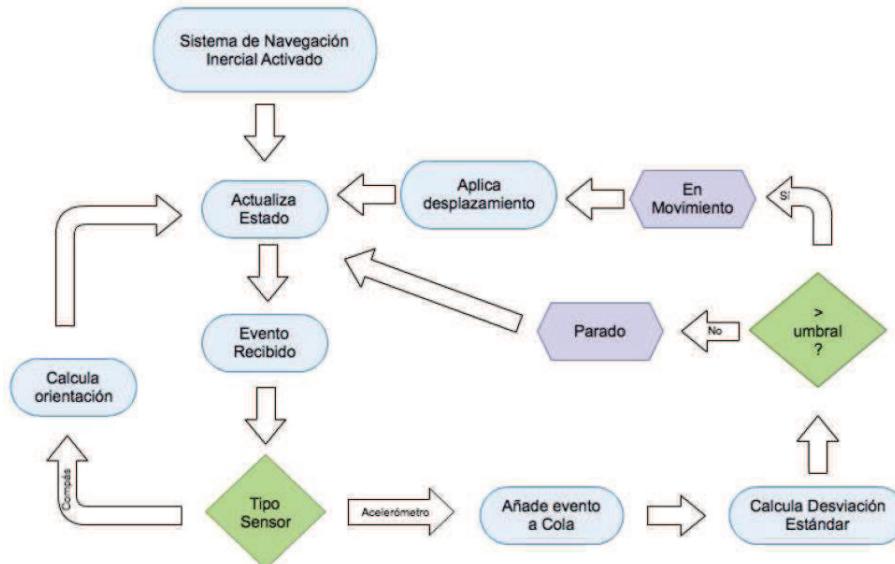


Figura 2.26: Diagrama de Flujo: Sistema de Navegación Inercial de Look

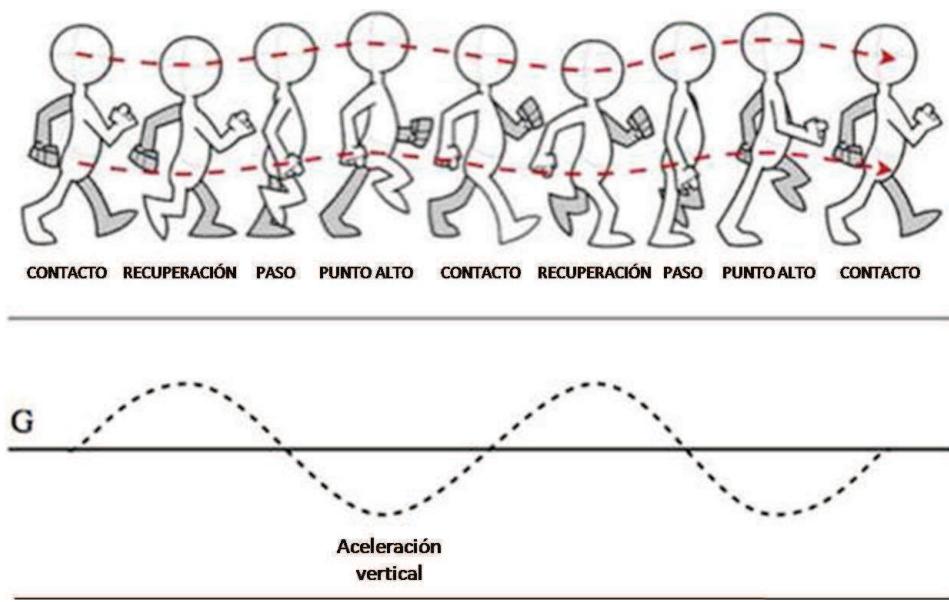


Figura 2.27: Comportamiento de la aceleración durante el ciclo de caminado

2.6. COMPONENTES BASE DEL PROYECTO

Además cuenta con una técnica alternativa de conteo de pasos (figura 2.27), como refuerzo al sistema de navegación inercial; ésta técnica le soluciona el problema de navegación inercial a corto plazo; pero al ser una técnica que no tiene en cuenta parámetros como la longitud de la zancada del usuario con el dispositivo, su precisión se escasea a largo plazo.

Para el conteo de pasos, (Perez, por defender) utiliza el módulo de la aceleración medida por el dispositivo, y la desviación respecto de la aceleración de fondo: la gravedad. Luego analiza la forma de la señal de la gravedad, y cada vez que detecta un pico alto y un pico bajo seguidamente, se estima que se ha producido un paso.

Ésta forma de contar pasos se ve que induce a muchos errores, puesto que los desplazamientos laterales, giros y otro tipo de movimientos voluntarios no pueden ser detectados como pasos. Como posible solución para evitar este problema, se debe limitar el conteo de pasos cuando se detectan determinadas circunstancias, a saber: giros y agitación del dispositivo. También se puede mejorar el conteo de pasos analizando la aceleración en el marco de referencia global.

En este marco de referencia, la aceleración que se produce por el movimiento de la cadera del usuario del dispositivo esté alineada con el eje de la gravedad. Si obviamos el resto de las componentes de la aceleración estamos eliminando los errores que se pueden cometer debidos a desplazamientos laterales o a giros (que producen fuerzas que se pueden interpretar como desplazamientos hacia adelante: la fuerza centrípeta). Cabe resaltar, que éste es un trabajo que se encuentra aún en fase de desarrollo y que por lo tanto se tiene un margen de error considerable.

Capítulo 3

Solución Adoptada

En éste capítulo se describirá como está estructurada la solución aportada; definiendo la arquitectura global y el diseño e implementación de la aplicación de realidad aumentada en interiores denominada GuIAR. Finalmente, se explicará el uso de la aplicación, en cuanto a su funcionamiento mediante un usuario.

3.1. Arquitectura Global

En ésta sección se presenta la descripción de la solución aportada al problema planteado, en la arquitectura global se incorporan los componentes base mencionados en el estado de la cuestión y se detalla el diseño e implementación de la aplicación GuIAR, como también su integración total con los componentes externos necesarios para el uso real del sistema propuesto.

En la figura 3.1 nos proporciona una visión general de como está estructurado el desarrollo realizado, conformado por la aplicación GuIAR y el intermediario REST; también se aprecia la interacción que se tiene con respecto al sistema MILES.

La aplicación de realidad aumentada en interiores **GuIAR**, ha sido implementada bajo la plataforma de Android (descrito en la sección 2.3.2), se ha utilizado la versión 2.3.x (Gingerbread)¹, lo cual permite acceder al soporte nativo del API de sensores como el giroscopio, vector de rotación, aceleración lineal, gravedad y otras características de mejoras para Android; y para este trabajo de fin de Máster es indispensable tener estos sensores en el Smartphone, en especial aquellos que son relevantes para la localización, como el campo magnético, acelerómetros, sensores de orientación y cámara; para llevar a cabo la Realidad Aumentada en Interiores.

Como herramienta de desarrollo se ha utilizado el Android SDK, que es el entorno de programación para este tipo de aplicaciones; se compone de un paquete

¹<http://developer.android.com/about/versions/android-2.3.html>

3.1. ARQUITECTURA GLOBAL

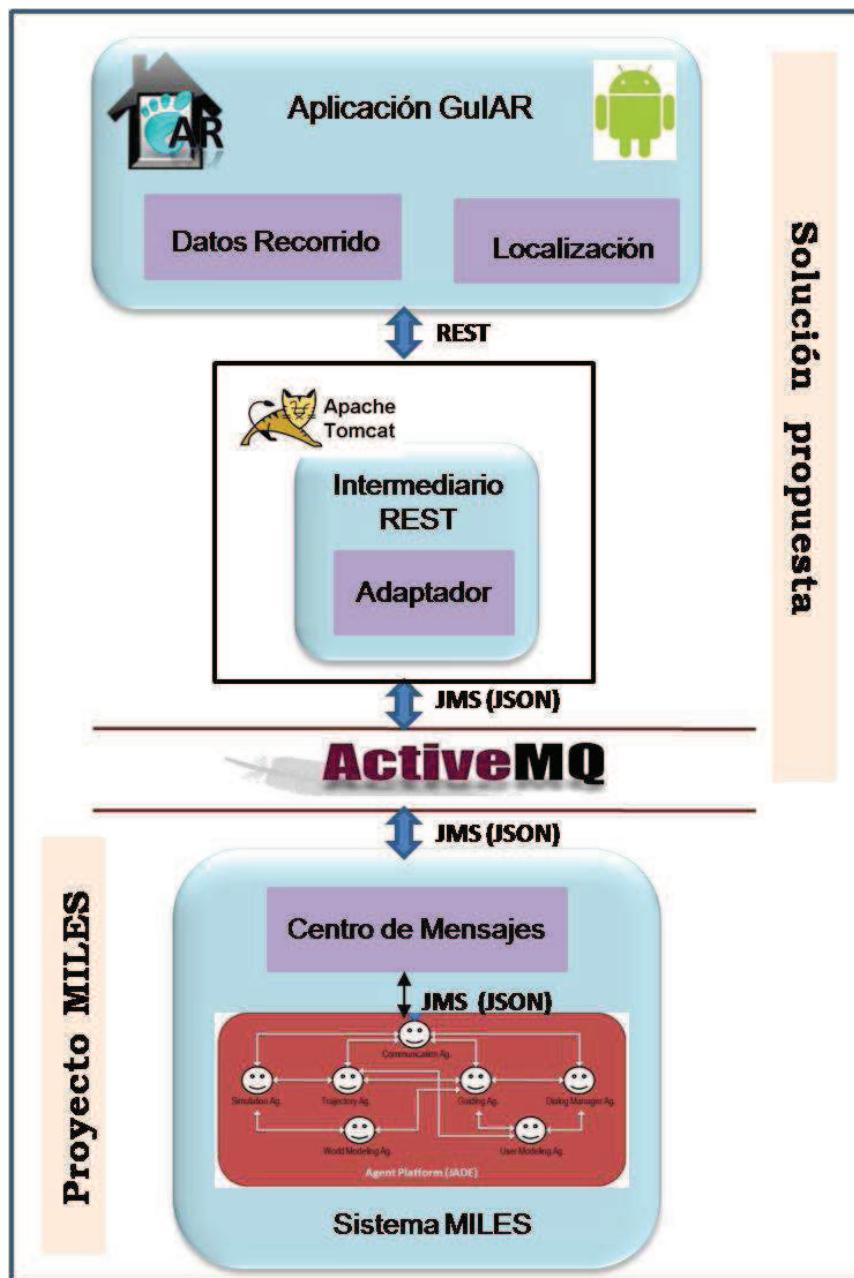


Figura 3.1: Arquitectura Global

3.1. ARQUITECTURA GLOBAL

de librerías de desarrollo y de un emulador de pruebas y tutoriales básicos; actualmente es posible la instalación de un plugging para el entorno de trabajo en Eclipse que facilita la programación y la compilación del Software. Cuando se desarrolla una aplicación en este entorno, se divide en lo que se llama **Activity** actividades; cada actividad lleva asociada una interfaz gráfica de usuario, por lo que obliga a lanzar una nueva para cambiar de interfaz. De esta forma se consigue tener un código con estructura modular y de forma ordenada.

GuIAR se encarga de aprovechar el campo de visión de la cámara del Smartphone, superponiendo la información en tiempo real que pueden servir de guía al usuario (texto y gráfico). Dicha información textual está relacionada con datos del recorrido ó tracking del usuario e instrucciones de trayectoria a seguir y puntos de interés² de manera interactiva que el usuario pueda elegir.

Para poder satisfacer lo mencionado en el párrafo anterior, GuIAR incorpora los componentes base que han sido descritos en el apartado 2.6, como son: el **Sensorstest** de (Perez, por defender) que nos permite obtener información física en tiempo real de un usuario que lleva el dispositivo móvil en un recorrido ó marcha habitual. Como datos de recorrido se tienen: el contador de pasos, la distancia y última velocidad. El componente es el **framework Look** de (Bellón Alcarazo Sergio, 2011), que nos aporta la localización del usuario dentro de un edificio. Estos dos componentes operan haciendo uso de los sensores propios del dispositivo móvil Android. Y por último la plataforma MAEVIF del sistema MILES que es el encargado de proveer la información correspondiente a la trayectoria que el usuario pueda realizar según el punto destino seleccionado.

Para lograr la comunicación entre el dispositivo móvil Android y la plataforma MAEVIF, se ha desarrollado como método de comunicación un intermediario REST (se describirá más adelante), que permitirá la publicación y suscripción de mensajería entre estas; mediante el uso del servicio de mensajería instantánea JMS³ (Sun Microsystems, 2012) y el formato de intercambio de datos JSON⁴ (Sun Microsystems, 2013) y (Crockford, 2006), a través del Bus de servicios ActiveMQ (Snyder, Bosnanac, y Davies, 2011)

Finalmente, de ésta forma se interactúa con la plataforma de agentes MAEVIF que a su vez contiene un componente llamado Centro de mensajes, encargado de proporcionar la información necesaria y simplificada entre éstas dos plataformas. A continuación se inicia la sección de diseño e implementación donde se describirá paso a paso lo que se ha mencionado en éste apartado.

²Punto de interés: es un punto de ubicación específica que un usuario puede encontrar útil o interesante. Un ejemplo de un punto de interés en el ámbito educativo, puede ser la Sala Wifi representado como un punto dentro del edificio 5 de la Universidad Politécnica de Madrid.

³JMS: es el API del servicio de mensajería de Java, una solución creada por Sun Microsystems para el uso de colas de mensajes. Este es un estándar de mensajería que permite a los componentes de aplicaciones basados en la plataforma Java2 crear, enviar, recibir y leer mensajes. También hace posible la comunicación confiable de manera síncrona y asíncrona

⁴JSON: es el acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos y tiene como ventaja que no requiere el uso de XML. Por lo que la simplicidad de este formato hace que sea más sencillo escribir un analizador sintáctico (parser) de JSON

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

3.2. Diseño e implementación de GuIAR

En ésta sección se explicará el diseño e implementación de cada uno de los componentes que se han desarrollado a largo de éste trabajo, para tener la aplicación de Realidad Aumentada en Interiores GuIAR, los componentes son: el Intermediario java que utiliza recursos REST y la Realidad Aumentada Indoor, ésta última a su vez se compone de tres partes: obtener la posición del usuario, la solicitud de datos al sistema MILES y el realizar el recorrido. Finalmente, se explicará el modo de uso de la aplicación GuIAR para que pueda ser usado por cualquier usuario.

3.2.1. Intermediario REST

Para el desarrollo de la comunicación entre el sistema Android de la aplicación GuIAR y la plataforma MAEVIF del sistema MILES, se plantearon varias estrategias, una de ellas fue con **MQ TelemetryTransport (MQTT)** (Hunkeler, Truong, y Stanford-Clark, 2008), es un protocolo de publicación/suscripción que permite el intercambio de mensajes para aplicaciones especializadas en pequeños dispositivos que deben tolerar un reducido ancho de banda y una comunicación poco fiable, este utiliza TCP/IP.

Los mensajes son reducidos gracias al pequeño tamaño de las cabeceras del protocolo y la carga del mensaje en matriz de bytes. Las cabeceras comprimen una cabecera fija de 2 bytes y hasta 12 bytes de otras cabeceras variables. El protocolo utiliza 12 bytes de cabeceras variables para suscribirse y conectarse y sólo 2 bytes para cabeceras variables para la mayoría de publicaciones(Corporation, 2010)

Después de haber realizado varias pruebas de conexión con MQTT⁵, se trató de establecer comunicación con la plataforma MAEVIF a través de colas y tópicos (queue & topic) que es lo que actualmente maneja MILES, y es aquí donde el proceso de distribución de carga de tareas se vio afectado, debido a que este protocolo liviano solo utiliza publicación y suscripción por tópicos y no por colas; por lo que la distribución entre suscriptores no se daría de manera equitativa; debido a que cada suscriptor es tratado igual y cada mensaje publicado en un tópico irá a cada suscriptor, y por cada suscripción no esperará respuesta para ser devuelto a quién lo envió⁶.

Debido a que con el protocolo MQTT no cumplió con la funcionalidad requerida, se decidió buscar otra alternativa que no afectara en lo más mínimo la plataforma de MILES. Es aquí donde se propone la idea de crear un intermediario java que utilice recursos REST (REpresentational State Transfer), éste es un estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la Web de (Fielding, 2000).

Éste intermediario adapta el flujo de información para que sea compatible con Android como con JMS permitiendo así la interacción del sistema Android con

⁵<http://mqtt.org/>

⁶<http://knolleary.net/2012/04/11/queuing-with-mqtt/>

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

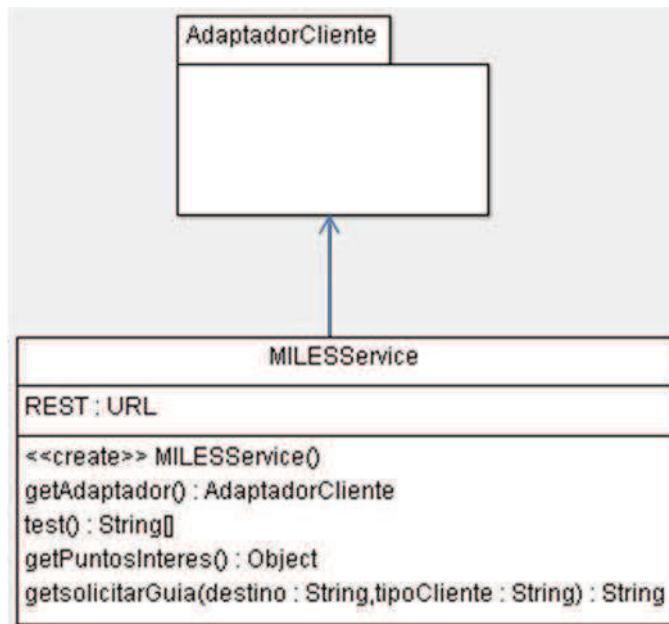


Figura 3.2: Diagrama de clases del intermediario REST

la plataforma MAEVIF a través del bus de servicio ActiveMQ⁷ y el centro de mensajes por medio de un adaptador mediante Sockets, de esta forma se puede escribir datos en formato JSON (formato que actualmente utiliza el sistema MILES para la suscripción y recepción de mensajes) directamente sobre el canal del socket y luego esos datos son leídos a través de un intermediario java con servicio REST por el cliente externo móvil que se detallará mas adelante.

En la figura 3.2, se muestra el diagrama de clases del intermediario REST, que a su vez se compone del paquete AdaptadorCliente (el mismo adaptador cliente que tiene el sistema MILES), encargado de realizar la conexión con el ActiveMQ. Los servicios Web que son basados en REST lo que intentan es emular el protocolo HTTP o similares mediante operaciones estándar (e.g. GET, POST, PUT,...) y es esto el objetivo la interacción con los recursos, el desarrollo del intermediario Web se ha realizado teniendo en cuenta el formato de datos JSON que utiliza actualmente la plataforma MAEVIF .

En la figura 3.1 se muestra de forma gráfica la arquitectura global, conformada por la solución propuesta (Intermediario REST y la aplicación GuiAR) y la interacción con el proyecto MILES a través del bus de servicio ActiveMQ.

Este intermediario tiene como ventajas, que se puede ubicar en el mismo sitio donde reside el servidor MILES (dentro del Tomcat) y otra ventaja es que si uno de los procesos se cae, la cola de mensajes asociado a ese proceso seguirá recibiendo mensajes, y cuando el proceso reviva, recibirá todos los mensajes que se

⁷<http://activemq.apache.org/>

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

perdieron, en orden FIFO (primero en entrar, primero en salir); de ésta forma se reduce la probabilidad de falla por comunicación entre las dos plataformas.

En cuanto al código fuente la clase del intermediario tiene como nombre: *MILESService.java*, se desarrolló utilizando el API Jersey (Sandoval, 2009) de JAVA⁸. Se crea bajo el paquete *es.upm.fi.ls.decoroso.miles.rest* importando la clase *AdaptadorCliente* del paquete java: *es.upm.fi.ls.decoroso.cliente.adaptador.conexion* (clase que sirve como intermediario entre un cliente y el Centro de Mensajes que permite la suscripción y publicación de mensajes a través de la plataforma MAEVIF).

A continuación se describe cada uno de los métodos que se encuentran en esta clase MILESService (tabla 3.1):

- **getAdaptador:** método privado de la clase, que permite indicarle al centro de mensajes de la plataforma MAEVIF, cual es el tipo de cliente (en este caso pueden ser tres tipos: ClienteGrafico, ClienteVoz y ClienteMóvil), se indica el tipo de idioma preferente del cliente y que entorno debe cargar la plataforma de agentes.
- **getPuntosInteres:** método público de tipo objeto, que permite recibir un listado de los puntos de interés filtrados por el entorno.
- **solicitarGuia:** método privado, que permite inicializar la actividad, el escenario, define el destino al cual quiere ir el usuario y consulta la trayectoria a seguir, indicando todos los puntos de coordenadas hasta su destino.

3.2.2. Realidad Aumentada Indoor

Uno de los objetivos importantes de este proyecto, es el desarrollar una técnica de realidad aumentada para la orientación personalizada en espacios interiores a través de dispositivos móviles, para satisfacer ésta solución adoptada se propone dividirla en tres partes (ver la solución propuesta de la figura 3.1):

1. Posicionamiento indoor, se describirá el método utilizado que permite obtener la posición del individuo a través de los sensores del dispositivo móvil en un recinto cerrado.
2. Interacción con MILES, Los métodos que se han desarrollado para solicitar la ruta personalizada a través del sistema MILES mediante el intermediario REST
3. Por último se explicará el tipo de información que se muestra como realidad aumentada, una vez se inicie la sesión del recorrido de la persona.

