Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo Fin de Grado

Aplicación Modelo de Realidad Aumentada

Autor: Javier Palomares Ruiz

Tutor/es: José María Gutiérrez Martínez

2016

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo Fin de Grado
Aplicación Modelo de Realidad Aumentada

Autor: Javier Palomares Ruiz

Tutor/es: José María Gutiérrez Martínez

TRIBUNAL:

Presidente:

Vocal 1º:

Vocal 2º:

Calificación:

FECHA:

Agradecimientos

Este trabajo final de grado está dedicado a todos los grandes compañeros con los que he tenido el placer de toparme en estos años de carrera, como José Andrés, Armando, Francisco Moreno, Álvaro Pastor, Rocío Montalvo (a cuál echamos mucho de menos por aquí), entre otros. ¡Gracias Chicos!

Esencialmente, a mi familia sin la cual no habría sido posible todo lo que he llegado a conseguir, gracias Papa, Mama, por todo, a mis abuelos por todo su cariño y afección y en especial a mis tíos y primos a los cuales tengo un cariño inmenso.

Por último, pero no por ello menos importante, mención honorifica a mi tutor José María Gutiérrez por su apoyo en esos momentos de flaqueza y sus consejos durante el desarrollo de este trabajo.

¡GRACIAS!

Resumen

En la actualidad, las aplicaciones de realidad aumentada se encuentran a la orden del día, estas aplicaciones tienen usos desde educativos a recreativos, incluso para información y turismo, este tipo de aplicaciones tratan de plasmar la información de una manera diferente, utilizando los elementos del móvil.

Por tanto, el objetivo de este proyecto es el desarrollo de una o varias aplicaciones que muestren el uso típico de estas aplicaciones sobre el plano, para ello realizaremos un análisis previo de algunas de las herramientas disponibles para el desarrollo de la misma, implementando de esta manera un Software modelo para un futuro.

Palabras clave

Realidad Aumentada, DroidAR, Android, Modelo, Aplicación móvil.

Abstract

Currently, the augmented reality applications are the order of the day, these applications have uses from educational to recreational, even for information and tourism, these applications try to capture the information in a different way, using the elements of mobile technology.

Therefore, the objective of this project is the development of one or more applications that show the typical use of these applications on the plane, for it we will make a preliminary analysis of some of the tools available for the development of it, implementing this Software way a model for the future.

Keywords

Augmented Reality, DroidAR, Android, Model, Mobile Application.

Resumen extendido

En la actualidad, las aplicaciones de realidad aumentada se encuentran a la orden del día, los campos de aplicación de la realidad aumentada son múltiples, desde el desarrollo de aplicaciones para la enseñanza permitiendo al profesorado hacer más interactiva la clase mediante el uso de modelos 3D que aparecerían en las cámaras móviles del alumnado, hasta su uso en el entretenimiento permitiendo el desarrollo de aplicaciones interactivas basados en modelos de realidad aumentada, su uso se extiende a la mayoría de áreas de conocimiento, por ejemplo, podríamos tener una aplicación de realidad aumentada que detectase los lugares de interés turístico mostrando información relativa al usuario.

El término Realidad Aumentada se usa para referirse a la visión mediante un dispositivo de un entorno físico combinado con uno virtual.

Este trabajo de fin de grado surge bajo la necesidad de disponer de un marco de trabajo para cualquier aplicación de realidad aumentada, de manera que este sirva de modelo para aplicaciones de usos similares, ya que se ha observado que actualmente la realidad aumentada es una tecnología de gran utilidad en el entorno de la telefonía móvil.

Por tanto, el objetivo de este proyecto es el desarrollo de una o varias aplicaciones que muestren el uso típico de estas aplicaciones sobre el plano, para ello realizaremos un análisis previo de algunas de las herramientas disponibles para el desarrollo de la misma, implementando de esta manera un Software modelo para un futuro.

Índice Resumido

Índice

1. Introducción

El siguiente documento tiene la finalidad de especificar los elementos utilizados en el desarrollo de la aplicación realizada por el alumno Javier Palomares Ruiz.

Con este proyecto se pretende demostrar los conocimientos adquiridos durante los cursos de la Ingeniería, permitiendo de esta manera un desarrollo más práctico de los mismos, por otro lado, el proyecto no se ve como una obligación si no como una manera de superación y satisfacción por la experiencia adquirida.

Este trabajo de fin de grado surge bajo la necesidad de disponer de un marco de trabajo para cualquier aplicación de realidad aumentada, de manera que este sirva de modelo para aplicaciones de usos similares, ya que se ha observado que actualmente la realidad aumentada es una tecnología de gran utilidad en el entorno de la telefonía móvil.