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

```
@Path("/rest")
public class MILESService {

    private Mundo m;

    public MILESService(){
        m=new Mundo();
    }

    private AdaptadorCliente getAdaptador(){
        return AdaptadorCliente.getInstancia( AdaptadorCliente.CLIENTE_GRAFICO, "EN", "guia_miles_simple");
    }

    //obtenemos la lista de puntos de interes que tiene MILES
    @GET
    @Path("/puntosinteres")
    @Produces(MediaType.APPLICATION_JSON)

    public HashMap<String, PuntoInteres> getPuntosInteres(){
        AdaptadorCliente adaptador=getAdaptador();
        try{
            Object[] puntos=adaptador.getListaPuntosInteres();
            adaptador.finalizar();
            return puntos[1];
        }catch(Exception e){
            adaptador.finalizar();
            return e.getMessage();
        }
    }
}
```

Cuadro 3.1: Clase: MILESService, fragmento de código del método getPunto-
sInteres

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

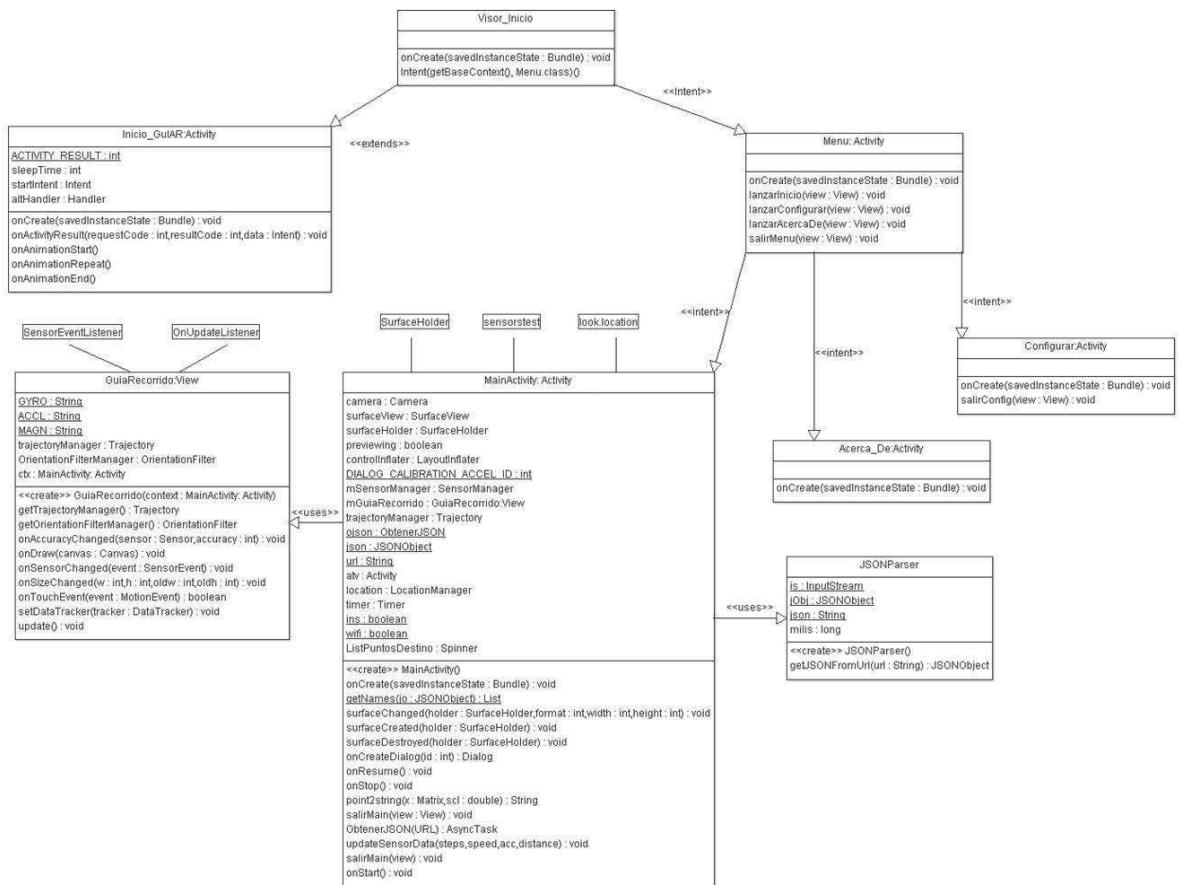


Figura 3.3: Diagrama de Clases de la aplicación GuIAR

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

En la imagen 3.3 se tiene el diagrama de clases, la cual contiene todos los objetos involucrados en el desarrollo de la realidad aumentada en interiores, se tienen las siguientes:

- Las clases *Visor Inicio* e *Inicio GuiAR*, realizan el arranque de la aplicación mediante una animación corta de 2 milisegundos dando la bienvenida.
- La clase *Menu* nos presenta un panel con diferentes opciones por las que el usuario podrá interactuar con la aplicación GuiAR, se tiene una clase activity por cada una:
 - La clase *Acerca De* muestra una breve descripción de la aplicación
 - La clase *Configurar* permite configurar la dirección URL donde estará ubicado el intermediario REST
 - La clase *MainActivity* es la opción que da inicio a la aplicación GuiAR (después de haber configurado la URL), ésta es una de las clases mas importante de éste desarrollo, pues en ella se integran todos los componentes que permite obtener la información relacionada al posicionamiento indoor y a los datos pertinentes del recorrido del usuario que se visualizan sobre la cámara del smartphone.
Todos estos recibiendo información de los sensores del dispositivo móvil, utilizados a través de la librería LocationManager de Look y del sensorstest; además de la recepción y emisión de datos de la plataforma de MAEVIF a través del intermediario REST; que permiten realizar la guia personalizada al usuario.
 - La clase *GuiaRecorrido* es donde se incorpora el componente SensorTest, en ella se realizan los diferentes cálculos matemáticos para obtener la velocidad, distancia recorrida y el conteo de pasos.
 - La clase *JSONParser* permite parsear el objeto JSON obtenido por la plataforma MAEVIF a través del intermediario REST, de tal forma que se pueda obtener una cadena sencilla de caracteres y poder ser usada en el sistema operativo Android.

3.2.2.1. Posicionamiento Indoor

Para obtener la posición del usuario dentro de un edificio con la ayuda de un dispositivo móvil, se incorporará el componente de Look ya mencionado en el apartado 2.6.2, como parte de la solución adoptada. Se utilizará el método de navegación inercial lo cual tiene como ventaja ser un método independiente que no requiere de elementos externos y permite detectar cuando un usuario está caminando y luego aplicar una velocidad estándar de paso en la dirección adecuada. Para ello se necesita de información de diversos sensores propios del dispositivo móvil (sensores descritos en la tabla 2.3):

⁸<http://jersey.java.net/>

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

- Acelerómetro: para detectar aceleraciones y movimientos
- giróscopo: para detectar la orientación en el espacio

Para detectar cuándo el usuario está o no en movimiento, Look realiza el cálculo de magnitud total por cada evento tomado de la aceleración, calcula la raíz de la suma de los cuadrados de cada uno de los ejes de coordenadas (X,Y,Z):

$$M = \sqrt{\sum_{i=1}^N v[i]^2}$$

Si la desviación estándar está por encima del umbral significa que el dispositivo está en movimiento, en caso contrario es que el dispositivo no está en movimiento.

De manera análoga, a intervalos de tiempo fijados se actualiza la información de la dirección del dispositivo a partir de los sensores disponibles. En caso de que el dispositivo se encuentre en movimiento, se aplica una velocidad estándar de paso en la dirección correspondiente proporcional al tiempo transcurrido desde el último evento de aceleración (Bellón Alcarazo Sergio, 2011).

Look nos presenta el siguiente diagrama de clases del sistema de navegación inercial figura 3.4 Tomada de (Bellón Alcarazo Sergio, 2011), donde se muestran las diferentes clases que lo componen,

- La clase *LocationProvider* es la encargada de centralizar la funcionalidad del subsistema de navegación inercial, registrando los listener para los distintos tipos de eventos y procesándolos según sea necesario.
- *DeviceSensor* se encarga de aplicar el algoritmo de detección de movimiento.
- *Positioning* es la clase que mantiene la información de la posición, el desplazamiento relativo y la detección/no detección de movimiento.
- *InertialNavigationSystem* se encarga de realizar los cálculos relativos al desplazamiento producido.
- *Motion* representa la información de cada uno de los desplazamientos relativos.

A continuación se presenta el diagrama de secuencia del sistema de navegación inercial de Look en la figura 3.5, en él se describe el comportamiento desde que se recibe un nuevo evento de aceleración mediante el método *getPosition()* hasta que la aplicación obtiene la nueva posición del usuario,

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

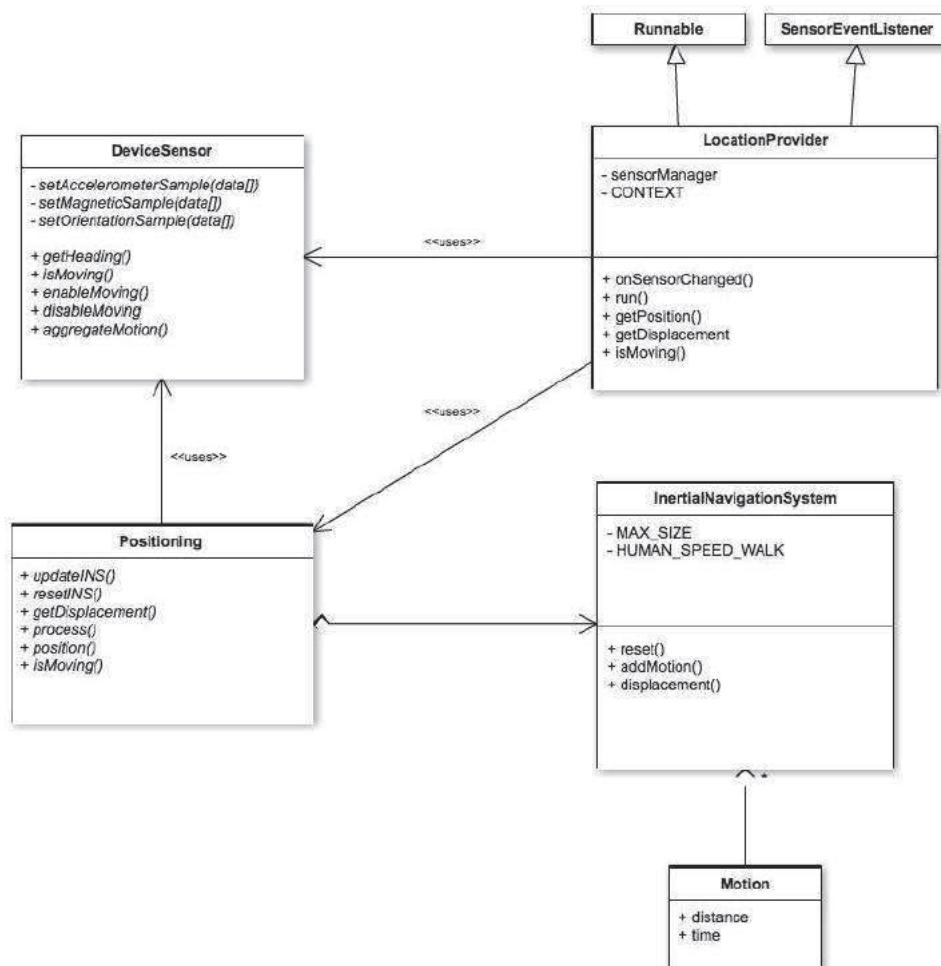


Figura 3.4: Diagrama de Clases del sistema de navegación inercial de Look.

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

1. La aplicación crea un nuevo LocationProvider, que registra listeners sobre los sensores y permanece a la espera de nuevos eventos. Además, inicializa el sistema de navegación inercial.
2. Se recibe un nuevo evento de aceleración y se notifica tanto a Positioning como a DeviceSensor.
3. DeviceSensor aplica el algoritmo de detección de movimiento. Si se detecta movimiento Positioning envía un mensaje a InertialNavigationSystem para que procese el nuevo evento desplazamiento relativo producido.
4. La aplicación llama a process para calcular la nueva posición en base a los desplazamientos relativos y obtiene la posición final.

IMPLEMENTACIÓN

Toda la implementación se ha realizado para el sistema operativo Android versión 2.3 o posterior. En nuestra clase principal *MainActivity* se ha incorporado la librería de localización de Look, para integrar la localización en nuestra aplicación, en primer lugar se debe indicar si se utilizará el sistema de localización por Wifi, sistema de navegación inercial ó ambos de manera combinada.

En éste trabajo se utilizó el sistema de navegación inercial, para detectar cuándo la persona se encuentra caminando y aplicar el desplazamiento; y para ello se deberá inicializar la clase *LocationManager*⁹ y luego se consulta si la persona está caminando a intervalos de tiempo fijos mediante un Timer, toda ésta parte dentro del método Create del Activity. Luego se hace el llamado al método *getPosition* que permitirá obtener las coordenadas (X,Y,Z) correspondiente Z a la planta del edificio donde se encuentra el usuario; después dentro de un hilo de ejecución se mantiene actualizando la posición cada 2 milisegundos, tal como se presenta en el cuadro 3.2. Además se ha implementado un sistema de logs informativos como fuente de datos para la realización de pruebas.

Una vez hecho esto, la localización se puede iniciar y detener mediante los métodos *start()* y *stop()*. Esta clase se encarga de inicializar y parar los servicios de localización necesarios, actualizar los datos de la posición, combinarlos si es necesario y proporcionar acceso a los mismos desde la aplicación.

3.2.2.2. Interacción con MILES

En ésta sección se explica como se ha integrado los métodos empleados para lograr la comunicación con la plataforma MAEVIF del sistema MILES con la aplicación GuIAR de sistema Android. Para cumplir con éste objetivo de recepción y emisión de datos con la plataforma MAEVIF, se ha hecho uso del

⁹es la interfaz con el sistema de localización

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

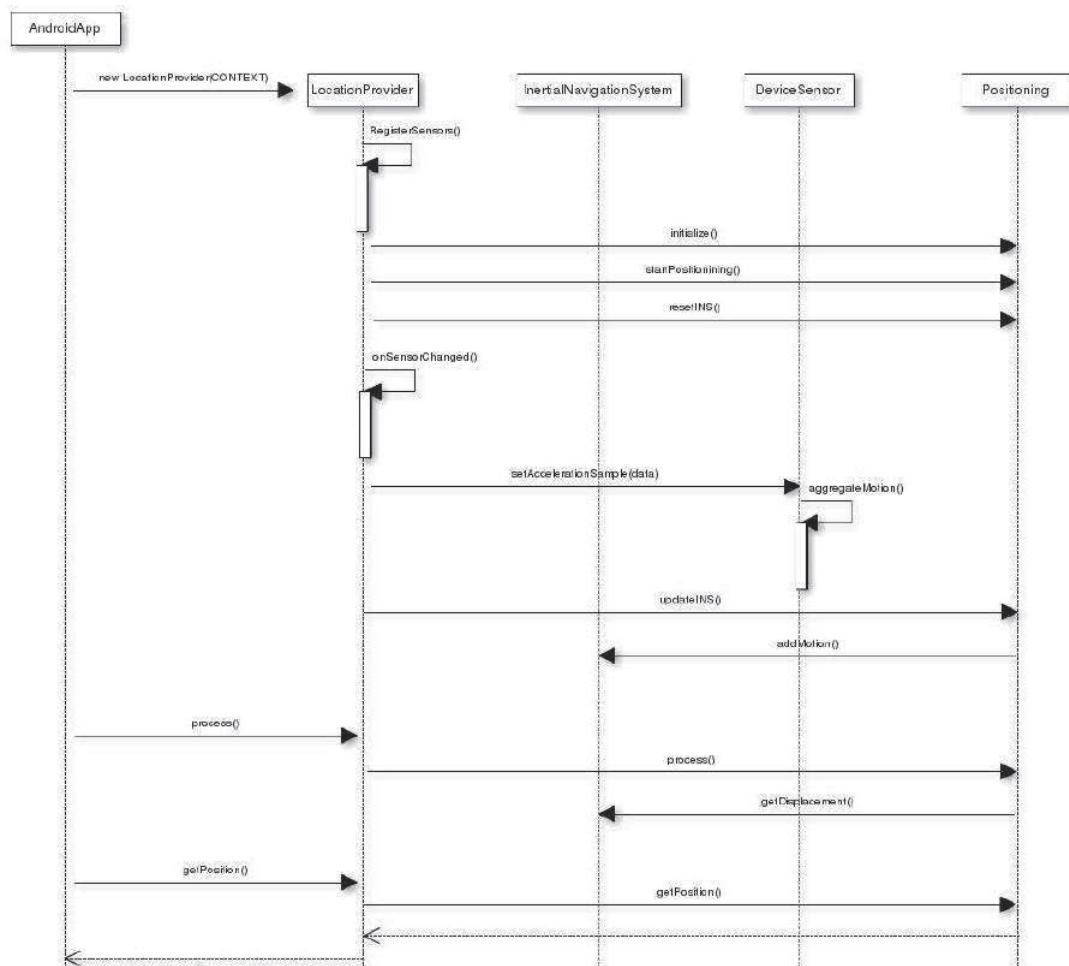


Figura 3.5: Diagrama de secuencia del sistema de navegación inercial

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

```
import es.ucm.look.location.LocationManager;

public class MainActivity extends Activity implements SurfaceHolder.Callback{

    //Variables necesarias para obtener coordenadas de Look!
    LocationManager location = null;
    Timer timer = null;
    public static boolean ins;
    public static boolean wifi;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Obtener la posición del usuario adicionando el componente Look!
        ins= true;
        location = new LocationManager(getApplicationContext(), ins, wifi);
        location.start();
        timer = new Timer();

        TimerTask timerTask = new TimerTask() {
            public void run() {
                runOnUiThread(new Runnable() {
                    public void run() {
                        ((TextView) findViewById(R.id.textCoord)).setText(String.
                                valueOf(location.getPosition().toString()));
                        //Transformación de coordenadas de Look a MILES
                        X_user=location.getPosition().x*100.0;
                        Y_user=location.getPosition().y*100.0;
                        Z_user=location.getPosition().z*100.0;

                        double fixed_orientation=0.0;
                        synchronized (orientacion) {
                            if(orientacion+90.0<360.0)
                                fixed_orientation=orientacion+90;
                            else{
                                fixed_orientation=(orientacion+90) %360.0;
                            }
                        }
                    }
                });
            }
        };
        timer.scheduleAtFixedRate(timerTask, 0, 500);
        super.onResume();
    }
}
```

Cuadro 3.2: Código fuente de la Clase MainActivity donde se incorpora el método INS

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

intermediario REST (descrito anteriormente en el apartado 3.2.1) que es llamado dentro de la clase principal *MainActivity* y para parsear¹⁰ la información obtenida se utiliza la clase *JSONParser* representados en el diagrama de clases de GuIAR figura 3.3

En el diagrama de secuencia figura 3.6 representa la interacción con el sistema MILES y la aplicación Android GuIAR, en él se muestra el comportamiento desde que se realiza el llamado al método para obtener el objeto JSON con los puntos de interés a través del intermediario REST, luego éste realiza la conexión con la plataforma MAEVIF a través del adaptador del cliente, éste último realiza el proceso de conexión, suscripción y obtención de datos, a través del centro de mensajes hasta que llega la información final a la aplicación Android, para ser analizado el objeto JSON recibido a través de la clase *JSONParser*, devolviendo la clave y el valor de los puntos de interés y finalmente en el método *getNames* obtener sólo los nombres de los puntos de interés, para ser mostrados en la lista desplegable que se encuentra en el campo de visión de la cámara y así el usuario poder elegir el punto destino de interés.

Para los diferentes métodos que se invocan a través del intermediario REST al sistema MILES, es el mismo diagrama de secuencia que el presentado en la figura anteriormente descrito, para obtener los puntos de interés (*solicitaPuntosInteres*), sólo cambia el nombre del método a invocar como: *solicitudGuia*, el cual devuelve las indicaciones que debe seguir el usuario hasta el punto final, dichas indicaciones están conformadas por un conjunto de puntos de coordenadas desde el origen hacia el destino y de igual forma para los demás métodos mencionados en el apartado 3.2.1.

IMPLEMENTACIÓN

Todos los componentes de la aplicación Android, como las actividades y los servicios se ejecutan en el mismo hilo de ejecución, llamado hilo principal, main thread o GUI thread, que como éste último nombre indica también es el hilo donde se ejecutan todas las operaciones que gestionan la interfaz de usuario de la aplicación.

Dentro de éste hilo principal una vez obtenida la posición del usuario dentro del edificio (explicado en la sección anterior), se conecta el cliente externo móvil GuIAR con el servidor del centro de mensajes de la plataforma MAEVIF, a través de la clase *AdaptadorCliente* incorporada en el intermediario REST (ver código fuente en el cuadro 3.1), ésta recibe como parámetros el tipo de cliente en éste caso gráfico, el idioma de preferencia y el entorno. Luego de conectarse se envía primero la posición de partida a la plataforma MAEVIF a través del intermediario REST.

Dentro de la clase principal *MainActivity* se tienen los siguientes objetos que permiten la recepción y emisión de datos al sistema MILES:

- Clase *ObtenerJSON* (cuadro 3.3), ésta clase permite obtener el objeto JSON, es decir la información devuelta por el intermediario REST y luego

¹⁰parsear: interpretar el objeto JSON recibido

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

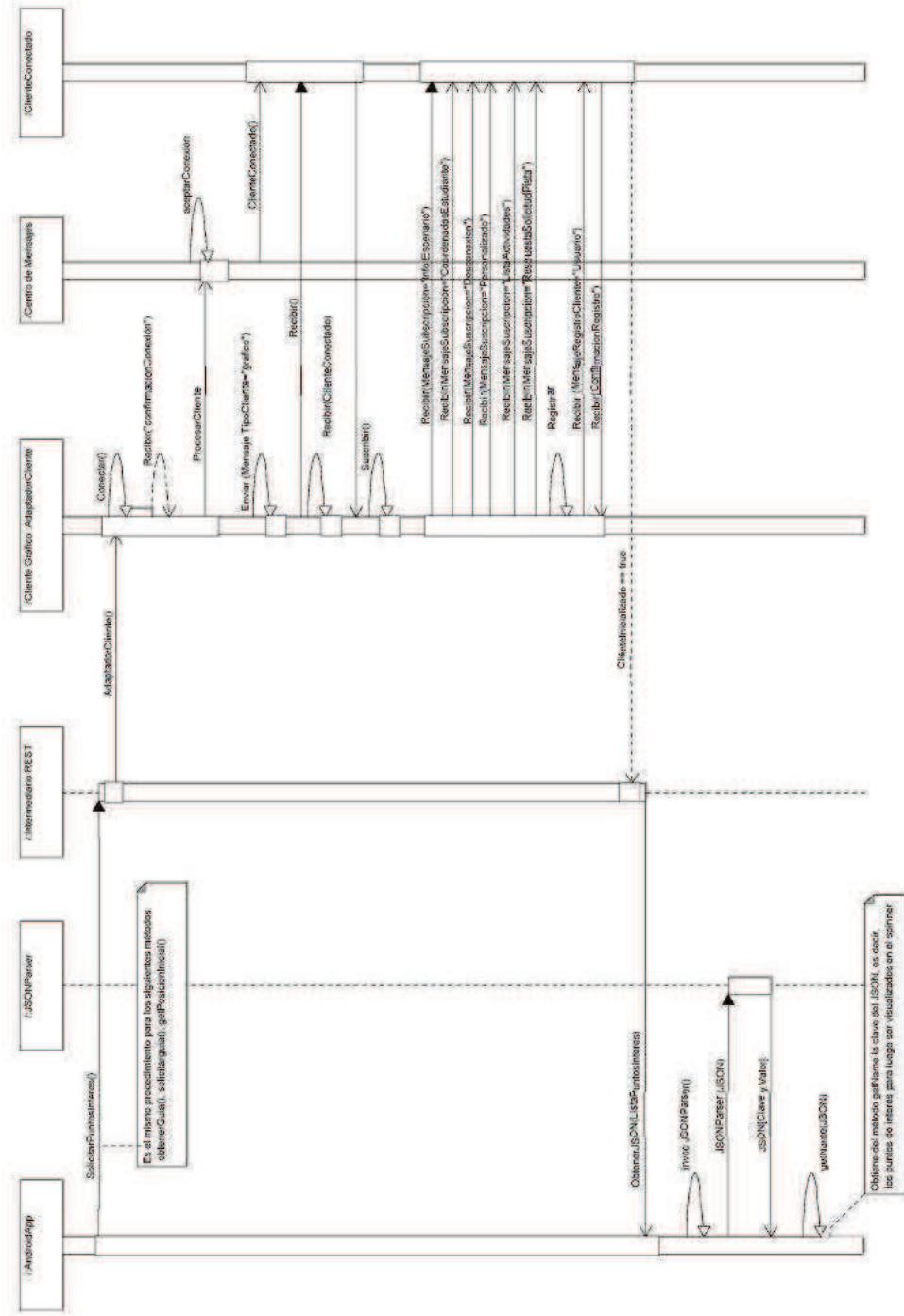


Figura 3.6: Diagrama de secuencia de la interacción de la aplicación Android con el sistema MILES

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

```
//Debido a que al realizar esta tarea larga, se creo una clase asincrona para poder obtener los datos
private class ObtenerJSON extends AsyncTask {
    @Override
    protected Object doInBackground(Object... arg0) {
        List<String> listaPuntos = null;
        ListPuntosDestino = (Spinner)findViewById(R.id.ListPuntosDestino);
        try{
            if(json==null){
                JSONParser jParser = new JSONParser();
                json=new JSONObject();
                json = jParser.getJSONFromUrl(op.url);
            }
            listaPuntos=getNames(json);
            if(listaPuntos!=null){
                ArrayAdapter<String> adp1 = new ArrayAdapter<String> (atv,
                    android.R.layout.simple_spinner_item,listaPuntos);
                adp1.setDropDownViewResource(android.R.layout.
                    simple_spinner_dropdown_item);
                ListPuntosDestino.setAdapter(adp1);
            }
        }catch(Exception e){
            e.printStackTrace();
        }
        return null;
    }
    protected void onPostExecute(Boolean result) {
    }
}
```

Cuadro 3.3: Código fuente de la clase asíncrona ObtenerJSON

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

ser interpretada por la clase *JSONParser* (ver código fuente en el apéndice codificaciones).

Ésta clase es de tipo asíncrona (permite ejecutar varias tareas al tiempo sin bloquear el hilo principal de la aplicación activity) debido a que se necesitan realizar varias operaciones de largos períodos de tiempo de interacción con MILES.

Por ejemplo el solicitar los puntos de interés al sistema MILES mediante el intermediario REST, el tiempo que se tarda MILES en devolver los puntos de interés fue exagerado en términos de tiempo de una aplicación normal, a tal punto de que la ejecución bloqueaba la ejecución del resto de los componentes de la aplicación y también de la interfaz, produciendo al usuario un efecto evidente de lentitud, bloqueo ó mal funcionamiento en general (el hilo principal detecta aquellas operaciones que superen los 5 segundos, en cuyo caso se muestra un mensaje de *Application Not Responding* y el usuario debe decidir entre forzar el cierre de la aplicación o esperar a que termine).

Por tal motivo se vio la necesidad de colocar un time por defecto de 30 segundos en la clase *AdapatadorCliente* y así de ésta forma evitar el bloqueo y permitir que el hilo principal realiza otras tareas mientras en la clase asincrona se ejecuta la petición a MILES.

- Método *getNames* (cuadro 3.4): recibe como parámetro el objeto JSON [clave y valor] donde la clave son los nombres de los puntos de interés y como valor se tienen las coordenadas donde se encuentra ubicado el punto. (ver el objeto JSON de puntos de interés en el apéndice codificaciones).

Éste método se creo con la finalidad de obtener sólo la clave del objeto, es decir los nombres de los puntos de interés para ser mostrados luego como parámetro de entrada en la pantalla de inicio de GuIAR. Como un apunte extra, *getName* fue desarrollado de forma manual, debido a que el método *JSONObject.getName* no existe en el API de Android *org.json*, Api por la cual se utiliza en ésta clase principal para la obtención del JSON.

3.2.2.3. Entorno Realidad Aumentada

A lo largo de éste capítulo se ha venido explicando como se han utilizado los distintos componentes (localización de look, intermediario REST, la plataforma MAEVIF y SensorTest) y también las diferentes dificultades técnicas que se han afrontado durante la realización de éste trabajo de fin de máster.

Ahora se va a describir como toda ésta información ha sido integrada, con el fin de crear el entorno de realidad aumentada; uno de los objetivos de éste proyecto. En el diagrama de secuencia de la figura 3.7, se representa el modelo de la creación del entorno de realidad aumentada en interiores, con mayor claridad.

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

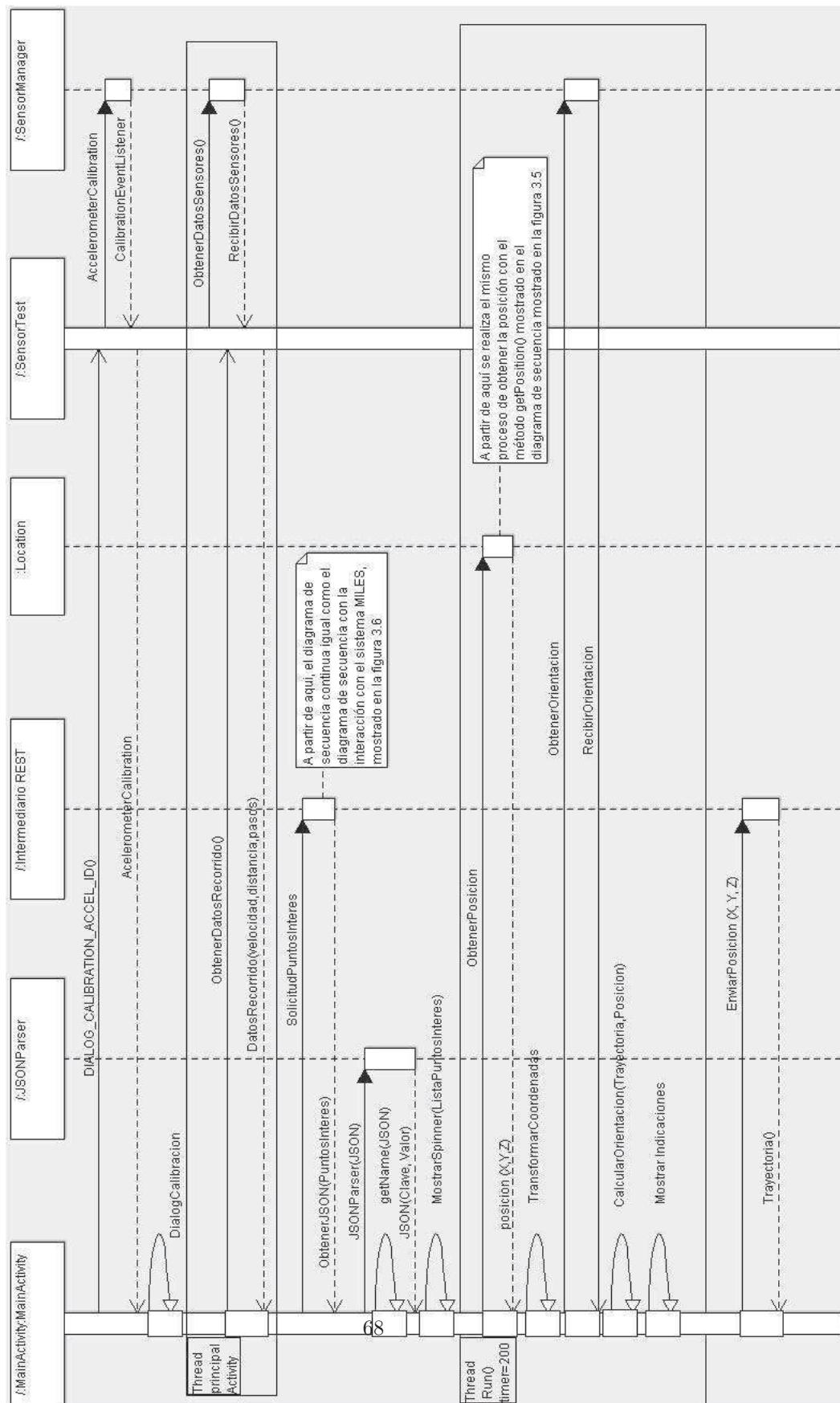


Figura 3.7: Diagrama de secuencia del entorno de Realidad Aumentada en Interiores GuiAR

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

```
/Debido a que el método JSONObject.getName no existe en el org.json API de Android creamos el siguiente
método, para retornar la clave
//http://developer.android.com/reference/org/json/JSONObject.html
private static List<String> getNames( JSONObject jo ) throws JSONException {
    int length = jo.length();
    if (length == 0) return null;
    ArrayList<String> names = new ArrayList<String>();
    JSONArray ja=jo.names();
    for(int i=0;i<ja.length();i++){
        names.add(ja.getString(i));
        Log.i("JSON", "NAMES: " + names.get(i));
    }
    return names;
}
```

Cuadro 3.4: Código fuente del método getNames

El diagrama de secuencia GuiAR, comienza en el momento en que el usuario se encuentra en la pantalla del campo de visión de la cámara que le permite ver el entorno de realidad aumentada. En primer lugar, la actividad principal muestra el cuadro de dialogo de calibración de sensores, que es a través del componente SensorTest; lo cual para poder realizar la calibración de los sensores, éste pide la información del sensor acelerómetro del dispositivo móvil para ser calibrado a través del sensorManager de Android, hasta que finalmente se calibra y muestra la pantalla principal (la calibración de sensores se detalla más adelante).

La aplicación Android cuenta con un hilo de ejecución principal permitiendo, que las actividades y lo que esté dentro de ella se ejecute; dentro de éste hilo está el mostrar durante el tiempo del entorno de RA iniciado, la información del recorrido del usuario, que se obtiene a través del componente sensorTest y del sensorManager de Android; donde se obtiene la velocidad, distancia y cantidad de pasos realizados por el usuario en su trayectoria.

Luego se presenta la interacción con el sistema MILES, que se realiza tan sólo una vez el solicitar los puntos de interés, se realiza a través del intermediario REST y este se encarga de realizar el procedimiento indicado en el diagrama de secuencia visto en la figura 3.6 (este procedimiento se realiza de la misma forma para los demás llamados a la plataforma de MILES); una vez obtenido el objeto JSON con los puntos de interés, se analiza la información obteniendo así la clave (nombres de los puntos de interés) y el valor (coordenadas del punto) del objeto; que dentro del método *getNames* permite obtener tan sólo la clave que corresponde a los nombres de los puntos de interés, para luego ser mostrados en el spinner ó lista desplegable y poder así el usuario elegir el punto destino.

Una vez obtenido el punto destino, se procede a obtener la posición del usuario, todo éste procedimiento se demarca dentro de otro hilo para que se ejecute cada dos milisegundos de tiempo. El obtener la posición, se realiza a través del componente de localización , donde el procedimiento que se realiza es el indicado

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

en la figura 3.5 antes explicada. Luego al obtener la posición (X,Y,Z) del usuario, en la clase principal se obtiene la orientación a través del sensorManager de Android, para luego realizar los cálculos necesarios para saber a que ángulo se encuentra el objeto ó punto destino en el espacio, en función del vector de orientación del usuario y finalmente mostrar las indicaciones posibles de la trayectoria de forma gráfica y/ó verbal en el campo de la visión de la pantalla, produciéndose así la realidad aumentada en interiores.

A continuación se muestra en la figura 3.8, la representación de la información que se muestra en la pantalla del smartphone, que más adelante se detalla.

▪ Información como parámetro de entrada

Se tiene como información verbal aquella que es proporcionada por la plataforma MAEVIF, por ejemplo los puntos de interés se mostrarán mediante una lista desplegable o spinner (función propia de Android que permite desplegar una lista con opción a selección). Mediante un rectángulo de color gris con esquinas redondeadas se denota los puntos destino a elegir por el usuario. Éste aparecerá en la parte superior de la pantalla.

▪ Campo de la visión de la cámara

De manera superpuesta se le muestra al usuario indicaciones ó pistas de la trayectoria de forma verbal (información obtenida por la plataforma de MILES a través del intermediario REST) y de forma gráfica como son las flechas de orientación (indican hacia adelante, hacia atrás, derecha e izquierda); de ésta forma se da una guía personalizada al usuario hasta su punto destino elegido.

▪ Datos del recorrido

Ésta información es aquella que ya se ha mencionado en apartados anteriores, relacionada con un contador de pasos, distancia y velocidad recorrida por el usuario, ésta se obtuvo incorporando y adaptando el componente de medición SensorTest explicado en el apartado 2.6.3 dentro de la clase principal de GuIAR.

▪ Botón Salir

Este botón permite que el usuario pueda salirse de la sesión iniciada de GuIAR, una vez presionado éste se devuelve a la pantalla del menú principal de la aplicación.

Calibración de sensores

Como se mencionó anteriormente en la explicación del diagama de secuencia del entorno de RA, éste es el primer proceso que realiza la aplicación. Los teléfonos Android necesitan ser recalibrados de vez en cuando para asegurar que

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

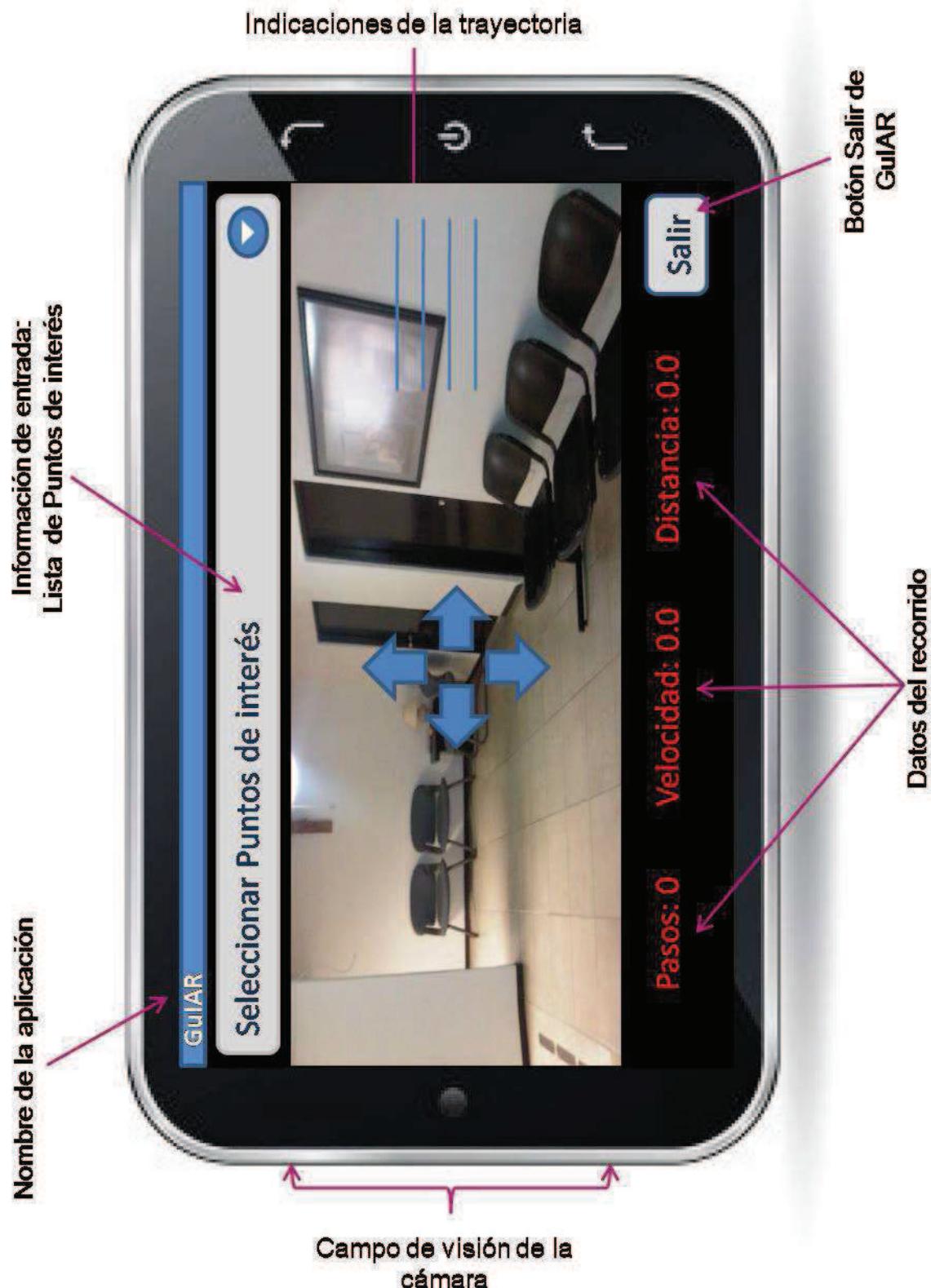


Figura 3.8: Interfaz gráfica de la sesión de inicio de GuIAR

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

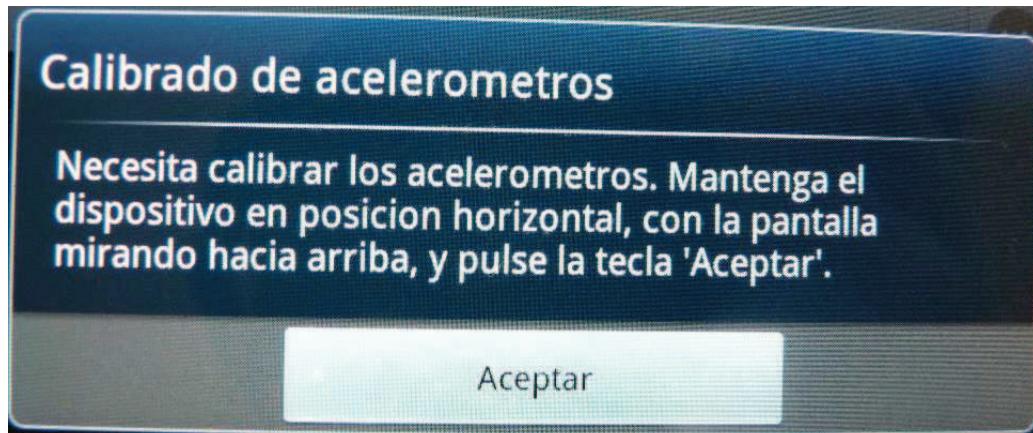


Figura 3.9: Mensaje de Calibración del smartphone

los sensores funcionan correctamente. Usa la función de calibración del sensor G para esto. El sensor G es un acelerómetro que mide las fuerzas que actúan sobre él y permite el giro automático de la pantalla y otras mejoras del dispositivo.

Se debe realizar lo que dice el mensaje (como se muestra en la figura 3.9) coloca el dispositivo móvil sobre una superficie plana ó nivelada como una mesa, y luego pulsar sobre el botón *Aceptar* y espera un segundo a que el teléfono se calibre.

Cálcular la Orientación

La orientación que nos proporciona Android nos indica la dirección a la que apunta el dispositivo y como valor resultante a través del sensorManager está dado en grados, es decir, el ángulo polar ¹¹ con respecto al norte (orientación azimuth descrito en el apartado 2.3.2.1).

Debido a que la posición del usuario se encuentra en coordenadas cartesianas (Julier y Uhlmann, 1997) al igual que cada uno de los puntos de la trayectoria que genera la plataforma MAEVIF, se hace necesario realizar una transformación de coordenadas cartesianas a coordenadas polares y así tener todos los puntos en un mismo eje de coordenadas de referencia. A continuación se hace una breve descripción de las coordenadas polares y se explica el modelo geométrico utilizado para la transformación de coordenadas.

1. Coordenadas polares

Las coordenadas polares son un sistema de dos dimensiones que sirven para encontrar puntos en un plano, en el plano se tiene un punto fijo y éste es llamado polo; además se elige un eje X por el polo y lo denominamos eje polar. El polo y el eje polar constituyen la base del sistema de coordenadas

¹¹Ángulo polar: ángulo que forman el eje polar y un radio vector, en un sistema de coordenadas polares

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

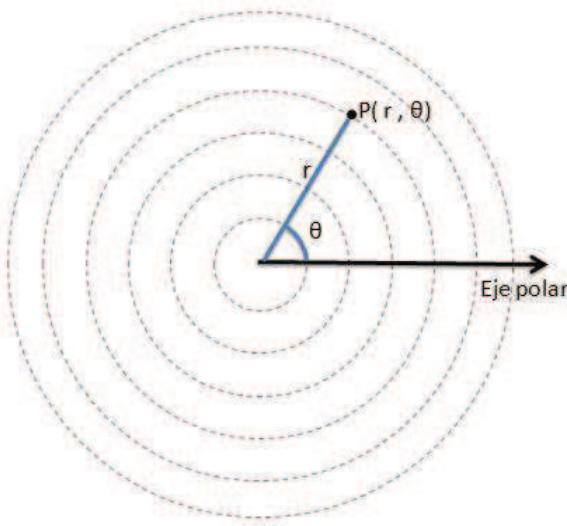


Figura 3.10: Coordenadas polares

polares tomado de (Claeys, 2013) y (Julier y Uhlmann, 1997) . Cada punto del plano polar esta definido por una distancia o radio.

Se define para cada punto un par de coordenadas (r,θ) donde r es la distancia que hay desde el origen del plano hasta el punto y θ es el ángulo medido desde el eje polar en sentido antihorario hasta la linea que atraviesa el punto desde el origen (el ángulo solo se trabaja en radianes) como se muestra en la figura 3.10¹².

Un ejemplo conciso con respecto a las coordenadas polares (como la orientación generada por la plataforma Android) y la posición del usuario ó punto de interés proporcionado por la plataforma MAEVIF, se presenta en la figura 3.11, donde los puntos se encuentran definidos por dos ejes X y Y denominados abscisa y ordenada respectivamente.

2. Transformación de coordenadas

Para transformar las coordenadas cartesianas $P(x,y)$ a coordenadas polares $P(r,\theta)$ (ver figura 3.11), se ha tenido en cuenta el teorema de pitágoras (Strathern, 1999) para poder realizar la conversión entre ellas y aplicando dicha trigonometría se obtienen las siguientes expresiones:

$$r = \sqrt{x^2 + y^2}$$

$$\theta = \tan^{-1} \left(\frac{y}{x} \right)$$

¹²[http://www.wikimatematica.org/index.php?title=Coordenadas polares](http://www.wikimatematica.org/index.php?title=Coordenadas_polares)

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

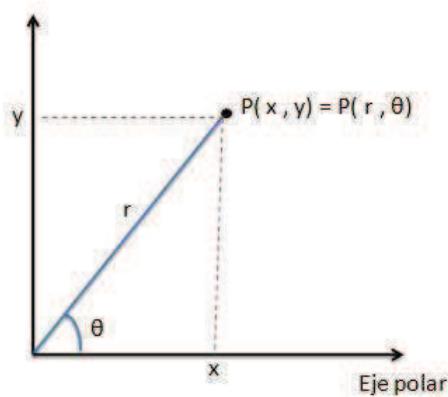


Figura 3.11: Coordenadas polares y cartesianas

Donde r es la distancia entre dos puntos cartesianos y θ es el valor del ángulo de orientación entre la posición del usuario y el eje de referencia.

IMPLEMENTACIÓN

Para el desarrollo de la información descrita anteriormente, se ha desarrollado casi todo dentro de la clase principal *MainActivity*. Se importó el componente de medición *SensorTest* como una librería, éste hace uso de tres sensores del dispositivo móvil: acelerómetro, giróscopo, y el campo magnético; declarados en ese orden en el método *onResume()*, para tener acceso a los datos del sensor se indica al *SensorManager* con el método *registerListener*, como se muestra en el cuadro 3.5.

A su vez se tiene la clase *GuiaRecorrido* donde se realiza el llamado a las distintas clases de la librería incorporada *SensorTest* para obtener los resultados de los siguientes cálculos realizados:

- La *velocidad*: siendo la velocidad del paso (en pasos por segundo) por la longitud del paso (longitud dada como una constante de 39cms).
- La *distancia*: corresponde a la longitud del paso por la cantidad de pasos dados por el usuario
- Y por último el conteo de *pasos*: es la aceleración medida por el dispositivo, y la desviación respecto de la aceleración de fondo: la gravedad. Luego analiza la forma de la señal de la gravedad, y cada vez que detecta un pico alto y un pico bajo seguidamente, se estima que se ha producido un paso (técnica descrita en el apartado 2.6.3).