Realidad Aumentada

Los campos de aplicación de la realidad aumentada son múltiples, desde el desarrollo de aplicaciones para la enseñanza permitiendo al profesorado hacer más interactiva la clase mediante el uso de modelos 3D que aparecerían en las cámaras móviles del alumnado, hasta su uso en el entretenimiento permitiendo el desarrollo de aplicaciones interactivas basados en modelos de realidad aumentada, su uso se extiende a la mayoría de áreas de conocimiento, por ejemplo, podríamos tener una aplicación de realidad aumentada que detectase los lugares de interés turístico mostrando información relativa al usuario.

El término Realidad Aumentada se usa para referirse a la visión mediante un dispositivo de un entorno físico combinado con uno virtual, ya sea de manera directa o indirecta. Los dispositivos son los encargados de añadir esa información virtual al objeto físico en la realidad sin sustituir en ningún momento al objeto físico ya que no se trata de una realidad virtual si no ampliada, añadiendo información a la realidad**. (Véase** [1]**)**

Actualmente la realidad aumentada se emplea en campos de investigación, es posible mediante imágenes generadas en tiempo real proyectarlas sobre ciertas superficies como la cabeza, o directamente acceder a estas imágenes con una pantalla colocada delante de la retina, creando así una sensación de una realidad ficticia.

El sistema operativo elegido

Por otro lado, la parte programática del proyecto se decidió desde un principio que su desarrollo se haría íntegramente en Android ya que este cumple con las especificaciones del mismo, además se trata de uno de los sistemas más empleados en la actualidad adquiriendo la mayor cantidad de usuarios posible.

Finalmente, la aplicación requería del uso de la geolocalización para funcionar correctamente, ya que esta debe saber en qué posición te encuentras actualmente para asegurar el reconocimiento de los objetos. Este servicio de geolocalización se ofrece es ofrecido por Android de una manera rápida y al alcance de todos, lo que hizo de este sistema el elegido para el desarrollo de la aplicación.

2. Objetivos

Este proyecto fin de grado tiene un objetivo claro:

Desarrollar una aplicación marco para la realidad aumentada en dispositivos móviles, utilizando un procedimiento gradual desarrollando las funcionalidades que se creían convenientes en cada momento.

Principalmente la aplicación se desarrolla con el objetivo de obtener un resultado lo suficientemente eficaz para que esta pueda ser considerada como un modelo para otras aplicaciones del mismo estilo.

Por lo tanto, el proyecto está destinado a un uso posterior por parte de la Universidad o bien para el beneficio propio, ya que se trata de una aplicación marco que podrá ser reutilizado en proyectos de mayor envergadura y por lo tanto supone un reto, ya que contra mejor sea el desarrollo de la misma más facilidades se encontrarán en sus futuros usos.

El desarrollo de esta aplicación permitirá un estándar dentro del campo de aplicación haciendo más ágil la programación de este tipo de aplicaciones, lo que ahorrará gran cantidad de recursos, tanto humanos como económicos en su necesidad.

Por lo tanto, y para que la aplicación sea lo más reutilizable posible será necesario elaborar un manual o una serie de instrucciones que permitan al usuario final de la aplicación utilizar el código de la misma como referencia para el desarrollo de la suya propia.

3. Estado tecnológico

La finalidad del siguiente apartado es introducir en detalle las tecnologías empleadas en el desarrollo del presente trabajo con el fin de familiarizar al usuario en las mismas. En el mismo, se introduce al uso de tecnologías Android, SQLite y los diferentes Kits de desarrollo de realidad aumentada.

3.1. Android

Android fue diseñado como un sistema operativo orientado a móviles y basado en núcleos Linux, el sistema opera principalmente con el uso de pantallas táctiles de smarthphones aunque en la actualidad se utilizan en gran variedad de dispositivos como televisores, automóviles y relojes inteligentes. **(Véase** [2]**)**

El proyecto fue iniciado por Android inc., aunque fue respaldado económicamente por Google por lo que la empresa tomó la mayoría de las decisiones, actualmente la empresa se encuentra en propiedad de Google, que en 2005 la compro definitivamente en sus versiones preliminares, presentando así en 2007 la primera versión definitiva.

A pesar de todos los indicios anteriores no se le tomó demasiada importancia ya que el sistema carecía de teléfonos que los empleasen, no fue hasta 2010 cuando Android se ganó un hueco en la industria como el sistema operativo móvil más utilizado del mercado.

La característica más destacable del sistema se trata de su accesibilidad ya que este está incluido dentro de la normativa OpenSource la cual pone a disposición de cualquier usuario el uso y modificación del producto permitiendo la programación sobre esta plataforma de manera más sencilla, además tanto para desarrolladores como para fabricantes el precio de adquisición del producto final es bajo.

El sistema permite por lo tanto el desarrollo de aplicaciones con código similar al utilizado por Java del que hablaremos posteriormente, además el propio sistema permite el acceso a las funciones primarias del teléfono como pueden ser la agenda, la cámara, las llamadas o la brújula, estas facilidades hacen que la en la actualidad existan multitud de aplicaciones basadas en este sistema operativo marca de Google.