Para obtener y mostrar en pantalla los tres datos anteriormente mencionados, se tiene el siguiente método *updateSensorData* dentro de la clase *MainActivity*,

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

```
mSensorManager.registerListener(mGuiaRecorrido, mSensorManager.  
getDefaultSensor(Sensor.TYPE_ACCELEROMETER),Constant.SENSOR_DELAY);  
  
mSensorManager.registerListener(mGuiaRecorrido, mSensorManager.  
getDefaultSensor(Sensor.TYPE_GYROSCOPE),Constant.SENSOR_DELAY);  
  
mSensorManager.registerListener(mGuiaRecorrido, mSensorManager.  
getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),Constant.SENSOR_DELAY);
```

Cuadro 3.5: Código fuente del acceso a los datos de los sensores del dispositivo móvil

```
public void updateSensorData(int steps,double speed,double distance){  
    DecimalFormat df = new DecimalFormat("#.##");  
    TextView txtSensores = (TextView) findViewById(R.id.textSensor);  
    txtSensores.setTextColor(Color.RED);  
    txtSensores.setText("Steps:" +steps+" Speed:"+df.format(speed)+"  
    Distance:" +distance);  
}
```

Cuadro 3.6: Código fuente del método updateSensorData

donde se pinta en el campo de visión del smartphone la información correspondiente en color rojo y formateado a dos décimas (cuadro 3.6).

En cuanto a la implementación del mensaje de calibración del móvil, se tiene el método protegido de crear el dialogo llamado *onCreateDialog* dentro de él invocamos la clase *AccelerometerCalibration.getDialog*, que es la clase perteneciente al componente de medición *SensorTest*, que permite recoger la información del acelerómetro a través del manager pasado como parámetro, y una vez calculada la desviación del mismo informa del éxito o fracaso del proceso de calibrado y de la desviación calculada (en caso de éxito) al objeto peticionario de la calibración.

A continuación se muestra en el cuadro 3.7, el contenido del método *onCreateDialog* que se encuentra dentro de la clase principal *MainActivity*

Para obtener la orientación del dispositivo móvil se hace a través del sensor del dispositivo móvil *TYPE_ORIENTATION* y también se tiene el siguiente método que accede a la clase de Android **SensorEventListener**¹³, donde se utiliza para recibir notificaciones del SensorManager cuando ha cambiado los valores del sensor con respecto a la orientación azimuth, siendo ésta orientación con respecto al norte y que la pantalla de la aplicación se mantendrá en horizontal y el resultado se almacena en la variable *orientacion* de tipo double, como se muestra en el cuadro 3.8.

¹³<http://developer.android.com/reference/android/hardware/SensorEventListener.html>

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

```
@Override  
protected Dialog onCreateDialog(final int id) {  
    final Builder builder;  
    final Dialog dialog;  
  
    switch (id) {  
        case DIALOG_CALIBRATION_ACCEL_ID:  
            dialog = AccelerometerCalibration.getDialog(this, mSensorManager,  
                mGuiaRecorrido.getTrajectoryManager());  
            break;  
        default:  
            dialog = null;  
    }  
    return dialog;  
}
```

Cuadro 3.7: Código fuente del método que permite la calibración del dispositivo móvil

```
//La orientacion se encuentra comprendida entre entre 0 y 359 grados  
private SensorEventListener mySensorEventListener = new SensorEventListener() {  
  
    public void onAccuracyChanged(Sensor sensor, int accuracy) {  
    }  
  
    @Override  
    public void onSensorChanged(SensorEvent event) {  
        orientacion= new Double(event.values[0]);  
    }  
};
```

Cuadro 3.8: Código fuente del método que obtiene la orientación generada por los sensores del móvil

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

En cuanto al modelo geométrico explicado anteriormente para la transformación de coordenadas cartesianas a coordenadas polares, se realizó en el método *obtenerOrientacionDestino* que recibe 5 parámetros descritos así:

1. *Orientacion*: es la orientación dada por el sensor del móvil (la obtenida por el escuchador de Android SensorEventListener)
2. *CX, CY*: Coordenadas X y Y actuales del usuario (las obtenidas en el posicionamiento indoor 3.2.2.1)
3. *TX, TY*: Coordenadas X y Y del punto de ruta de una trayectoria sugerida al usuario por la plataforma MAEVIF (las obtenidas en la interacción con MILES 3.2.2.2)

Para emplear la fórmula que permite convertir coordenadas cartesianas a polares, en java se tiene la clase *Math* que contiene métodos para realizar operaciones numéricas básicas y para éste caso que es una fórmula trigonométrica, se utilizó la función *toDegrees*, completando así la fórmula:

```
Math.toDegrees(Math.atan2(ty-cy, tx-cx));
```

Y para obtener la distancia entre dos puntos de interés, la fórmula en java se desarrolla de la siguiente manera:

```
Math.sqrt(Math.pow(p2.getX()-p1.getX(),2) +
Math.pow(p2.getY()-p1.getY(),2));
```

En el cuadro 3.9 se presenta el código fuente del método *obtenerOrientacionDestino*, que permite calcular la orientación del móvil con respecto al eje de coordenadas de referencia y obtener el ángulo de orientación entre el móvil, el eje de referencia y el punto de ruta.

Finalmente para la implementación del diseño de interfaz gráfica del dispositivo móvil, se ha tenido muy en cuenta el espacio de la pantalla debido a su limitación, por lo que se ha seleccionado distintos tipos de Layout ó capas simples para mostrar toda la información requerida.

Para el desarrollo de la Realidad Aumentada, es decir, superponer la información gráfica sobre la cámara del smartphone, se hace uso imágenes (flechas de orientación que indican seguir recto, girar a la izquierda y girar a la derecha) y cuando se ha llegado al destino se utiliza la imagen logo de la aplicación GuIAR como trayectoria finalizada; además presentar información verbal sobre algunas indicaciones de la trayectoria a seguir sobre la superficie de la cámara. Toda la parte gráfica de la aplicación se desarrolla mediante ficheros XML (ver código fuente en el anexo de codificaciones), como los siguientes:

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

```
/*
 * Parametros
 * orientacion: Orientacion dada por el sensor del movil
 * cx, cy: Coordenadas X y Y actuales del usuario
 * tx, ty: Coordenadas X y Y del punto de ruta.
 */
private String obtenerOrientacionDestino(double orientation,double cx,double cy,double tx,double
ty){
    //Calcula la orientacion del movil con respecto al eje de coordenadas de referencia
    double relative_orientation=orientation-axis_orientation;
    if(relative_orientation<0)
        relative_orientation=360+relative_orientation;
    Log.i("RO", ""+relative_orientation);
    //Calcula el angulo de orientacion entre el movil, el eje de referencia y el punto de ruta
    //http://www.wikimematica.org/index.php?title=Coordenadas_Polares
    double t_abs_or=Math.toDegrees(Math.atan2(ty-cy, tx-cx));
    if(t_abs_or>0)
        t_abs_or=360-t_abs_or;
    else
        t_abs_or=Math.abs(t_abs_or);

    t_abs_or=360-t_abs_or;

    //Calcula la orientacion del punto de ruta con respecto al movil
    double diff=relative_orientation-t_abs_or;

    //Si el angulo es casi cero ir recto
    if(diff>=-min_angle && diff<=min_angle)
        return "RECTO";
    else{
        //Si el angulo es negativo y no supera 180 ir a la derecha, de lo contrario a la izquierda
        if(diff<-min_angle){
            if(diff>-180){
                Log.i("TARGETORI", "DERECHA: " + diff);
                return "DERECHA";
            }
            else
                Log.i("TARGETORI", "IZQUIERDA: " + diff);
                return "IZQUIERDA";
        }
        else{
            //Si el angulo es positivo y no supera 180 ir a la izquierda, de lo contrario a la
            //derecha
            if(diff<180){
                Log.i("TARGETORI", "IZQUIERDA: " + diff);
                return "IZQUIERDA";
            }
            else
                Log.i("TARGETORI", "DERECHA: " + diff);
                return "DERECHA";
        }
    }
}
```

Cuadro 3.9: Código fuente del método obtenerOrientacionDestino

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

- *activity_mail.xml*: este fichero crea una vista en la pantalla del móvil, conformada por 3 paneles en la pantalla una vez iniciado GuiAR (descrito al inicio del apartado 3.2.2.3, ver figura 3.8), como son:
 - Información como parámetro de entrada: se creó mediante un Spinner que permite desplegar y a la vez poder seleccionar una opción, identificado como *ListPuntosDestino*.
 - Campo de visión de la cámara en tiempo real: éste es un punto importante para la Realidad Aumentada, mantener el campo de visión de la cámara para luego poder superponer objetos sobre ella, por esto se utilizó *Surface View*¹⁴, que proporciona una superficie de dibujo dedicado incrustado dentro de una jerarquía de vistas.
 - Datos del recorrido: éste panel se creó mediante un *TableLayout*, basado en filas (*TableRow*) y dentro de cada fila se coloca la información descrita en el apartado anteriormente mencionado.
- *control.xml*: este fichero se encuentra conformado por una capa llamada *LinearLayout* donde los elementos se colocan uno detrás del otro, indicando la orientación de forma vertical y dentro de ésta capa se tiene un *FrameLayout* que consiste en un marco que ocupa lo mismo que el campo de visión y de ésta forma colocar los objetos sobre la cámara del smartphone,

3.2.3. Modo de uso de la aplicación

GuAR es una aplicación de fácil uso y reconocimiento; tal como se aprecia en la figura 3.12, pues su logo de por sí describe el nombre y una vez haya sido instalada la aplicación en el dispositivo móvil Android 2.3 ó superior (los pasos de la instalación se encuentra dentro del apéndice) el icono se quedará en el panel de aplicaciones del smartphone.

Para ejecutar la aplicación se deberá dar doble clic sobre el icono de GuAR, éste inmediatamente mostrará una ventana con la animación de bienvenida a la aplicación GuAR por unos breves milisegundos y luego automáticamente pasará al menú de la aplicación, que permite la interacción con el usuario a través del cuadro de opciones.

Para saber un poco más en detalle sobre el funcionamiento de la aplicación y por quien fue desarrollado, dar clic en la opción **Acerca De**, para dar inicio a la aplicación GuAR se debe antes seleccionar la opción **Configurar**, luego aparecerá un campo de texto donde se deberá digitar la dirección URL donde se encuentra ubicado el intermediario REST.

¹⁴<http://developer.android.com/reference/android/view/SurfaceView.html>

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

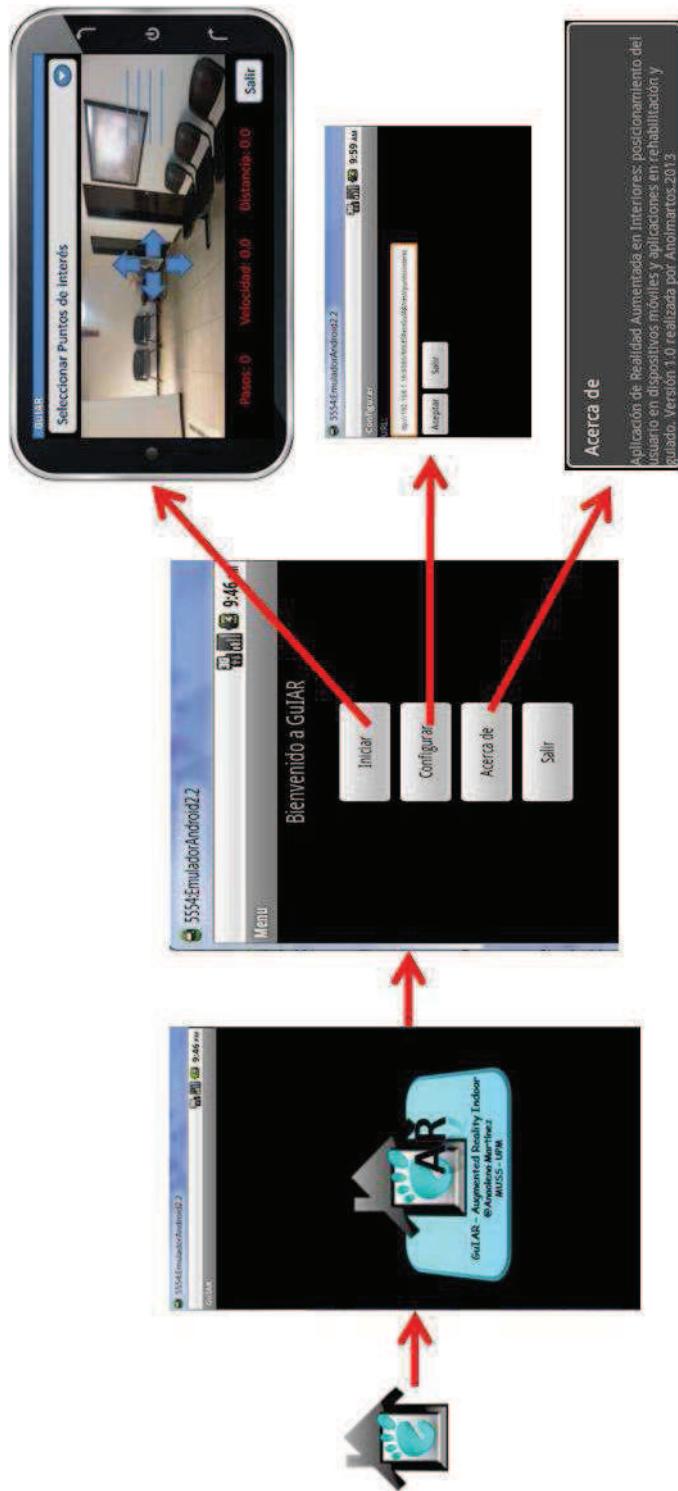


Figura 3.12: Modo de uso de la aplicación GuiAR

3.2. DISEÑO E IMPLEMENTACIÓN DE GUIAR

Una vez hecho ésto se puede dar **Iniciar** la sesión de GuIAR; donde al comienzo se mostrará el mensaje de calibrado de sensores (explicado anteriormente). En la lista desplegable se deberá seleccionar el punto de interés destino al que se quiere llegar y luego la aplicación le indicará al usuario de forma gráfica mediante las flechas de orientación e indicaciones verbales como llegar a su destino final.

Luego de terminada la trayectoria o querer finalizar la sesión, se puede dar clic en el botón **Salir**; éste retorna a la ventana del *Menú* que también tiene el botón Salir para ya cerrar la aplicación GuIAR.

Capítulo 4

Experimentación y Resultados

El presente capítulo tendrá dos tipos de caso de uso para la experimentación, el primero será de ámbito médico concerniente al enfoque que trata el proyecto REHABILITA que se explicará mas adelante, y el caso de uso de ámbito educativo, correspondiente al proyecto MILES. De ésta forma se presenta para cada caso la experimentación que permitirá evaluar la aplicación GuIAR y finalmente, se analizan los resultados obtenidos por cada caso de uso.

4.1. Sistema REHABILITA

REHABILITA es un proyecto de investigación sobre Tecnologías Disruptivas para la Rehabilitación del Futuro II, cuenta con la participación activa de empresas, centros de investigación y hospitales a nivel nacional, en los campos informáticos, médicos, electrónicos, robótica, biomédicas y entre otras; con el propósito de mejorar la calidad de vida de los pacientes desde el punto de vista científico y tecnológico, para poner en marcha el reto de crear un sistema de rehabilitación más eficiente y sostenible para la sociedad en general (S.A., 2009).

Este proyecto es un ejemplo de interacción universidad-centros clínicos-empresa, tomado del modelo ARS (Aquiles Robotics Systems), que considera ser replicado en España, tal como Rehabilita siendo este un proyecto I+D+I CENIT (Consorcios Estratégicos Nacionales de Investigación Técnica) donde se conjugan las capacidades de los integrantes, permitiendo así afianzar las tecnologías en los campos de actual aplicación (Instituto de Rehabilitación Guttmann., 2011).

Los médicos para la rehabilitación cardíaca y respiratoria, es el ejercicio físico diario del paciente; y que actualmente su implantación es debido a diversos factores como: económicos, climáticos y ausencia de personal clínico que pueda servir de apoyo a cada paciente (la población de pacientes es mayor que el personal clínico de asistencia).

4.1. SISTEMA REHABILITA

Por ello se plantea para éste campo médico, que sea un caso de uso utilizando el dispositivo móvil como una aplicación de rehabilitación ó la metáfora del dispositivo móvil como médico acompañante, donde el posicionamiento del paciente no dependa de medidas de localización externas y a su vez poder realizar el ejercicio físico dentro de un edificio si las condiciones lo requieren, y de esta forma se da una solución a la asistencia al paciente, con tan solo llevando el dispositivo móvil que le mostrará en pantalla la velocidad, cantidad de pasos y distancia recorrida que ha realizado el paciente, a través de los sensores del Smartphone como el giróscopo y acelerómetro.

4.1.1. Escenario para REHABILITA

Uno de los objetivos que se tiene al realizar este escenario de ámbito médico, es poder monitorizar y concientizar a un paciente sobre la importancia de la realización de un mínimo de terapias que incluyan sesiones de marcha, sin que las condiciones climáticas lo impidan ó el estar dentro de un edificio lo cohíban. Estos tipos de terapia son en la mayoría por recomendación médica, para pacientes que presentan traumatismos derivados del corazón ó que se encuentran en alguna fase de calentamiento ó entrenamiento.

Es por tal motivo que GuIAR presta diversas funcionalidades y las apropiadas para este contexto médico esta dado por un contador de pasos, la distancia recorrida y la velocidad final. Siendo esta la maner de permitirle a muchas personas poder realizar sus actividades de terapia dentro de recintos cerrados y tomando el dispositivo como un tipo de “médico acompañante”, que le brinde información ó el desempeño sobre la actividad realizada en el día.

En los siguientes apartados se explicará el propósito de la marcha realizada, la metodología empleada y el tracking realizado, finalmente se realiza las estadísticas y análisis de los resultados.

4.1.1.1. Propósito

A continuación se presenta las diferentes sesiones de marcha con un nivel de detalle en lo que se quiere profundizar, como es la cantidad de pasos que da la persona, la distancia recorrida y la última velocidad obtenida. Esta experimentación se realizó con la primera versión de GuIAR, donde sólo se mostraba en pantalla la información necesaria para las pruebas.

4.1.1.2. Metodología

Las pruebas se han realizado dentro de un edificio (en este caso mi apartamento), se ha tomado medidas y puesto marcas en el suelo de la medición realizada para emplear las diferentes pruebas de trayectoria planificadas, en la fotografía 4.1 se muestra el trazado del recorrido realizado.

4.1. SISTEMA REHABILITA

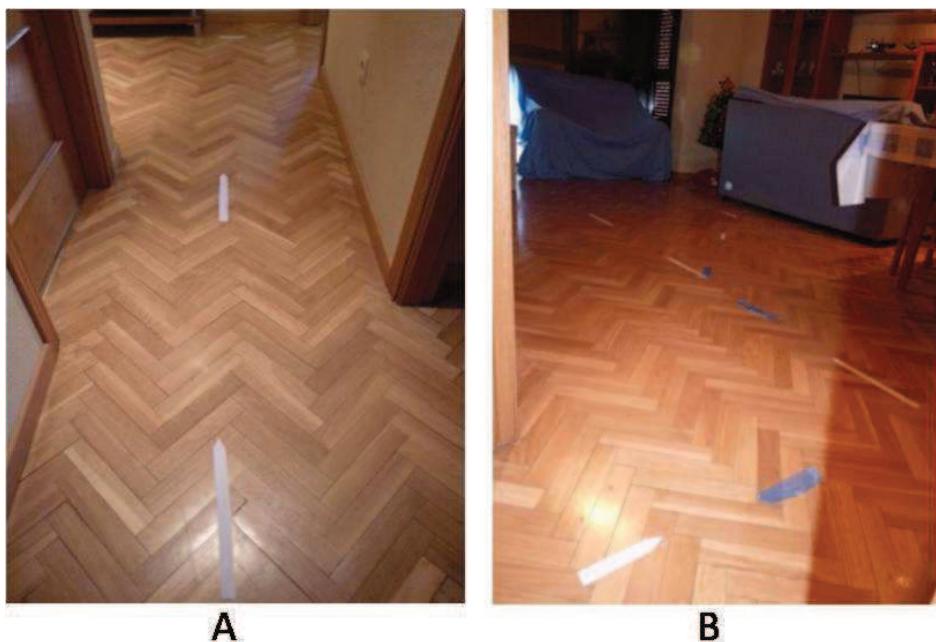


Figura 4.1: Fotografía A y B Marcación de la ruta

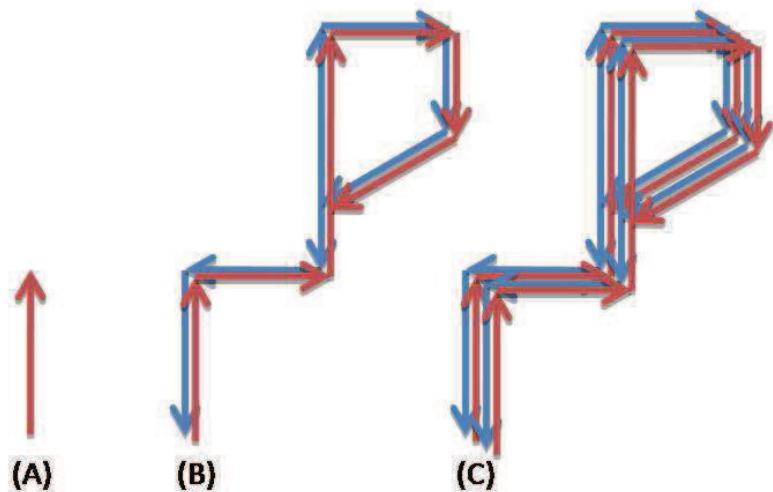


Figura 4.2: (A) Ruta de 277 cms (2.77mts), (B) Ruta de 1300 cms (13mts) y (C) Ruta de 52000 cms (52mts)

4.1. SISTEMA REHABILITACIÓN



Figura 4.3: Captura de la pantalla del móvil cuando queremos iniciar la marcha y aún se encuentran en cero las variables

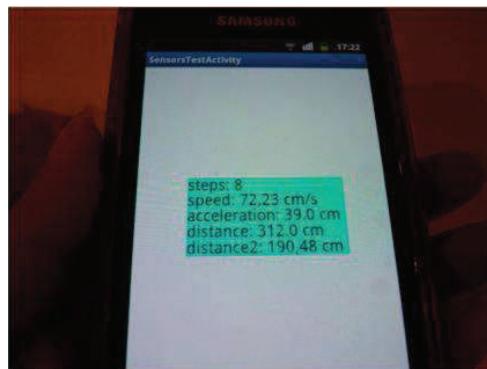


Figura 4.4: Captura de la pantalla del móvil cuando finalizó la primera ruta (A)

Para mejor visualización de lo que será el recorrido dentro del edificio, se tiene la representación de las rutas de marcha en la figura 4.2, en ella se puede apreciar las tres rutas de prueba que se han realizado para este caso práctico, donde la primera ruta tiene una distancia de 2.77 metros; la segunda ruta representa una distancia de 13 metros donde su punto de inicio es el mismo punto final; y por último se tiene una ruta más larga, donde se aprovechó la misma ruta de los 13 metros pero recorriéndola 4 veces hasta obtener los 52mts.

4.1.1.3. Tracking

Para realizar la actividad de marcha o tracking la persona portaba un Smartphone (este hace uso de los sensores acelerómetro, campo magnético y brújula), antes de empezar a realizar la marcha, se debe calibrar del móvil (al iniciar la aplicación en el móvil, este mostrará un mensaje de calibración), de forma sencilla y es colocando el móvil en una superficie plana, después se pulsa en el botón aceptar para que empiece la calibración durante unos segundos de forma automática (ver la fotografía 3.9).

4.1. SISTEMA REHABILITA

Prueba I

***Distancia real =277 cms { 2.77metros } *Pasos reales= 7**

No. Recorridos	Cantidad de Pasos	Distancia 1 (cm)	Distancia 2 (cm)	Velocidad final (cm/seg)
1	8	312,00	165,54	114,72
2	8	312,00	185,00	150,00
3	7	273,00	178,28	99,79
4	7	273,00	203,04	181,94
5	7	273,00	172,25	185,83
6	8	312,00	190,48	72,23
7	7	273,00	181,06	134,48
8	9	351,00	257,35	182,15
9	8	312,00	182,88	149,77
10	9	351,00	192,56	102,65
Total promedio =	8	304	191	137
Porcentaje Error =	14,28%	9,74%	31,04%	

Cuadro 4.1: Estadísticas Ruta (A)

Después de calibrado el móvil, se da inicio a la marcha en la fotografía 4.3, se puede apreciar que todos los contadores se encuentran en cero, excepto la longitud del paso que se muestra con el valor de 39.0cm, este es un valor constante definido dentro del código fuente del sistema de (Perez, por defender), para las pruebas se tuvieron en cuenta dos tipos de distancia, fuera de la real, la primera distancia es el cálculo de la cantidad de pasos realizado por la longitud del paso y la segunda distancia se tuvo en cuenta a partir de la velocidad del paso (en pasos por segundo) por la longitud del paso (esta distancia 2 se propuso con el fin de saber cuál sería la que más se acercaba a la realidad, después de varias pruebas esta distancia 2 no ha sido una buena opción, se acerca más a la realidad la distancia 1), Ver fotografía 4.4.

Para la prueba 1 y 2 correspondientes a las rutas (A) y (B) (tablas 4.1 y 4.2), se realizaron 10 recorridos para cada una de ellas y para la ruta (C) se realizó 5 recorridos (tabla 4.3) y para el tracking largo se realizó 3 recorridos lo cual se ve reflejada en la tabla 4.5 a la final de cada una de las estadísticas se obtiene el promedio por cada variable y el porcentaje de error con respecto a la información real.

Por último, se realizó una experimentación de trayectoria larga (como se muestra en la imagen capturada de Google ??), se tiene una distancia de referencia aproximada según Google Maps es de 250 metros aproximados andando (teniendo en cuenta que las rutas a pie están en versión beta por google).

También se utilizó otro podómetro como referencia, descargado del Play Store de Google y utilizado en el SmartPhone Sony Xperia y La aplicación GuIAR en

4.1. SISTEMA REHABILITA

Prueba II

*Distancia real =1300 cms (13 metros) *Pasos reales=32

No. Recorridos	Cantidad de Pasos	Distancia 1 (cm)	Distancia 2 (cm)	Velocidad final (cm/seg)
1	32	1.248,00	788,80	114,74
2	31	1.209,00	861,00	162,46
3	33	1.287,00	886,87	169,06
4	36	1.404,00	859,14	142,04
5	39	1.521,00	1.055,82	149,88
6	32	1.248,00	801,78	162,63
7	35	1.365,00	917,19	70,91
8	41	1.599,00	1.069,46	177,12
9	36	1.404,00	869,17	185,08
10	36	1.404,00	1.024,04	144,05
Total promedio =	35	1.369	913	148
Porcentaje Error =	9,37%	4,61%	29,76%	

Cuadro 4.2: Estadisticas Ruta (B)

Prueba III

*Distancia real =5200 cms (52 metros) *Pasos reales=128

No. Recorridos	Cantidad de Pasos	Distancia 1 (cm)	Distancia 2 (cm)	Velocidad final (cm/seg)
1	128	4.992,00	3.330,28	185,86
2	130	5.070,00	3.511,78	54,93
3	131	5.109,00	3.605,31	75,01
4	110	4.290,00	2.915,92	99,91
5	126	4.914,00	3.406,62	129,98
Total promedio =	125	4.875	3.354	109
Porcentaje Error =	2,34%	6,25%	35,5%	

Cuadro 4.3: Estadisticas Ruta (C)

4.1. SISTEMA REHABILITA

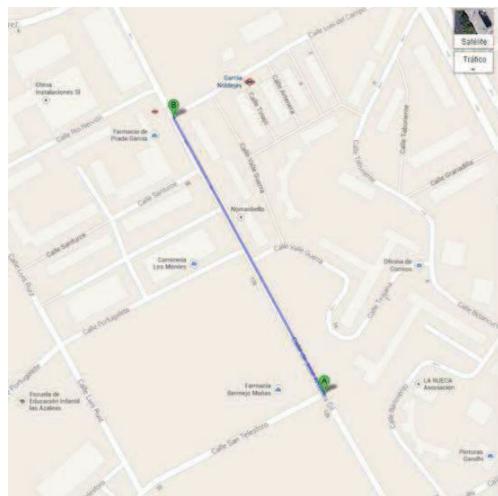


Figura 4.5: Recorrido de 250 metros aproximados. Tomada de Google Maps

Prueba IV

*Distancia real =30.000cms (300metros) *Pasos reales=432apr.

No. Recorridos	Cantidad de Pasos GuiAR	Distancia GuiAR (cm)	Cantidad de Pasos con podómetro	Distancia con podómetro (mts)
1	317	12.363	156	10.700
2	417	16.263	446	11.100
3	376	14.664	320	10.950
Total promedio =	370	14.430	307	10.917
Porcentaje Error =	14,35%	51,90%	28,93%	63,61%

Cuadro 4.4: Estadísticas tracking largo

el SmartPhone Samsung. En la tabla 4.4 se aprecia las estadísticas registradas de 3 recorridos realizados desde el punto A al punto B como se muestra en la figura ?? y se un margen de error notorio tanto del podómetro descargado como el conteo de pasos de GuiAR.

4.1.1.4. Resultados y Conclusiones

Una vez finalizadas cada una de las pruebas, se pudo apreciar que existe un margen de error notorio, y esto sucede justo después de ser calibrado el dispositivo (dejando el móvil sobre una mesa o sitio horizontal) y al recogerlo para empezar

4.1. SISTEMA REHABILITA

el tracking, ese pequeño desplazamiento ó movimiento, es contado como uno o varios pasos dependiendo de la fuerza y aceleración con la que se tome en la mano, por lo que el dispositivo no se llega a dar cuenta cuando inicia el primer paso de verdad (y al final del trayecto termine con mas pasos contados, que los que realmente se realizó).

Análisis de las tablas:

La tabla 4.1 presenta en el conteo de pasos, un porcentaje de error de 14,28 % algo aceptable, mientras que la primera distancia tiene un margen de error de 9,74 % y la segunda distancia presenta un 31,04 % por lo que es despreciable para esta prueba.

La tabla 4.2 tiene como porcentaje de erro el contador de pasos del 9,37 % y la distancial 4,61 % mientras que la distancia2 del 29,76 % (diferencia entre ellas bastante notoria), en esta pruebas se sigue viendo que es mejor la distancial1 con respecto a la 2.

La tabla 4.3 presenta un margen de error en el conteo de pasos del 2,34 % (el porcentaje de error más pequeño obtenido para estas pruebas), en cuanto a la distancial1 tiene un porcentaje de error de 6,25 y la distancia2 de 35,5 %

La tabla 4.4 es la prueba de mayor trayecto con 250metros la distancia (evaluandosé sólo la distancial mencionada anteriormente) y pasos 432 reales, en cuanto a los datos experimentales se muestra que el porcentaje de error de conteo de pasos es de 14,35 % y una distancia de 51,90 % siendo estos datos del prototipo de GuiAR inferiores a una aplicación de podómetro descargada en el otro Smartphone de prueba, donde el resultado de la distancia del podometro tiene un error de 63,61 % y el conteo de pasos del 28,93 %.

Como conclusiones se tienen varios puntos a tener en cuenta, la primera es que en distancias cortas se ve un comportamiento relativamente estable y se acerca a la real, pero a medida que la trayectoria es aún mas larga como la última prueba realizada, el sistema se desfaza notoriamente. Sin embargo, un punto a favor para este trabajo, es la comparación que se hace con la aplicación de podómetro descargado, siendo éste mas deficiente que el sistema de conteo de GuiAR. Por lo que también se puede concluir, que existe una falencia en cuanto a los sistemas de cálculos para obtener con precisión éste tipo de funcionalidades; aunque se encuentren disponibles para los usuarios en la tienda de Google.

otro punto a tener en cuenta y que afecta en mayor medida el conteo de pasos, es el valor constante de la longitud del paso, debido a que este debería variar según las características propias de la persona.

Finalmente también se tiene el problema de los sistemas de navegación inercial, se debe a que ir a una velocidad constante es bastante difícil en el caso de irse andando, por eso siempre deberían completarse con otros métodos que ofrezcan una posición absoluta, de forma que la navegación inercial llene los huecos entre mediciones absolutas.

4.2. EXPERIMENTACIÓN DE ÁMBITO EDUCATIVO MILES

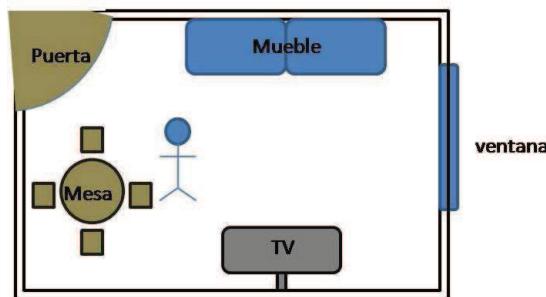


Figura 4.6: Plano del interior de la sala principal

4.2. Experimentación de ámbito Educativo MILES

El sistema MILES es un proyecto que tiene como propósito construir un sistema de navegación semántico en entornos virtuales y reales, orientado a recintos cerrados en el ámbito educativo, por lo que el prototipo de Realidad Aumentada en Interiores GuIAR permite ésta funcionalidad de guía personalizada y adaptativa en un escenario concreto. Inicialmente se planteó la posibilidad de realizar la experimentación en la Universidad Politécnica de Madrid, sin embargo al ser un prototipo inicial se decidió emprender las pruebas en un recinto cerrado de menor escala (mi apartamento) y de esta forma analizar los resultados obtenidos, para finalmente realizar pruebas en un recinto cerrado de mayor escala como la Universidad.

De tal forma para la realización de estas experimentaciones dentro del edificio de prueba (piso de 120mts cuadrados), se tomaron medidas para marcar puntos de coordenadas cartesianas, se determinó objetos relevantes y existentes en el escenario. Por lo que se creó un plano del interior del piso, con sus respectivas medidas y se simuló la trayectoria de puntos de coordenadas (información que generaría MILES), para esto se implementando un método que utilizará el algoritmo de Dijkstra (W., 2002), donde el resultado sería la ruta más corta entre dos puntos dados; el punto de inicio se detecta automáticamente por el dispositivo móvil (calculando la distancia corta entre todos los puntos de interés del plano) y el punto destino es el que selecciona el usuario en la aplicación.

A continuación se encontrarán las diferentes pruebas realizadas de la versión final, del prototipo de Realidad Aumentada en Interiores GuIAR. Para ello se han creado diversos escenarios para evaluar la funcionalidad de la aplicación, mediante resultados estadísticos en factores cómo: la orientación, el posicionamiento en interiores y el tracking con GuIAR.

4.2.1. Escenario I

Se realizó en un espacio cerrado, como fue la sala principal; en él se definieron puntos importantes para el usuario definidos como objetos: mesa, televisión,

4.2. EXPERIMENTACIÓN DE ÁMBITO EDUCATIVO MILES

Objetos de Interés	Posiciones fijas del usuario		
	Coordenada [150,150]	Coordenada [150,300]	Coordenada [300,150]
Televisión	32,4º	5º	5,2º
Ventana	24,7º	37,7º	43,60º
Mueble	41,2º	0º	83,8º
Puerta	8º	33,9º	53,8º
Mesa	0º	25,5º	40,8º
Error Promedio=	21,26º	20,42º	45,44º
			Total Error Prom.: 29,04º

Cuadro 4.5: Error de orientación obtenido por cada objeto de interés

ventana, mueble y puerta, la representación gráfica se muestra en la imagen 4.6

4.2.1.1. Propósito

Se pretende evaluar la orientación angular del usuario, ubicandose en una posición fija y a través de GuIAR le indicará al usuario a que dirección se encuentra el objeto de interés seleccionado por él mismo.

4.2.1.2. Metodología

Se definieron un conjunto de objetos de interés en un plano de coordenadas determinado y se ubicó el usuario en un punto estático y se le indicó por el móvil mediante la aplicación GuIAR, los objetos de interés y finalmente se determina el margen de error en grados.

4.2.1.3. Orientación

Para esta prueba se tomarón 3 puntos de referencia en el plano de la figura 4.6 (coordenadas: [150,150], [150,300] y [300,150] coordenadas en centímetros) de los cuales se encuentran dentro del espacio libre de la sala (20,42 metros cuadrados) y en cada punto definido se van apuntando las coordenadas polares con respecto a la indicación de ubicación del objeto de interés.

4.2.1.4. Resultados y Conclusiones

Cómo análisis de la tabla 4.5, se muestra el margen de error obtenido en grados por cada uno de los objetos de interés de la sala, estando el usuario en una posición fija definida, por ejemplo para el punto fijo coordenada [150,150] el usuario seleccionó ir al objeto Televisión, y al calcular la distancia ángular

4.2. EXPERIMENTACIÓN DE ÁMBITO EDUCATIVO MILES

entre el ángulo dado por el dispositivo móvil y la real, se obtuvo 32,4 grados de margen de error; y así de esta manera para cada uno de los objetos. Como puntos relevantes se aprecian los siguientes casos positivos, la indicación de la mesa estando en la posición [150,150] y del mueble para la posición [150,300] sin ningún margen de error osea 0° , aunque en el tercer caso para la posición [300,150] la indicación hacia el objeto televisión tuvo en menor medida un margen de error de tan sólo 5.2° . En general el promedio total del margen de error en grados es de 29 grados.

También podemos decir que la orientación se obtiene satisfactoriamente cuando su objetivo se encuentra cerca, a medida que el objeto se encuentra lejano varía el ángulo de orientación.

Además es importante mencionar que la zona cercana al objeto televisor, hace que distorsione en gran medida la orientación; ésto debido a campos electromagnéticos que se tienen cerca del televisor aunque este estuviese apagado (tomadores de corrientes, antenas, portátiles, etc.)

Finalmente para este escenario se puede concluir, que se han obtenido resultados positivos, ya que el margen de error angular en total es de tan sólo 29° y como se mencionó anteriormente, por cada prueba realizada en un punto fijo se obtuvo una precisión de la orientación con respecto al objetivo fijado. Este resultado final promete una aceptación del modelo geométrico empleado para el cálculo de la orientación, aunque habría que optimizar más el modelo para una mayor precisión.

4.2.2. Escenario II

El elemento más importante a la hora de guiar a un usuario dentro de un espacio interior es el de detectar correctamente su posición. Como se ha mencionado anteriormente, al no funcionar los sistemas GPS al interior de un edificio se hace necesario utilizar otras estrategias, concretamente, se han propuesto dos tipos de sistemas: aquellos basados en cálculo del movimiento inercial y aquellos basados en análisis de señales WiFi. Este trabajo utilizó la estrategia de cálculo inercial implementado en la herramienta LookAR y en esta sección se procederá a evaluarlo.

4.2.2.1. Propósito

Evaluando la capacidad del sistema GuIAR de calcular la posición de un usuario que recorre un espacio interior y medir el error obtenido en dicha estimación.

4.2. EXPERIMENTACIÓN DE ÁMBITO EDUCATIVO MILES

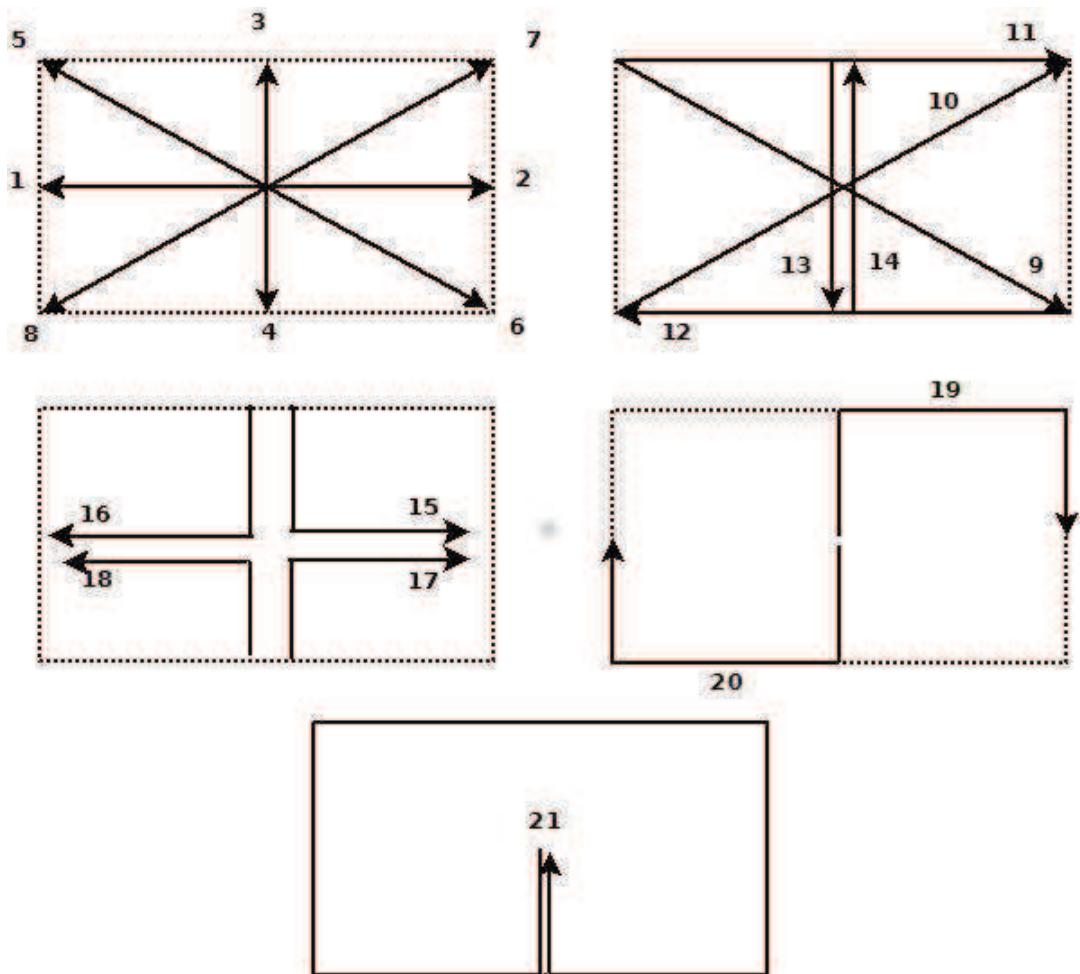


Figura 4.7: Representación de cada uno de las trayectorias realizadas

4.2. EXPERIMENTACIÓN DE ÁMBITO EDUCATIVO MILES

4.2.2.2. Metodología

Se definió un rectángulo de 3 mts x 2 mts ubicado en un espacio interior (en un apartamento) este espacio fue tratado como un espacio de coordenadas cartesianas con el punto de origen en el centro del rectángulo. Sobre este plano cartesiano se realizaron recorridos de diferentes longitudes (ver figura 4.7) y al final de cada uno de ellos se calculó el error entre lo estimado y lo real usando distancia euclíadiana y Manhattan, estos resultados son analizados con el fin de extraer conclusiones.

4.2.2.3. Tracking

En la figura 4.7 se presenta cada una de los 21 recorridos realizados en este escenario (recorridos simples, semicomplejos y complejos), para evaluar el posicionamiento indoor.

En los dos primeros cuadrantes se tienen los recorridos simples (recorrido 1 hasta el recorrido 15), los recorridos semicomplejos se presentan en los dos cuadrantes siguientes del 15 al 20 y uno complejo que corresponde al recorrido 21.

4.2.2.4. Resultados y Conclusiones

Los resultados para los 21 recorridos son mostrados en la tabla 4.6, en ella se presenta la posición (x,y) real donde termina el recorrido y también la estimada por el sistema, además de la distancia recorrida en cada caso. Se calculó la distancia euclíadiana y Manhattan (Black, 2006) entre los dos puntos y con ellas se realizaron las gráficas presentadas en las figuras 4.8 y 4.9. La distancia Manhattan es de especial interés ya que suma el desfase obtenido en los dos ejes, el cual afecta el cálculo de la orientación de los objetos de interés con respecto al usuario y por ende la capacidad del sistema para realizar una guía efectiva.

En los datos presentados en la figura 4.9 (usando la distancia Manhattan) se observa que el error tiene a aumentar en la medida que el recorrido sea más largo (aunque también influye los giros realizados durante el mismo), realizando una aproximación lineal de los datos se obtiene que por cada metro adicional en el recorrido el error aumenta en un promedio de 48.21 centímetros, el cual interfiere con el cálculo de la orientación al cabo de unos metros recorridos. Es por esto que el cálculo de la posición del usuario es la mayor fuente de error en el sistema GuIAR.

4.2.3. Escenario III

En este último escenario se prueba el prototipo completo de la aplicación GuIAR, este es uno de los escenarios más importantes donde se evaluará el comportamiento del sistema en un trayecto semicomplejo dentro del piso, para así mismo saber si es factible proceder a una experimentación en un espacio físico más grande.

4.2. EXPERIMENTACIÓN DE ÁMBITO EDUCATIVO MILES

Cant.	Posición Final Real (mts)		Posición Final Estimado (mts)		Distancia Recorrida	Error (Euclídea)	Error (Manhattan)
	X	Y	X	Y			
1	1,5	0	3,39	1,43	1,5	2,37	3,32
2	-1,5	0	-3,74	1,47	1,5	2,68	3,71
3	0	1	1,68	1,47	1	1,74	2,15
4	0	-1	-0,72	-1,81	1	1,08	1,53
5	1,5	1	3,79	-0,22	1,8	2,59	3,51
6	-1,5	-1	-4,11	-0,08	1,8	2,77	3,53
7	-1,5	1	0,71	2,34	1,8	2,58	3,55
8	-3	-2	-2,55	-0,97	3,6	1,12	1,48
9	-3	2	-1,27	4,29	3,6	2,87	4,02
10	-3	0	-3,28	0,77	3	0,82	1,05
11	3	0	6,05	-2,26	3	3,8	5,31
12	0	-2	0,25	-2,88	2	0,91	1,13
13	0	2	2,6	1,76	2	2,61	2,84
14	1,5	-1	3,86	-3,11	1,8	3,17	4,47
15	-1,5	-1	-3,39	-0,49	2,5	1,96	2,4
16	1,5	-1	4,58	-3,72	2,5	4,11	5,8
17	-1,5	1	-0,24	2,11	2,5	1,68	2,37
18	1,5	1	5,85	0,45	2,5	4,38	4,9
19	-1,5	0	-1,53	0,68	3,5	0,68	0,71
20	1,5	0	3,77	-0,96	3,5	2,46	3,23
21	0	0	5,15	-3,36	12	6,15	8,51

Cuadro 4.6: Datos de posición real, la estimada y la distancia recorrida por cada trayectoria

4.2. EXPERIMENTACIÓN DE ÁMBITO EDUCATIVO MILES

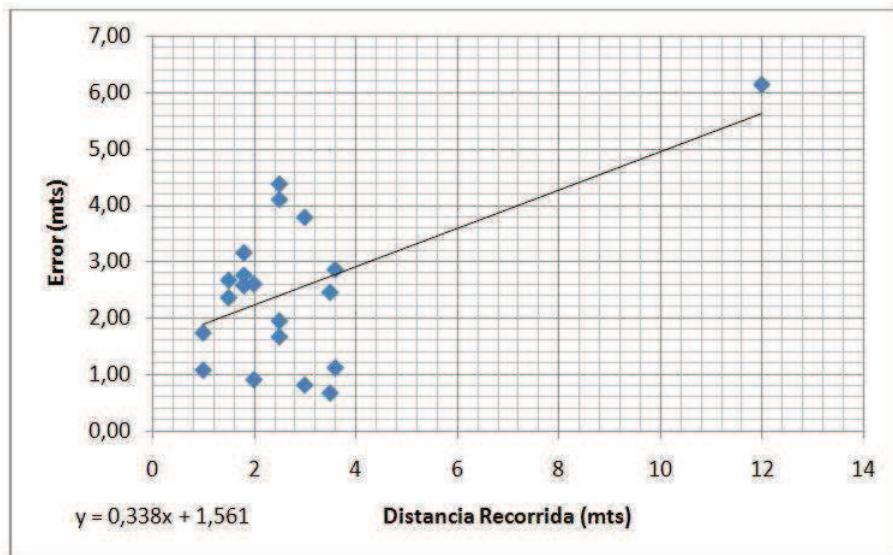


Figura 4.8: Correlación lineal que relaciona el crecimiento del error (empleando el cálculo de la distancia Euclídea) a medida que aumenta la distancia del recorrido

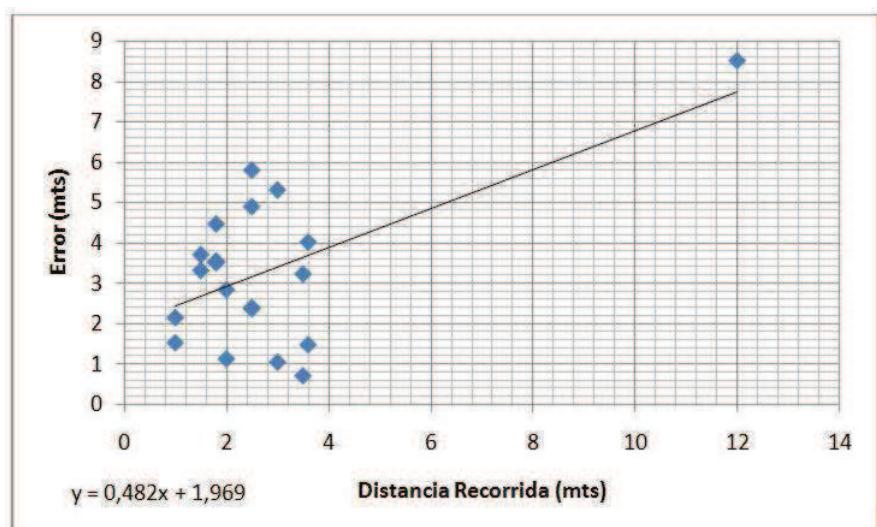


Figura 4.9: Correlación lineal que relaciona el crecimiento del error (empleando el cálculo de la distancia Manhattan) a medida que aumenta la distancia del recorrido

4.2. EXPERIMENTACIÓN DE ÁMBITO EDUCATIVO MILES

4.2.3.1. Propósito

Evaluar la funcionalidad completa del prototipo de GuIAR, y saber si es posible llegar a la trayectoria final, además de calcular un margen de error del punto destino al generado por el dispositivo móvil.

4.2.3.2. Metodología

Se realizó el plano interior del apartamento y se marcaron puntos de interés estratégicos para distintos tipos de pruebas. Una de ellas como punto de inicio es la sala y como punto destino el cuarto de aseo (otros puntos de interés la cocina, habitación, salida del apartamento), luego sobre esta trayectoria se dibujaron coordenadas cartesianas, para luego emprender la guía y al final se calcula el margen de error obtenido entre lo estimado y lo real.

4.2.3.3. Tracking con GuIAR

Se realizaron 5 pruebas con diferentes destinos y puntos de partida. La primera prueba se realizó cuando el usuario se encuentra en la sala y desea ser orientado hasta el cuarto de baño. Entonces, se da inicio a la aplicación GuIAR y el usuario se empieza mover según las indicaciones dadas por el móvil mediante la Realidad Aumentada.

En las fotografías 4.10, 4.13, 4.11 y 4.12 se muestra como el móvil indica al usuario hacia donde debe dirigirse, de este modo también se ven los diferentes tipos de flechas de orientación. La segunda prueba fue desde la habitación hasta la cocina, la tercera comprendía desde la cocina hasta la sala, la cuarta desde la sala hasta la habitación y finalmente de la sala hasta la salida del apartamento.

4.2.3.4. Resultados y Conclusiones

En cuanto a resultados de esta trayectoria, el prototipo GuIAR inicio con un comportamiento positivo mientras estaba aún en el primer punto de origen (la sala), pero luego no fue posible traspasar la puerta de salida de la sala para dirigirse al punto destino (cuarto de baño); por lo tanto no hubo una manera medible de obtener el error, debido a que no se logró llegar al punto final. Y así con cada una de las 5 pruebas realizadas.

Se debe tener en cuenta que este prototipo de realidad aumentada en interiores, tiene integrado componentes base que se encuentran en fase beta y otros en desarrollo, por lo tanto no se podía esperar resultados ambiciosos; sin embargo toda la lógica construida es coherente y en los escenarios anteriores se ha visto resultados positivos con márgenes de error medibles. Finalmente, hasta no tener una versión mejorada de los componentes base y una optimización a la aplicación GuIAR, no se podría realizar una experimentación en grandes dimensiones.

4.2. EXPERIMENTACIÓN DE ÁMBITO EDUCATIVO MILES

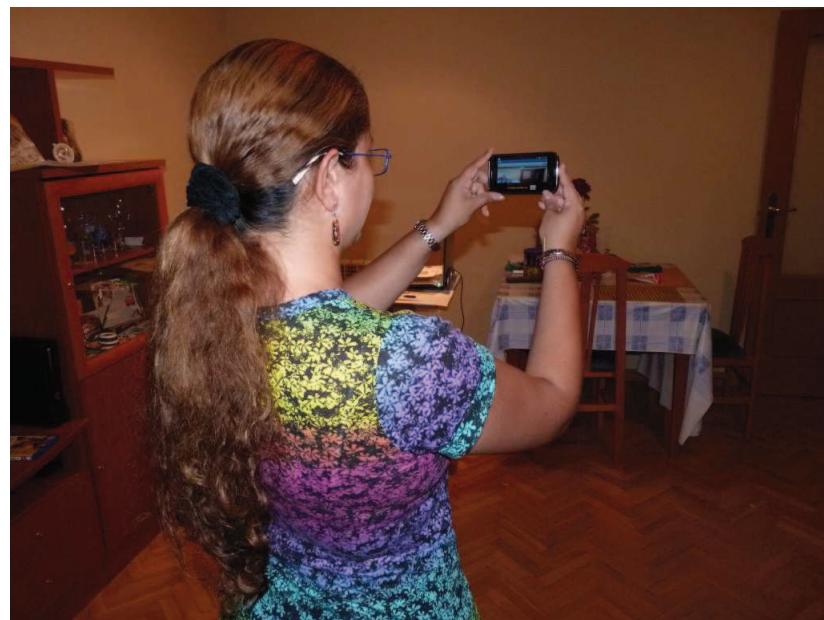


Figura 4.10: Seguimiento de las indicaciones de GuIAR, vista completa.

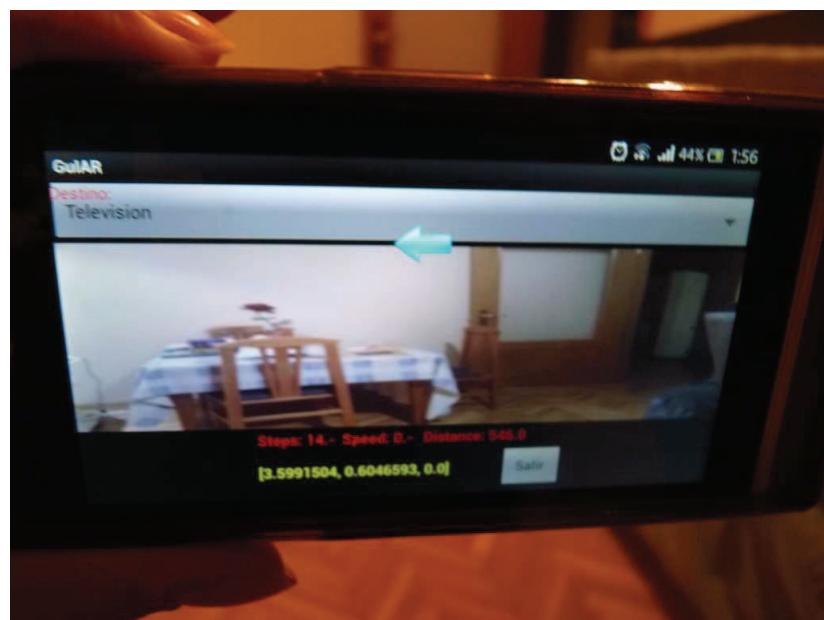


Figura 4.11: Indicación de GuIAR orientando a la izquierda

4.2. EXPERIMENTACIÓN DE ÁMBITO EDUCATIVO MILES

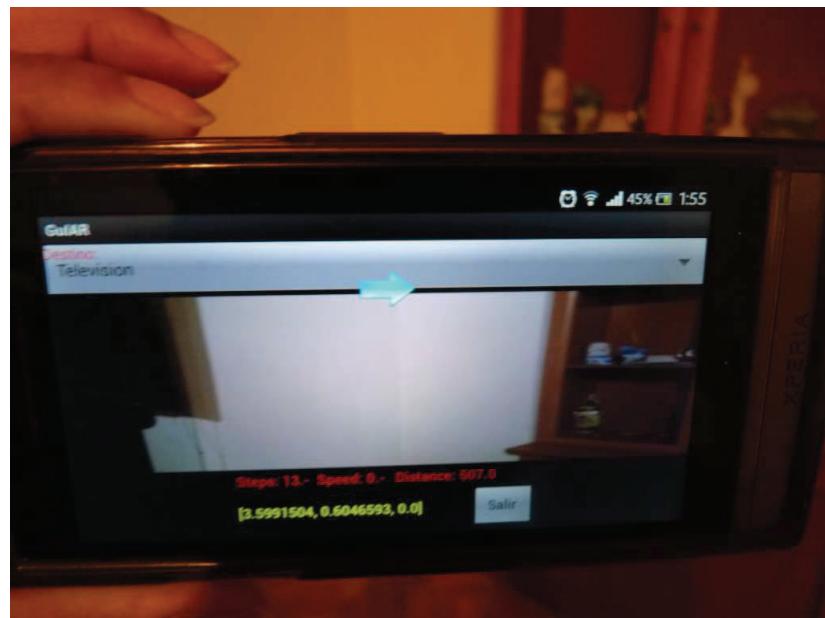


Figura 4.12: Indicación de GuIAR orientando a la derecha

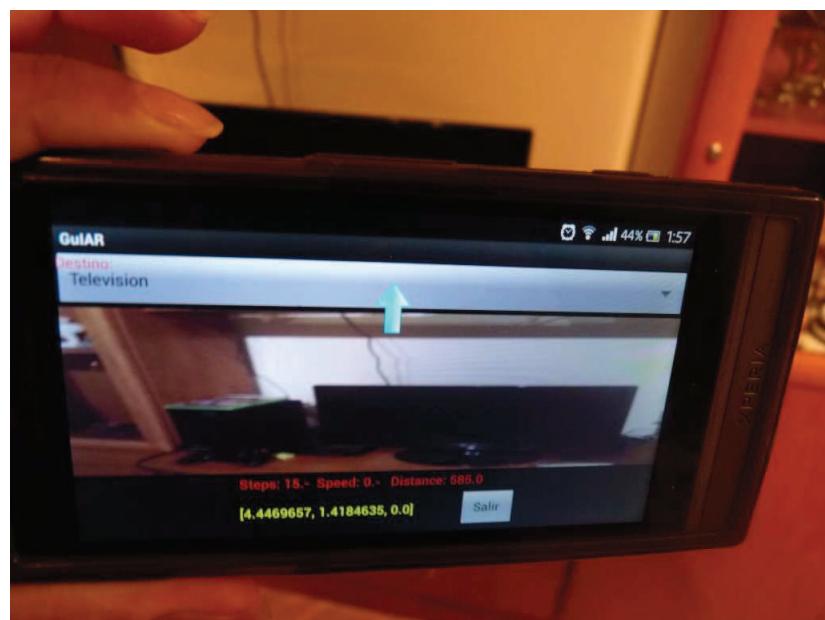


Figura 4.13: Indicación de GuIAR orientando a seguir recto

Capítulo 5

Conclusiones y Trabajo Futuro

Actualmente la Realidad Aumentada se ha vuelto una de las tendencias más populares en el Internet de las cosas y esto sumado con el sistema de localización en recintos cerrados, se obtiene la Realidad Aumentada en Interiores, convirtiéndose en una potente herramienta para cualquier campo de aplicación.

En esta memoria se presentó al principio los objetivos del proyecto y estos en su totalidad han sido cumplidos. Cabe resaltar que este trabajo ha proporcionado diferentes insumos que permiten entender, analizar y tratar la Realidad Aumentada en Interiores, para ello se exploró extensivamente el estado de la cuestión presentando no sólo las aplicaciones relevantes a fecha de hoy, sino también las diferentes plataformas móviles, los sistemas de localización en interiores y los distintos componentes base incorporados en este trabajo. Todo esto proporciona un completo trasfondo a cualquier interesado en el problema.

De igual manera este trabajo no solo explora el estado de la cuestión sino que además incorpora varios elementos de esta en una implementación real de posicionamiento en interiores (GuIAR) con el fin de evaluar de primera mano estos elementos. Esta implementación se realizó dentro del contexto de los proyectos MILES y REHABILITA, como objetivo principal de este trabajo. Es importante mencionar que los objetivos planteados para este trabajo fueron bastante ambiciosos y también un alto grado de incertidumbres desde el principio, lo cual se ha limitado el desarrollo y a la vez planteado grandes retos. Sin embargo, a medida que se analizaba el estado de la cuestión y se fue implementando la aplicación GuIAR; de una u otra forma se han ido solventando algunas dificultades, mientras que otras han afectado en mayor medida, como el caso de tener componentes base aún en fase de desarrollo, provocando así reestructuración de clases y código, de forma que puede afectar el progreso del mismo.

Una de las dificultades más significativas que se han tenido a lo largo de este trabajo, proviene del hecho de enfrentarnos al desafío de la Realidad Aumentada en Interiores, siendo una tendencia aún novedosa y poco explorada; que toca tener en cuenta muchos aspectos no tan triviales.

5.1. TRABAJO FUTURO

Otras problemáticas que se ha comprobado al realizar experimentaciones, es en el funcionamiento de la brújula ó giróscopo del dispositivo móvil, se ve afectada ante cualquier corriente eléctrica o campo magnético que detecte a su alrededor. También el sistema de navegación inercial utilizado para obtener el posicionamiento en interiores, tiene unas variaciones erróneas en la aceleración, obteniéndose un error acumulado de forma creciente, por lo que esto afecta en gran medida el funcionamiento de la aplicación y limitó las capacidades del proyecto.

Además este tipo de aplicaciones tienen la peculiaridad de que son demasiado personalizadas, es decir, sólo dan funcionalidad al escenario para el cual fue creado, por ejemplo una de las variables indispensables relacionadas con la orientación respecto al eje, como la Azimuth, saber el ángulo de orientación de los ejes de coordenadas del plano, con respecto al norte. Todo este tipo de dificultades que se han presentado a la hora de desarrollar la aplicación GuIAR, ha impedido llevar a cabo algunas de las ideas que se tenían inicialmente planteadas, pero que no afecta a los objetivos establecidos.

Sin embargo, el posicionamiento en interiores es una área poco explorada y que alberga una gran complejidad, sin despreciar los importantes esfuerzos de investigación y experimentación con varios tipos de tecnologías, para en un futuro tener resultados óptimos en este sistema de localización.

Este proyecto realizado no sólo permitió la construcción de la aplicación de Realidad Aumentada en Interiores GuIAR, sino también permitió entender mejor el problema, a través del análisis, diseño, implementación y experimentación del mismo, lo cual ha sido un trabajo arduo, pero enriquecedor y positivo, a pesar de las expectativas iniciales demasiadas optimistas, se pudo aprender de la experiencia ocasionada y fomentar de ésta forma una base para futuros desarrollos en esta área.

Finalmente, también puedo decir que se cumplieron objetivos a nivel personal, como el refuerzo y nuevo conocimiento tanto a nivel de programación como en lo relacionado al mundo laboral. Uno de los objetivos marcados era aprender por mí misma a trabajar en algo que no hubiera visto en ninguna asignatura. En ese sentido la experiencia ha sido muy positiva, siendo que todos los requisitos marcados para la culminación de la aplicación han sido cumplidos.

5.1. Trabajo Futuro

Este trabajo al ser parte de una tendencia novedosa, incentiva en mayor medida el desarrollo de futuras líneas de acción sobre la Realidad Aumentada en Interiores, por lo que sería interesante que se emplearan las siguientes estrategias ó características como trabajo futuro:

5.1. TRABAJO FUTURO

- Posicionamiento en interiores: la Realidad Aumentada en Interiores, necesita de una buen posicionamiento dentro de un edificio pues es el objetivo principal y en este trabajo sólo se utilizó el Sistema de Navegación Inercial, lo cual para poder obtener un mejor posicionamiento, este sistema se debería complementar con el uso de otra infraestructura como la red inalámbrica del edificio (WiFi), teniendo un buen conocimiento de la misma, haciendo uso de mediciones absolutas de la posición dentro del edificio y para mayor presición de la posición, se debería tener algunas marcas dentro del edificio que permita calibrar la orientación en espacios largos y de esta forma habrá una sincronización con respecto al plano del escenario donde se realicen las experimentaciones.
- Incluir reconocimiento de voz: sería una gran ventaja que la aplicación GuIAR se le añade el reconocimiento de voz, para la interacción interfaz gráfica y usuario, y a su vez que el dispositivo móvil responda a su solicitud configurada.
- Incluir animaciones en realidad aumentada: sería muy interesante poder incluir animaciones en la escena aumentada con la herramienta Unity3D de (Creighton, 2010) (herramienta para desarrollo de videojuegos), por lo que se plantea un estudio de alternativas para integrar motores de renderizado ó poder construir uno propio, de manera que se puedan incluir objetos 3D con formatos que soporten animaciones, como por ejemplo el formato md2.
- Emplear un filtro de partículas: para emplear este filtro que permite mejorar la estimación de la posición del dispositivo dentro del edificio, sería necesario conocer el plano del interior del edificio y cargarlo dentro del móvil. Estos dos últimos casos expuestos, hace que la información sobre la posición de la persona dentro de un edificio sea más aproximada ó al menos que el margen de error sea mínimo, debido a que entre más información se tenga del entorno y del movimiento del dispositivo móvil, más precisa será estimación de su posición.
- Tratamiento de rotaciones: para próximas versiones se podría realizar mejoras en las técnicas de tratamiento de las rotaciones ó cálculo de orientación y escalas en cuanto al tiempo de respuesta que actualmente se tienen.

Finalmente es importante para cualquier tipo de mejora, seguir investigando el desarrollo de sistemas de Realidad Aumentada en Interiores, además ver la posibilidad de que la aplicación sea independiente de la plataforma (arquitecturas dirigidas por modelos) ó si se pudiese tener los servicios e información en la computación en la nube, para que la aplicación no sea tan pesada en cuanto a ejecución.

Referencias

- A, D. A. (2012). An approach for enriching virtual environments with semantic spatial information and personalized user guidance.
- AndAR, A. (2010). *Andar, android augmented reality*. Descargado de <http://code.google.com/p/andar/>
- Android, G. (2013). *Sistema operativo android y tutorial para el desarrollo de aplicaciones*. Descargado de <http://www.android.com/>, <http://developer.android.com/>
- Angelica De Antonio, G. M., Ricardo Imbert. (2005). Intelligent virtual environments for training: An agentbased approach. multiagent systems and applications. , 82–91.
- Antoniades, C. N., y Stephanedes, Y. J. (1996). Single-station incident detection algorithm (ssid) for sparsely instrumented freeway sites. En *Applications of advanced technologies in transportation engineering* (1995) (pp. 218–221).
- Arias, L., Fuentes, M., y García, R. (s.f.). Análisis de las técnicas de localización y posicionamiento en los sistemas de telecomunicaciones.
- Bellón Alcarazo Sergio, S. L. n., Creixell Rojo Jorge. (2011). *Look: Framework para aplicaciones de realidad aumentada en android*. Tesis de Master no publicada, Universidad Complutense de Madrid. Descargado de <http://www.lookar.net/>
- Black, P. E. (2006). Manhattan distance. En *Dictionary of algorithms and data structures, nist*.
- Burnette, E. (2012). *Programación android* (Vol. 1). Juan Ignacio Luca de Tena,15. 28027 Madrid: Ediciones Anaya Multimedia S.A.
- Claeys, J. (2013). *Polar coordinates*. Descargado de <http://www.ping.be/  ping1339/polar.htm>
- Corporation, I. (2010). *Information center ibm. ibm websphere mq* (n.º 7). Descargado de
- Creighton, R. H. (2010). *Unity 3d game development by example: Beginner's guide*. Packt Publishing.
- Crockford, D. (2006). The application/json media type for javascript object notation (json).
- Enrique, O. R. C. (2011). Realidad aumentada en medicina. , 18(1), 213–246.
- Fielding, R. T. (2000). *Architectural styles and the design of networkbased software architectures*. Tesis Doctoral no publicada, Universidad of California. Descargado de http://www.ics.uci.edu/  fielding/pubs/dissertation/fielding_dissertation.pdf

Referencias

- Froján, J. E. P., y Lorenzo, A. G. (2011). Aplicación de los códigos bidimensionales qr (quick response) en la prestación de los servicios de mantenimiento y asistencia técnica. En *V international conference on industrial engineering and industrial management* (pp. 532–541).
- G, W. (2004). Dictionary of philosophy of mind ontology [Manual de software informático]. Descargado de <http://www.artsci.wustl.edu/~philos/MindDict/ontology.html>
- Gironés, J. T. (2013). *Android: Programación de aplicaciones* (Vol. 1). Universidad Politécnica de Valencia. Descargado de https://www.miriadax.net/web/android_programacion
- Guzmán Ullán, C. (2011). Guía virtual sensible al contexto mediante códigos de barras bidimensionales.
- Hatami, A. (2006a). Application of channel modeling for indoor localization using toa and rss.
- Hatami, A. (2006b). Application of channel modeling for indoor localization using toa and rssn. *Electrical & Computer Engineering*.
- Hazas, M. (2004). Location-aware computing comes of age. , 95–97.
- Hristov, A. (2012). *Desarrollo de aplicaciones para móviles y tabletas android* (Vol. 1). Planetalia.
- Hunkeler, U., Truong, H. L., y Stanford-Clark, A. (2008). Mqtt-sa publish/subscribe protocol for wireless sensor networks. En *Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on* (pp. 791–798).
- Inc, A. (2013). *Sistema operativo ios*. Descargado de <http://www.apple.com/es/ios/>
- Instituto de Rehabilitación Guttmann., E. T. H. S. (2011). Diseño de los procedimientos en escenarios de rehabilitación, funcional, cognitiva, cardíaca y respiratoria.
- Julier, S. J., y Uhlmann, J. K. (1997). Consistent debiased method for converting between polar and cartesian coordinate systems. En *Aerosense'97* (pp. 110–121).
- Kalman, R. E. (1960). A new approach to linear filtering and prediction problems, transactions of the asme. , 82, 35–45.
- Klyne G, J. J. C. (2004). *Resource description framework rdf: Concepts and abstract syntax*. Descargado de <http://www.w3.org/TR/rdf-concepts/>
- LaMarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I., Scott, J., ... others (2005). Place lab: Device positioning using radio beacons in the wild. En *Pervasive computing* (pp. 116–133). Springer.
- Madden, L. (2011). *Professional augmented reality browsers for smartphones: programming for junaio, layar and wikitude*. Wiley. com.
- McGuinness, D., y Van Harmelen, F. (2003). *Web-ontology (webont) working group*.
- Mixare. (2012). *Mixare augmented reality engine*. Descargado de <http://www.mixare.org/>
- Noy, N. F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R. W., y Musen, M. A. (2001). Creating semantic web contents with protege-2000. *Intelligent Systems, IEEE*, 16(2), 60–71.
- Ocana, M., Bergasa, L. M., Sotelo, M. A., Revenga, P., y Flores, R. (s.f.). Sistema de localizacion de personas basado en medida de la senal wifi.

Referencias

- Palazón, J. A., Gozalvez, J., y Prieto, G. (s.f.). Implementación hardware de un sistema de localización en interiores basado en redes sensoriales inalámbricas.
- Perez, A. (por defender). *Cliente móvil para sistema de guiado semántico en entornos educativos*. Tesis de Master no publicada, Universidad Politécnica de Madrid.
- Prieto Tejedor, J., y cols. (2012). Estimación adaptativa bayesiana aplicada a la localización de usuarios móviles.
- S.A., R. G. S. G. I. (2009). Tecnologías disruptivas para la rehabilitación del futuro.
- Sandoval, J. (2009). *Restful java web services: Master core rest concepts and create restful web services in java*. Packt Publishing.
- Shepard, S. (2005). *Rfid: radio frequency identification*. McGraw-Hill New York.
- Snyder, B., Bosnanac, D., y Davies, R. (2011). *Activemq in action*. Manning.
- Soto, G., Peinado, A., y Ortiz, A. (s.f.). Sistema rfid para publicidad personalizada no invasiva a través de pantallas.
- Strathern, P. (1999). *Pitágoras y su teorema*. Siglo XXI de España Editores.
- Sun Microsystems, I. (2012). *Java message service tutorial*. Descargado de <http://docs.oracle.com/javaee/1.3/jms/tutorial/index.html>
- Sun Microsystems, I. (2013). *Java message service tutorial*. Descargado de <http://www.json.org/>
- W., E. (2002). *The notational conventions i adopted, and why, ewd 1300*. Tesis Doctoral no publicada, Universidad of California. Descargado de <http://www.cs.utexas.edu/users/EWD/ewd13xx/EWD1300.PDF>
- W3Schools. (2006). *Introduction to owl*. Descargado de http://www.w3schools.com/rdf/rdf_owl.asp

Apéndices

Dispositivos Móviles

Para las experimentaciones realizadas de la aplicación GuIAR desarrollado en este proyecto, se han utilizado dos SmartPhones de diferentes versiones y modelos, pero ambos de la plataforma Android.

.1. Smartphone I

El primer dispositivo de referencia que se ha utilizado para la experimentación, ha sido el Samsung Galaxy S-Plus¹. Las características de éste móvil son las siguientes:

- Samsung GALAXY S Plus
 - Procesador de 1,4 GHz y conexión HSPA de hasta 14,4 Mbps
 - Batería de 1.650 mAh
 - Sistema Operativo Android 2.3
 - Resolución: WVGA (800 x 480)
 - Navegación GPS
 - Wifi
 - Bluetooth
 - Sensores:
 - Acelerómetro
 - Campo magnético
 - Orientación o brújula
 - Giroscopio (este móvil no dispone de giróscopo, si llegase a tenerlo sería mejor la precisión de los datos)

¹<http://www.samsung.com/es/consumer/mobile-phone/smartphones/galaxy/GT-I9001HKDFOP>

.2. SMARTPHONE II



Figura 1: Smartphone Samsung Galaxi S Plus

.2. Smartphone II

El segundo dispositivo de referencia es el Sony Xperia S.² Tiene como características las siguientes:

- Sony Xperia S

- Procesador Qualcomm MsM8260 dual-core 1.5GHZ, GPU Adreno 220
- Memoria interna de 16GB/32GB, 1GB RAM
- Batería Standard, Li-Ion 1750 mAh
- Sistema Operativo Android 4.1.2 Jelly Bean
- Navegación GPS con soporte A-GPS
- Wifi 802.11 b/g/n; DNLA
- Cámera de 12 MP, 4000x3000 pixels
- Bluetooth v2.1 A2DP, EDR
- Sensores:
 - Acelerómetro
 - Giróscopo
 - Brújula digital
 - Sensor de proximidad para auto apagado

²<http://www.sonymobile.com/es/products/phones/xperia-s/>

.2. SMARTPHONE II



Figura 2: Smartphone Sony Xperia S

Instalación de GuIAR

Instalar el archivo ejecutable Android de la aplicación **GuIAR.apk** (archivo generado después de haberse compilado la aplicación Android en la herramienta de desarrollo Eclipse) es realmente sencillo, pero se debe tener en cuenta ciertos pasos a seguir la primera vez que se entra en contacto con Android.

Antes de iniciar con la instalación, es importante recordar habilitar la opción de poder instalar archivos que no sean del Market, ya que si no, no nos dejará realizar la instalación. Para ello se debe realizar lo siguiente:

Entrar en el menú **Ajustes** luego **Aplicaciones** del sistema Android y después habilitar la opción **Orígenes desconocidos**

Una vez hecho ésto, para instalar **GuIAR.apk** en el smartphone, simplemente se descarga el fichero y pasarlo a una carpeta de la micro SD (o el sistema de almacenamiento que utilicemos) del dispositivo u otra forma más sencilla es enviarnos el apk por correo electrónico, de tal forma que podamos descargarlo en nuestro Android y tenerlo en memoria. Una vez hecho cualquiera de las dos opciones anteriores, para instalar el archivo **GuIAR.apk** en Android, desde el gestor de archivos del Smartphone nos moveremos hasta la carpeta de la micro SD u otra ubicación donde se ha dejado el archivo y lo ejecutamos; luego el sistema instalará la aplicación GuIAR y sólo nos quedará comprobar que la aplicación se ha instalado correctamente.

NOTA: *Es importante mencionar que lo ideal siempre es instalar todas nuestras aplicaciones directamente desde el Android Market, ya que de esta forma evitaremos problemas. Esto es debido a que si descargamos un archivo .apk de un sitio poco fiable, podría contener código malicioso que infecte nuestro Smartphone, ya que no olvidemos que los Smartphones son similares a un ordenador. Por ello, sólo es recomendable instalar archivos apk en Android en casos en los que no sea posible conseguirlo a través del Market y en caso de que no tengamos alternativa, asegurarnos de que el sitio desde el cual lo descargamos es un sitio de confianza.*

Por último, si se quiere desinstalar la aplicación GuIAR, se podrá hacer desde **Ajustes** luego **Aplicaciones** y seleccionar **Administrar aplicaciones**. Luego pulsar sobre la aplicación GuIAR que queremos eliminar y en la siguiente pantalla pulsar sobre la opción **Desinstalar**.

Herramientas de Desarrollo

Para el desarrollo de la aplicación GuIAR ha sido necesaria la utilización de las siguiente herramientas de software:

- **Eclipse³**: entorno de desarrollo integrado de código abierto multiplataforma.
- **JDK⁷⁴**: Kit de desarrollo Java.
- **SDK Android⁵**: Kit de desarrollo para aplicaciones Android.
- **ActiveMQ⁶**: bróker de mensajerías open source
- **Tomcat⁷**: Servidor web de código abierto que soporta tecnología J2EE de java

³<http://www.eclipse.org/>

⁴<http://www.oracle.com/technetwork/java/javaee/downloads/java-ee-sdk-6u3-jdk-6u29-downloads-523388.html>

⁵<http://developer.android.com/sdk/index.html>

⁶<http://activemq.apache.org>

⁷<http://tomcat.apache.org/>

Estructura del código fuente

En cuanto a código fuente, la estructura de desarrollo de GuIAR se encuentra todo dentro del paquete **es.upm.guiar**, tal como se muestra en la figura 3.

La aplicación GuIAR se trabajó bajo la versión de Android 2.3.3 (versión mínima que contiene sensores necesarios para la aplicación). Dentro del paquete **es.upm.guiar** se tiene las clases principales java de la aplicación (éstas se explicaron en el capítulo 3), dentro de la carpeta **res** se encontrará toda la parte gráfica de la aplicación, como: imágenes, ficheros XML (con descripción detallada del contenido gráfico de GuIAR, como: tamaño y color de la fuente, ubicación y nombre de las imágenes a incluir, posición, etc.). Por último se tiene uno de los ficheros XML más importantes de una aplicación de Android y es el **AndroidManifest.XML**, es aquel donde se declara cada Activity que se tiene para la aplicación GuIAR.



Figura 3: Estructura del código GuIAR

Codificaciones

A continuación se presenta el código fuente completo de cada una de las clases desarrolladas en éste trabajo.

es.upm.guiar.MainActivity

```
import com.example.sensorstest.AccelerometerCalibration;
import com.example.sensorstest.Constant;
import com.example.sensorstest.Trajectory;

import es.ucm.look.location.LocationManager;

public class MainActivity extends Activity implements SurfaceHolder.Callback{

    Camera camera;
    SurfaceView surfaceView;
    SurfaceHolder surfaceHolder;
    boolean previewing = false;
    LayoutInflator controllInflater = null;
    private static final int DIALOG_CALIBRATION_ACCEL_ID = 0;
    private SensorManager mSensorManager;
    private GuiaRecorrido mGuiaRecorrido;
    private Trajectory trajectoryManager;
    public static ObtenerJSON ojson;

    //variables imagenes flecha
    private ImageView irec;
    private ImageView ider;
    private ImageView iizq;
    //ImageView ides;

    //axis_orientation: orientación con respecto al eje X – Norte, dado por
    MILES
    public static double axis_orientation=295.0;

    //Obtener la URL ingresada por el usuario

    // obtener JSON

    private static String server=null;
    private static String solguia=null;
```

```

private static JSONObject json=null;
public static String url=null;
private Activity atv;

//Variables necesarias para obtener coordenadas de Look!
LocationManager location = null;
Timer timer = null;
public static boolean ins;
public static boolean wifi;

//variable sensor
private Sensor sensor;

//variables posicion del usuario
private double X_user;
private double Y_user;
private double Z_user;

private Double orientacion;

private String op_url="";

private double min_dist=50.0;
private double min_angle=5.0;

private Vector<PuntoInteres> camino;

public MainActivity(){
    super();
    op_url=url+”/puntosinteres”;
    ojson=new ObtenerJSON();
    ojson.execute();
    atv=this;
    camino=new Vector<PuntoInteres>();
    orientacion=new Double(0.0);
}

//Control del Spinner (listas de puntos de interes)
//private Spinner ListPuntosOrigen;
private Spinner ListPuntosDestino;

//Orientacion devuelve un número comprendido entre 0 y 359grados
private SensorEventListener mySensorEventListener = new
    SensorEventListener() {

        public void onAccuracyChanged(Sensor sensor, int accuracy) {
        }

        @Override
        public void onSensorChanged(SensorEvent event) {
            orientacion= new Double(event.values[0]);
        }
    };

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    setRequestedOrientation(ActivityInfo.
        SCREEN_ORIENTATION_LANDSCAPE);
}

```

```

getWindow().setFormat(PixelFormat.UNKNOWN);
surfaceView = (SurfaceView) findViewById(R.id.camerapreview);
surfaceHolder = surfaceView.getHolder();
surfaceHolder.addCallback(this);
surfaceHolder.setType(SurfaceHolder.
    SURFACE_TYPE_PUSH_BUFFERS);
controlInflater = LayoutInflater.from(getApplicationContext());
View viewControl = controlInflater.inflate(R.layout.control, null);
LayoutParams layoutParamsControl
= new LayoutParams(LayoutParams.FILL_PARENT,
    LayoutParams.FILL_PARENT);
this.addView(viewControl, layoutParamsControl);

//imagenes flecha del layout control
irec = (ImageView) findViewById(R.id.image_rec);
ider = (ImageView) findViewById(R.id.image_der);
iizq = (ImageView) findViewById(R.id.image_izq);

mSensorManager = (SensorManager) this.getSystemService(Context
    .SENSOR_SERVICE);
mGuiaRecorrido = new GuiaRecorrido(this);

//Obtener la posición del usuario adicionando el componente Look!
ins= true;
location = new LocationManager(this.getApplicationContext(), ins,
    wifi);
location.start();
timer = new Timer();

ListPuntosDestino=(Spinner)findViewById(R.id.ListPuntosDestino);
ListPuntosDestino.setOnItemSelectedListener(new
    OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View
            view, int position, long id) {
            try{
                //Object puntoDestino =
                ListPuntosDestino.getSelectedItem();
                String puntoDestino = (String)parent.
                    getItemAtPosition(position);
                //Mostrar mensaje del destino seleccionado,
                //en un corto periodo de tiempo
                Toast.makeText(getApplicationContext(), "
                    Destino:" +puntoDestino, Toast.
                    LENGTH_SHORT).show();
                String rurl=url+"/solicitarguia?x=" +X_user
                    +"&y=" +Y_user+"&destino=" +
                    URLEncoder.encode(puntoDestino, "
                    UTF-8");
                JSONParser jParser = new JSONParser();
                String out=jParser.getRawFromUrl(rurl);
                JSONArray jsonRuta = new JSONArray(
                    out);
                synchronized (camino) {
                    camino=new Vector<PuntoInteres
                        >();
                    for(int i=0;i<jsonRuta.length();i
                        ++){
                        JSONObject o=jsonRuta.
                            getJSONObject(i);
                        PuntoInteres pi=new
                            PuntoInteres(o.

```

```

                getDouble("x"), o.
                getDouble("y"), o.
                getString("label"));
            camino.add(pi);
        }
        Toast.makeText(
           (getApplicationContext(), "
            Dirijase a: "+camino.get(0).
            getLabel(), Toast.
            LENGTH_SHORT).show();
    }
} catch(Exception e){
    e.printStackTrace();
}
}

@Override
public void onNothingSelected(AdapterView parent) {
    //Mostrar mensaje de solicitud de selección del
    destino, en un corto periodo de tiempo
    Toast.makeText(getApplicationContext(), "
    Selecciona tu destino..", Toast.
    LENGTH_SHORT).show();
}

TimerTask timerTask = new TimerTask() {
    public void run() {
        runOnUiThread(new Runnable() {
            public void run() {
                ((TextView) findViewById(R.id.
                    textCoord)).setText(String.
                    valueOf(location.getPosition().
                    toString()));
                X_user=location.getPosition().x
                    *100.0;
                Y_user=location.getPosition().y
                    *100.0;
                Z_user=location.getPosition().z
                    *100.0;

                double fixed_orientation=0.0;
                synchronized (orientacion) {
                    if(orientacion+90.0<360.0)
                        fixed_orientation=
                            orientacion
                            +90;
                    else
                        fixed_orientation=((
                            orientacion
                            +90) %360.0;
                }

                synchronized (camino) {
                    if(camino.size()>0){
                        PuntoInteres pi=
                            camino.get(0);
                        PuntoInteres ac=
                            new
                            PuntoInteres(
                            X_user, Y_user

```

```

        , "actual");
    if(PuntoInteres.
        distancia(pi,
        ac)<min_dist)
    {
        camino.
            remove
            (0);
        pi=camino.
            get(0);
        Toast.
            makeText
            (
            getApplicationContext
            (), "
            Dirijase
            a: "+pi.
            getLabel
            ());
        Toast.
            LENGTH_SHORT
            ).show
            ();
    }
    String direccion=
        obtenerOrientacionDestino
        (
        fixed_orientation
        ,X_user,Y_user
        ,pi.getX(),pi.
        getY());
        mostrarFlecha(
        direccion);
    }
}
});
}
);
};

timer.scheduleAtFixedRate(timerTask, 0, 500);
super.onResume();

//Orientacion
sensor = mSensorManager.getDefaultSensor(Sensor.
    TYPE_ORIENTATION);
mSensorManager.registerListener(mySensorEventListener, sensor,
    SensorManager.SENSOR_DELAY_NORMAL);

}

//Debido a que al realizar esta tarea larga, se creo una clase asincrona para
// poder obtener los datos
private class ObtenerJSON extends AsyncTask {
    @Override
    protected Object doInBackground(Object... arg0) {
        List<String> listaPuntos = null;
        ListPuntosDestino = (Spinner)findViewById(R.id.
        ListPuntosDestino);
        try{
            if(json==null){

```

```

                JSONParser jParser = new JSONParser();
                json=new JSONObject();
                json = jParser.getJSONFromUrl(op_url);
            }
            listaPuntos=getNames(json);
            if(listaPuntos!=null){
                ArrayAdapter<String> adp1 = new
                    ArrayAdapter<String> (atv,android.R.
                        layout.simple_spinner_item,listaPuntos
                    );
                adp1.setDropDownViewResource(android.R.
                    layout.simple_spinner_dropdown_item);
                ListPuntosDestino.setAdapter(adp1);
                //adp1.notifyDataSetChanged();

                //obtener el valor de la selección del punto
                //destino que desea el usuario por el
                //spinner
            }

        }

    }catch(Exception e){
        e.printStackTrace();
    }
    return null;
}

protected void onPostExecute(Boolean result) {
}

}

//Debido a que el método JSONObject.getName no existe en el org.json API
//de Android creamos el siguiente método, para retornar la clave
//http://developer.android.com/reference/org/json/JSONObject.html
private static List<String> getNames( JSONObject jo ) throws
    JSONException {

    int length = jo.length();
    if (length == 0) return null;
    ArrayList<String> names = new ArrayList<String>();
    JSONArray ja=jo.names();
    for(int i=0;i<ja.length();i++){
        names.add(ja.getString(i));
        Log.i("JSON", "NAMES: " + names.get(i));
    }
    return names;
}

@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width,
    int height) {
    // TODO Auto-generated method stub
    if(previewing){
        camera.stopPreview();
        previewing = false;
    }

    if (camera != null){
        try {
            camera.setPreviewDisplay(surfaceHolder);

```

```

        camera.startPreview();
        previewing = true;
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

@Override
public void surfaceCreated(SurfaceHolder holder) {
    // TODO Auto-generated method stub
    camera = Camera.open();
}

@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    // TODO Auto-generated method stub
    camera.stopPreview();
    camera.release();
    camera = null;
    previewing = false;
}

@Override
protected Dialog onCreateDialog(final int id) {
    final Builder builder;
    final Dialog dialog;

    switch (id) {
        case DIALOG_CALIBRATION_ACCEL_ID:
            dialog = AccelerometerCalibration.getDialog(this,
                mSensorManager, mGuiaRecorrido.
                getTrajectoryManager());
            break;
        default:
            dialog = null;
    }

    return dialog;
}

@Override
protected void onResume() {
    super.onResume();
    mSensorManager.registerListener(mGuiaRecorrido, mSensorManager
        .getDefaultValue(Sensor.TYPE_ACCELEROMETER),
        Constant.SENSOR_DELAY);
    mSensorManager.registerListener(mGuiaRecorrido, mSensorManager
        .getDefaultValue(Sensor.TYPE_GYROSCOPE),
        Constant.SENSOR_DELAY);
    mSensorManager.registerListener(mGuiaRecorrido, mSensorManager
        .getDefaultValue(Sensor.TYPE_MAGNETIC_FIELD),
        Constant.SENSOR_DELAY);

    // Recalibraremos el aceler?metro
    this.showDialog(MainActivity.
        DIALOG_CALIBRATION_ACCEL_ID);
}

@Override
protected void onStop() {
}

```

```

super.onPause();
//Parada de SensorTest
mSensorManager.unregisterListener(mGuiaRecorrido);
//Parada de Look
timer.cancel();
location.stop();
finish();
}

//Obtengo los datos de los sensores y muestro en pantalla los resultados
public void updateSensorData(int steps,double speed,double acc,double
distance, Matrix sg, Matrix sl){
    DecimalFormat df = new DecimalFormat("#.###");
    TextView txtSensores = (TextView)findViewById(R.id.textSensor);
    txtSensores.setTextColor(Color.RED);
    txtSensores.setText("Steps: " +steps+ ".- Speed: " +df.format(speed)
        + ".- Distance: " +distance);
}

private String point2string(Matrix x,double scl){
    DecimalFormat df = new DecimalFormat("#.###");
    String txt="";
    txt=( "+"+df.format(x.get(0,0)/scl)+"," +df.format(x.get(1,0)/scl)+",
        "+df.format(x.get(2,0)/scl)+");
    return txt;
}

private void mostrarFlecha(String direccion){
    if(direccion.equals("RECTO")){
        irec.setVisibility(View.VISIBLE);

    }else if(direccion.equals("DERECHA")){
        ider.setVisibility(View.VISIBLE);

    }else if(direccion.equals("IZQUIERDA")){
        iizq.setVisibility(View.VISIBLE);

    }
}
/*
 * Parametros
 * orientacion: Orientacion dada por el sensor del movil
 * cx, cy: Coordenadas X y Y actuales del usuario
 * tx, ty: Coordenadas X y Y del punto de ruta.
 */
private String obtenerOrientacionDestino(double orientation,double cx,
double cy,double tx,double ty){
    //Calcula la orientacion del movil con respecto al eje de coordenadas
    //de referencia
    double relative_orientation=orientation-axis_orientation;
    if(relative_orientation<0)
        relative_orientation=360+relative_orientation;
    Log.i("RO", ""+relative_orientation);
    //Calcula el angulo de orientacion entre el movil, el eje de referencia
    //y el punto de ruta
    double t_abs_or=Math.toDegrees(Math.atan2(ty-cy, tx-cx));
    if(t_abs_or>0)
        t_abs_or=360-t_abs_or;
    else
        t_abs_or=Math.abs(t_abs_or);
}

```

es.upm.guiar.JSONParser

```
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.util.Log;

public class JSONParser {

    static InputStream is = null;
    static JSONObject jObj = null;
```

```

static String json = "";
long milis=0;

// constructor
public JSONParser() {

}

public JSONObject getJSONFromUrl(String url) {
    try {
        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpGet httpGet =new HttpGet(url);
        milis=System.currentTimeMillis();
        Log.i("JSONPARSER", "Getting URL:" + url);
        HttpResponse httpResponse = httpClient.execute(httpGet);
        Log.i("JSONPARSER", "Got URL:" + url + ":"+(System.
                currentTimeMillis()-milis));
        HttpEntity httpEntity = httpResponse.getEntity();
        is = httpEntity.getContent();
    }

    catch (Exception e) {
        Log.i("JSONPARSER", "Error Getting URL:" +e.
                getMessage() + ":" +e.getLocalizedMessage());
    }

    try {
        BufferedReader reader = new BufferedReader(new
            InputStreamReader(
                is, "iso-8859-1"), 8);
        StringBuilder sb = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null) {
            sb.append(line + "\n");
        }
        is.close();
        json = sb.toString();
        Log.i("JSON Parser", "valor en parsing:" +json);
    }

    catch (Exception e) {
        Log.e("Buffer Error", "Error converting result " + e.toString
            ());
    }

    // try parse the string to a JSON object
    try {
        jObj = new JSONObject(json);
    }

    catch (JSONException e) {
        Log.e("JSON Parser", "Error parsing data " + e.toString());
    }

    // return JSON String
    return jObj;
}

public String getRawFromUrl(String url) {
    try {
        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpGet httpGet =new HttpGet(url);
        milis=System.currentTimeMillis();
        Log.i("JSONPARSER", "Getting URL:" + url);
        HttpResponse httpResponse = httpClient.execute(httpGet);
    }
}

```

```

        Log.i(" JSONPARSER", "Got URL:" + url + ":"+(System.
                currentTimeMillis() - milis));
        HttpEntity httpEntity = httpResponse.getEntity();
        is = httpEntity.getContent();

        BufferedReader reader = new BufferedReader(new
                InputStreamReader(is, "iso-8859-1"), 8);
        StringBuilder sb = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null) {
            sb.append(line + "\n");
        }
        is.close();
        json = sb.toString();
        return json;

    } catch (Exception e) {
        Log.i(" JSONPARSER", "Error Getting URL:" +e.
                getMessage() + ":" +e.getLocalizedMessage());
    }
    return null;
}

```

es.upm.guiar.Menu

```

import org.fusesource.mqtt.client.Callback;
import org.fusesource.mqtt.client.FutureConnection;
import org.fusesource.mqtt.client.MQTT;
import org.fusesource.mqtt.client.Message;
import org.fusesource.mqtt.client.QoS;
import org.fusesource.mqtt.client.Topic;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;

public class Menu extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu);

    }

    public void lanzarInicio(View view){
        Intent i = new Intent(getApplicationContext(), MainActivity.class);
        startActivity(i);
    }

    public void lanzarConfigurar(View view){
        Intent i = new Intent(this, Configurar.class);
        startActivity(i);
    }
}

```

```

public void lanzarAcercaDe(View view){
    Intent i = new Intent(this, Acerca_De.class);
    startActivity(i);
}

public void salirMenu(View view) {
    finish();
}

```

es.upm.guiar.GuiaRecorrido

```

package es.upm.guiar;

import jama.Matrix;

import java.text.DecimalFormat;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import com.example.sensorstest.DataTracker;
import com.example.sensorstest.InertialMagnitude;
import com.example.sensorstest.OnUpdateListener;
import com.example.sensorstest.RouteView;
import com.example.sensorstest.SensorData;
import com.example.sensorstest.SensorType;
import com.example.sensorstest.Trajectory;
import com.example.sensorstest.OrientationFilter;

public class GuiaRecorrido extends View implements SensorEventListener,
    OnUpdateListener{

    private static final String TAG = RouteView.class.getSimpleName();

    /**
     * Nombre de los datos de tipo gyro
     */
    private static final String GYRO = "rrx,rry,rrz,t";
    /**
     * Nombre de los datos de tipo accl
     */
    private static final String ACCL = "ax,ay,az,t";
    private static final String MAGN = "mx,my,mz,t";

    private DataTracker<SensorData> dataTracker;

    private final float mScaleFactor = 10f;
    private float mScale;
    private float mYOffset;
    private float mMaxX;
    private final float mSpeed = 1.0f;
    private float mLastX;

    // TrajectoryManager
    private final Trajectory trajectoryManager;
    private final OrientationFilter orientationFilterManager;
    // Anaolena

```

```

private double totalDistance =0.0;
private int previous_steps =0;
private MainActivity ctx;
private int refresh=0;

public GuiaRecorrido(final MainActivity context) {
    super(context);
    ctx=context;
    trajectoryManager = new Trajectory();
    trajectoryManager.reset();
    OrientationFilterManager = new OrientationFilter();
    // Desviaci?n de 100 grados negativos respecto de la posici?n de la
    // im?gen
    trajectoryManager.setNorthDeviation((float) -(100 * (Math.PI /
        180)));
    trajectoryManager.setOnUpdateListener(this);
}

public Trajectory getTrajectoryManager() {
    return trajectoryManager;
}

public OrientationFilter getOrientationFilterManager() {
    return OrientationFilterManager;
}

@Override
public void onAccuracyChanged(final Sensor sensor, final int accuracy) {
}

@Override
protected void onDraw(final Canvas canvas) {
    synchronized (this) {
        if (mRouteBitmap != null) {
            final Matrix gs = trajectoryManager.get(
                InertialMagnitude.SPACE_GLOBAL);
            final Matrix o = trajectoryManager.get(
                InertialMagnitude.ORIENTATION);
            final Matrix accel = trajectoryManager.get(
                InertialMagnitude.LINEAR_ACCELERATION
            );
            final Matrix grav = trajectoryManager.get(
                InertialMagnitude.GRAVITY);
            final Matrix rawAccl = trajectoryManager.get(
                InertialMagnitude.RAW_ACCELERATION);
            final Matrix speed = trajectoryManager.get(
                InertialMagnitude.SPEED);

            // draw actual point in space
            final float x = (float) (gs.get(0, 0) + mWidth /
                3.0);
            final float y = (float) (-gs.get(2, 0) + mHeight /
                2.0);
            mRoute.drawPoint(x, y, mRoutePaint);

            // update the orientation mark
            mOrientation.drawColor(Color.TRANSPARENT,
                Mode.CLEAR);
        }
    }
}

```

```

mOrientationPaint.setColor(Color.RED);
mOrientation.save();
mOrientation.rotate((float) (o.get(1, 0) * 180 /
    Math.PI), x, y);
mOrientation.drawLine(x, y, x, y - 20,
    mOrientationPaint);
mOrientation.drawLine(x, y - 20, x - 10, y - 10,
    mOrientationPaint);
mOrientation.drawLine(x, y - 20, x + 10, y - 10,
    mOrientationPaint);
mOrientation.restore();

// flush the double buffer for painting the route

mText.drawRect(0, 0, 300, 150, mWhitePaint);
mText.drawText("steps: " + String.valueOf(
    trajectoryManager.getSteps()), 0, 25,
    mTextPaint);
// @Anaolena BEGIN
//mText.drawText("step speed: " + String.valueOf(
//    trajectoryManager.getStepSpeed()) + " steps/s",
//    0, 40, mTextPaint);
//mText.drawText("acceleration: " + String.valueOf(
//    trajectoryManager.getStepLongitude()) + " m",
//    0, 60, mTextPaint);
DecimalFormat df = new DecimalFormat("#.##")
;
float dt=trajectoryManager.getTimeDiff();
if(trajectoryManager.getSteps()>previous_steps)
    totalDistance+=dt*10*(trajectoryManager.
        getStepSpeed()*trajectoryManager.
        getStepLongitude());
previous_steps=trajectoryManager.getSteps();
mText.drawText("speed: " + df.format(
    trajectoryManager.getStepSpeed()*
    trajectoryManager.getStepLongitude()) + " cm
/s", 0, 55, mTextPaint);
mText.drawText("acceleration: " + String.valueOf(
    trajectoryManager.getStepLongitude()) + " cm
", 0, 85, mTextPaint);
mText.drawText("distance: " + String.valueOf(
    trajectoryManager.getStepLongitude()*
    trajectoryManager.getSteps()) + " cm", 0, 115,
    mTextPaint);
mText.drawText("distance2: " + df.format(
    totalDistance) + " cm", 0, 145, mTextPaint);

if (mLastX >= mMaxX) {
    mLastX = 0;
    final float yoffset = mYOffset;
    final float maxx = mMaxX;
    final float oneG = mScaleFactor * mScale;
    mSensorPaint.setColor(0xFFAAAAAA);
    mSensor.drawColor(0xFFFFFFFF);
    mSensor.drawLine(0, yoffset, maxx, yoffset,
        mSensorPaint);
    mSensor.drawLine(0, yoffset + oneG, maxx,
        yoffset + oneG, mSensorPaint);
}

```

```

        mSensor.drawLine(0, yoffset - oneG, maxx,
                         yoffset - oneG, mSensorPaint);
    }

    for (int i = 0; i < 3; i++) {
        mSensorPaint.setColor(mColors[i]);
        mSensorPaint2.setColor(mColors[i]);

        final float vA = (float) (mYOffset + accel.
            get(i, 0) * mScale);
        mSensor.drawLine(mLastX, mLastValuesA[i],
                         mLastX + mSpeed, vA,
                         mSensorPaint);
        mLastValuesA[i] = vA;

        /*
         * final float vB = (float) (mYOffset + gs.
         * get(i, 0) * mScale); mSensor.drawLine
         * (mLastX,
         * * mLastValuesB[i], mLastX + mSpeed, vB,
         * mSensorPaint2); mLastValuesB[i] =
         * vB;
         */
    }

    mLastX += mSpeed;
    // @Anaolena: elimina el dibujo de la aceleracion y
    // demas sensores
    // canvas.drawBitmap(mSensorBitmap, 0, 0, null);

}

}

@Override
public void onSensorChanged(final SensorEvent event) {

    switch (event.sensor.getType()) {
        case Sensor.TYPE_ACCELEROMETER:
            trajectoryManager.update(SensorType.ACCELERATION,
                event.values, event.timestamp);

            break;
        case Sensor.TYPE_GYROSCOPE:
            trajectoryManager.update(SensorType.ROTATION_RATE,
                event.values, event.timestamp);

            break;
        case Sensor.TYPE_MAGNETIC_FIELD:
            trajectoryManager.update(SensorType.MAGNETIC_FIELD,
                event.values, event.timestamp);

            break;
    }
    // Se obtiene el punto del espacio global y la orientación
    refresh++;
    if(refresh>100){
        final Matrix gs = trajectoryManager.get(InertialMagnitude.
            SPACE_GLOBAL);
    }
}

```

```

        final Matrix gl = trajectoryManager.get(InertialMagnitude.
            SPACE_LOCAL);
        final Matrix o = trajectoryManager.get(InertialMagnitude.
            ORIENTATION);

        // actual point (X,Y) in space
        final float x = (float) (gs.get(0, 0) + mWidth / 3.0);
        final float y = (float) (-gs.get(2, 0) + mHeight / 2.0);
        //final Matrix or = OrientationFilterManager.getOrientation
        ();
        //Se realiza el llamado en el MainActivity para mostrar los
        sensores
        ctx.updateSensorData(trajectoryManager.getSteps(),
            trajectoryManager.getStepSpeed()*trajectoryManager.
            getStepLongitude(),0,trajectoryManager.
            getStepLongitude()*trajectoryManager.getSteps(),gs,gl)
        ;//x,y,o);
        refresh=0;
    }

}

@Override
protected void onSizeChanged(final int w, final int h, final int oldw,
    final int oldh) {
    mRouteBitmap = Bitmap.createBitmap(2 * w / 3, h, Bitmap.Config
        .ARGB_8888);
    mOrientationBitmap = Bitmap.createBitmap(2 * w / 3, h, Bitmap.
        Config.ARGB_8888);
    mSensorBitmap = Bitmap.createBitmap(w / 3, h, Bitmap.Config.
        ARGB_8888);
    // @Anaolena
    //mTextBitmap = Bitmap.createBitmap(200, 100, Bitmap.Config.
        ARGB_8888);
    mTextBitmap = Bitmap.createBitmap(500, 500, Bitmap.Config.
        ARGB_8888);
    mMapBitmap = Bitmap.createBitmap(2 * w / 3, h, Bitmap.Config.
        ARGB_8888);

    mRoute.setImageBitmap(mRouteBitmap);
    mRoute.drawColor(0x00FFFFFF);

    mOrientation.setImageBitmap(mOrientationBitmap);
    mOrientation.drawColor(0xFFFFFFFF);

    mSensor.setImageBitmap(mSensorBitmap);
    mSensor.drawARGB(0, 0, 255, 0);

    mWidth = w;
    mHeight = h;

    mMaxX = w / 3;
    mLastX = w;

    mYOffset = h * 0.5f;
    mScale = -(h * 0.5f * (1.0f / (mScaleFactor * 2)));
}

super.onSizeChanged(w, h, oldw, oldh);
}

@Override
public boolean onTouchEvent(final MotionEvent event) {

```

```

        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            final Matrix pos = new Matrix(3, 1);
            pos.set(0, 0, event.getX() - 2 * mWidth / 3);
            pos.set(2, 0, mHeight / 2 - event.getY());

            trajectoryManager.resetStepCount();

            this.onSizeChanged(mWidth, mHeight, mWidth, mHeight);

            trajectoryManager.set(InertialMagnitude.SPACE_GLOBAL,
                pos);

            trajectoryManager.reset();

            totalDistance=0;
            previous_steps=0;

            mText.drawRect(0, 50, 200, 50, mWhitePaint);

            mText.drawText(String.format("point %d x %d", (int) (2 *
                mWidth / 3 - event.getX()), (int) (event.getY() -
                mHeight / 2)), 0, 90, mTextPaint);
        }

        return true;
    }

    /**
     * Asigna un Datatracker para que recoja la informaci?n generada por los
     * sensores.
     *
     * @param tracker
     * DataTracker que recibir? los datos
     *
     */
    public void setDataTracker(final DataTracker<SensorData> tracker) {
        dataTracker = tracker;
    }

    @Override
    public void update() {

        this.invalidate();
    }
}

```

es.upm.guiar.Inicio_GuIAR

```

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.view.animation.Animation.AnimationListener;
import android.widget.ImageView;

```

```

import android.widget.RelativeLayout;

/*Clase que permite visualizar la animación de GuiAR*/

public class Inicio_GuiAR extends Activity{
    private static final int ACTIVITY_RESULT = 1;

    private int sleepTime = 4000;
    protected Intent startIntent = null;

    private Handler altHandler = new Handler(){
        @Override
        public void handleMessage(Message msg) {
            final ImageView iv = (ImageView)findViewById(R.id.
                iv_inicio);
            Animation anim = AnimationUtils.loadAnimation(
                getBaseContext(), R.anim.zoom_exit);
            anim.setAnimationListener(new AnimationListener() {

                @Override
                public void onAnimationStart(Animation
                    animation) {}

                @Override
                public void onAnimationRepeat(Animation
                    animation) {}

                @Override
                public void onAnimationEnd(Animation animation
                    ) {
                    iv.setImageBitmap(null);
                    iv.invalidate();

                    if(!isFinishing() && (startIntent != null)){
                        if(getIntent().getExtras() != null)
                            startIntent.putExtras(
                                getIntent().getExtras()
                            );
                        startActivityForResult(startIntent,
                            ACTIVITY_RESULT);
                    }else{
                        finish();
                    }
                }
            });
            ((RelativeLayout)findViewById(R.id.rl_inicio)).
                startAnimation(anim);
        }
    };

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.inicio);

        Thread splashThread = new Thread() {

            @Override
            public void run() {

```

```

        try {
            Thread.sleep(sleepTime);
            if(altHandler != null)
                altHandler.sendMessage(Message.obtain());
        } catch (InterruptedException e) {
            Log.e("Splash", "", e);
        }
    };
    splashThread.start();
}

protected void onActivityResult (int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
        case ACTIVITY_RESULT:
            if( resultCode != Activity.RESULT_CANCELED ){
                setResult(Activity.RESULT_OK, data);
            }
            finish();
            break;
        default:
            break;
    }
}

```

es.upm.guiar.Visor_Inicio

```

import android.content.Intent;
import android.os.Bundle;

public class Visor_Inicio extends Inicio_GuiAR{
    protected void onCreate(Bundle savedInstanceState) {
        startIntent = new Intent(getApplicationContext(), Menu.class);
        //startIntent = new Intent(getApplicationContext(), MainActivity.class);
        super.onCreate(savedInstanceState);
    }
}

```

res.layout.activity_menu.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

```

```
        android:orientation="vertical" >

    <TextView
        android:id="@+id/text_bienvenida"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textSize="20sp"
        android:text="@string/Bienvenida"
        android:layout_marginBottom="20dip" />

    <Button
        android:id="@+id/button_inicio"
        android:layout_width="132dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:onClick="lanzarInicio"
        android:text="@string/Iniciar" />

    <Button
        android:id="@+id/button_configurar"
        android:layout_width="132dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:onClick="lanzarConfigurar"
        android:text="@string/Configurar" />

    <Button
        android:id="@+id/button_acerca"
        android:layout_width="132dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:onClick="lanzarAcercaDe"
        android:text="@string/Acercade" />

    <Button
        android:id="@+id/button_salir"
        android:layout_width="132dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:onClick="salirMenu"
        android:text="@string/Salir" />

</LinearLayout>
```

res.layout.activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <Spinner
        android:id="@+id/ListPuntosDestino"
        android:layout_width="match_parent"
        android:layout_height="30dp"
        android:layout_weight="0.09"
        android:prompt="@string/TxtSpinner" />
```

```
<SurfaceView
    android:id="@+id/camerapreview"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="0.71" />

<TableLayout
    android:layout_width="329dp"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_weight="0.02"
    android:gravity="center"
    android:orientation="horizontal"
    android:stretchColumns="0|1" >

    <TableRow>

        <TextView
            android:id="@+id/textSensor"
            android:layout_width="191dp"
            android:layout_height="25dp"
            android:layout_alignParentLeft="true"
            android:layout_marginLeft="10dip"
            android:layout_weight="0.95"
            android:text="Sensores:"
            android:textColor="#FFFF00"
            android:textSize="15dip"
            android:textStyle="bold" />

    </TableRow>
    <TableRow>
        <TextView
            android:id="@+id/textCoord"
            android:layout_width="191dp"
            android:layout_height="30dp"
            android:layout_alignParentLeft="false"
            android:layout_marginLeft="10dip"
            android:layout_weight="0.95"
            android:text="Coodenadas: "
            android:textColor="#FFFF00"
            android:textSize="15dip"
            android:textStyle="bold" />
        <Button
            android:id="@+id/buttonMain"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_marginRight="10dip"
            android:layout_span="1"
            android:layout_weight="0.25"
            android:onClick="salirMain"
            android:text="@string/Salir"
            android:textSize="15dip" />
    </TableRow>

</TableLayout>
</LinearLayout>
```

res.layout.control.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/text1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Destino:"
        android:textColor="#de0000" />

    <FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:orientation="vertical" >

        <ImageView
            android:id="@+id/image_der"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="0.48"
            android:scaleType="center"
            android:src="@drawable/flecha_derecha2"
            android:visibility="invisible" />

        <ImageView
            android:id="@+id/image_rec"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="0.48"
            android:scaleType="center"
            android:src="@drawable/flecha_arriba2"
            android:visibility="invisible" />

        <ImageView
            android:id="@+id/image_izq"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="0.48"
            android:scaleType="center"
            android:src="@drawable/flecha_izquierda2"
            android:visibility="invisible" />

    </FrameLayout>

    <FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:orientation="vertical" >
        <ImageView
```

```

        android:id="@+id/image_des"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="0.48"
        android:scaleType="center"
        android:src="@drawable/par_huella_delantera_acostada"
        android:visibility="invisible" />

    </FrameLayout>
</LinearLayout>

```

res.layout.lista_spinner.xml

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    tools:context=".GuIAR" >

    <Spinner
        android:id="@+id/ListPuntosDestino"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:text="Origen"
        android:textSize="6dp" />

</LinearLayout>

```

es.upm.fi.ls.decoroso.miles.rest.MILESService

```

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;

import es.upm.fi.decoroso.guiar.ruta.Mundo;
import es.upm.fi.decoroso.guiar.ruta.PuntoInteres;
import es.upm.fi.ls.decoroso.cliente.adaptador.conexion.AdaptadorCliente;

@Path("/rest")
public class MILESService {

    private Mundo m;

    private AdaptadorCliente getAdaptador(){
        return AdaptadorCliente.getInstancia( AdaptadorCliente.
                CLIENTE_GRAFICO, "EN", "guia_miles_simple");
    }

    //obtenemos la lista de puntos de interes que tiene MILES
    @GET
    @Path("/puntosinteres")
    @Produces(MediaType.APPLICATION_JSON)
    public HashMap<String, PuntoInteres> getPuntosInteres(){

```

```

        AdaptadorCliente adaptador=getAdaptador();
    try{
        Object[] puntos=adaptador.getListaPuntosInteres();
        adaptador.finalizar();
        return puntos[1];
    }catch(Exception e){
        adaptador.finalizar();
        return e.getMessage();
    }

}

//Obtenemos la trayectoria guia para un destino específico
//La parte comentada corresponde al llamado a MILES el resto hace parte
//del escenario de la experimentación
@GET
@Path("/solicitarguia")
@Produces(MediaType.APPLICATION_JSON)
public Vector<PuntoInteres> getsolicitarGuia(@QueryParam("x") double
    x,@QueryParam("y") double y,@QueryParam("destino") String
    destino){
    //Prueba Sala
    Vector<PuntoInteres> v=new Vector<PuntoInteres>();
    v.add(m.getPuntosInteres().get(destino));
    return v
    ;
    //Determina el punto mas cercano y empieza el camino desde ahí. El
    //camino mas cercano se calcula usando Disktra
    /*Collection<PuntoInteres> pis=m.getPuntosInteres().values();
    PuntoInteres i=new PuntoInteres(x, y, "ubicacion");
    double dist=Double.MAX_VALUE;
    String or="";
    for(PuntoInteres pi:pis){
        if(PuntoInteres.distancia(i, pi)<dist){
            or=pi.getLabel();
            dist=PuntoInteres.distancia(i, pi);
        }
    }
    return m.getRuta(or, destino);
    */
    /*AdaptadorCliente adaptador=getAdaptador();
    try {
        String indicaciones = adaptador.solicitarGuia(destino,
            tipoCliente);
        adaptador.finalizar();
        return indicaciones;
    }
    catch(Exception e){
        adaptador.finalizar();
        return e.getMessage();
    }
    */
}
}

```

Objeto JSON de puntos de interés generado por el sistema MILES

```
{ "Escritorio3": [1850.0, 100.0, 600.0],  
  "Mesa1": [1450.0, 350.0, 0.0],  
  "Escritorio4": [1800.0, 700.0, 600.0],  
  "Pecera1": [1200.0, 230.0, 300.0],  
  "Escritorio1": [1850.0, 100.0, 300.0],  
  "Escritorio2": [1850.0, 700.0, 300.0],  
  "Mesa5": [950.0, 650.0, 300.0],  
  "Mesa4": [550.0, 50.0, 300.0],  
  "Pecera2": [990.0, 490.0, 600.0],  
  "Mesa3": [350.0, 880.0, 0.0],  
  "Mesa2": [1500.0, 850.0, 0.0],  
  "Laboratorio 301": [1240.0, 400.0, 600.0],  
  "Pasillo 102": [340.0, 240.0, 0.0],  
  "Laboratorio 302": [340.0, 740.0, 600.0],  
  "Pasillo 101": [340.0, 740.0, 0.0],  
  "Despacho 201": [1740.0, 740.0, 300.0],  
  "Despacho 202": [340.0, 740.0, 300.0],  
  "Despacho 101": [1240.0, 740.0, 0.0],  
  "Sala WIFI": [1740.0, 240.0, 0.0],  
  "Pasillo 202": [1240.0, 740.0, 300.0],  
  "Mesa6": [250.0, 650.0, 600.0],  
  "Pasillo 201": [340.0, 240.0, 300.0],  
  "Laboratorio 201": [1500.0, 240.0, 300.0],  
  "Cafeteria Profesores": [340.0, 240.0, 600.0],  
  "Laboratorio 101": [1240.0, 240.0, 0.0],  
  "Despacho 301": [1740.0, 240.0, 600.0],  
  "Ventana3": [800.0, 400.0, 300.0],  
  "Ventana4": [300.0, 400.0, 650.0],  
  "Ventana1": [300.0, 400.0, 0.0],  
  "Ventana2": [300.0, 400.0, 300.0],  
  "Despacho 302": [1740.0, 740.0, 600.0],  
  "Cafeteria Alumnos": [340.0, 740.0, 0.0]  
}
```