Cualquiera de las características del aparato electrónico que lo implementa son altamente personalizables para adaptarse al usuario, utilizar el teléfono como cámara de fotos, como sistema de navegación, utilizar internet en cualquier parte, comprobar las cuentas bancarias, envías mensajes de manera gratuita, llamadas virtuales sin coste, personalizar la pantalla como si de un ordenador se tratase, es exactamente lo que pretende la empresa, llevar un ordenador en miniatura en tu bolsillo para cualquier consulta.

Android está destinado a almacenar toda la información personal posible y para ello cada teléfono móvil debe tener una cuenta de la compañía (Google) para poder acceder a todas las aplicaciones de la misma entre las que se incluye la tienda donde se pueden adquirir todas las de terceras partes, de esta manera el sistema almacenará en la cuenta los datos de contactos, fechas de calendario, sitios web visitados e información de configuración del teléfono.

Estructura Android

El núcleo Android está basado en la arquitectura Linux, proporcionando de esta manera servicios como seguridad, acceso a memoria, multiproceso o drivers de dispositivos, es una capa de abstracción dependiente por tanto del software.

Todas las capas subyacentes se comunican a través de este núcleo, para el correcto funcionamiento del sistema se utilizan entornos de ejecución en lenguaje Java (Dalvik), implementando además ciertas librerías de lenguajes externos como C o C++, entre estas librerías se encuentra SQLite un motor de bases de datos muy utilizado.

Por último, en las capas más altas se encuentran los entornos de la aplicación que son la plataforma de acceso a las diferentes características del sistema que permiten la elaboración de aplicaciones de desarrollo libre, se diseñó específicamente para la reutilización de componentes, beneficiándose así unas aplicaciones de las características de otras.

¿Qué tipo de aplicaciones admite Android?

Existen multitud de aplicaciones y utilidades para el sistema Android, desde aplicaciones para la personalización de la apariencia del teléfono hasta la realización de fotografía artística con aplicaciones de cámara, hasta reproductores de música, aunque estas son las más usuales existen otras que emplean los servicios ofrecidos por Android (Cámara, GPS, brújula, agenda) para elaborar aplicaciones complejas como geolocalización o realidad aumentada.

Por otro lado, existen otro tipo de aplicaciones más complejas denominadas widgets que se anclan a las diferentes pantallas virtuales (escritorios) permitiendo la personalización de las mismas, entre estos se encuentran los predictores de temporal, el calendario, los reproductores de música, las galerías de fotos y algunas aplicaciones como Youtube u otras de terceros que permiten el uso de los mismos.

Además, el mercado de Android está creciendo en materia de videojuegos ya que también se pueden realizar pequeñas obras digitales para esta plataforma móvil, es decir, también existen aplicaciones de entretenimiento, permitiendo así al usuario utilizar su propio dispositivo móvil como una forma de diversión.

Finalmente, y para adaptarse al mercado actual se ha dividido el espectro actual de Android en dos vertientes una disponible para la telefonía y similares y otra para los relojes inteligentes de la compañía, este tipo de aplicaciones son específicas y se tienen que desarrollar para Android Wear, muchas de las aplicaciones actuales no son compatibles con estos nuevos dispositivos (Android Watch).

Actualizaciones de Android

Desde su lanzamiento Android ha ido actualizando los elementos disponibles en cada una de sus versiones desde la más básica (Android 1.0 Apple Cake) en septiembre de 2008 cuyo contenido era bastante escaso, aunque muy novedoso para la época hasta el actual lanzado en octubre del año pasado (Android 6.0 Marshmallow), cuya novedad más interesante es el soporte para huellas dactilares que no todos los dispositivos incluían. **(Véase** [3]**)**

Estas versiones del sistema operativo siguen la normativa OpenSource y por tanto los fabricantes deciden implementar sus propias versiones del sistema operativo, de esta manera se hace posible la inclusión de funcionalidades externas a la versión del sistema operativo, como el reconocimiento facial o la detección de voz.

Versionado Android	
Nombre del sistema	**Fecha de lanzamiento**
A: Apple Pie (1.0)	**22 de octubre de 2008**
B: Banana Bread (1.1)	**9 de febrero de 2009**
C: Cupcake (1.5)	**30 de abril de 2009**
D: Donut (v1.6)	**15 de septiembre de 2009**
E: Éclair (v2.0/v2.1)	**26 de octubre de 2009**
F: Froyo (v2.2)	**20 de mayo de 2010**
G: Gingerbread (v2.3)	**6 de diciembre de 2010**
H: Honeycomb (v3.0/v3.1/v3.2)	**22 de febrero de 2011**
I: Ice Cream Sandwich (v4.0)	**19 de octubre de 2011**
J: Jelly Bean (v4.1/v4.2/v4.3)	**27 de junio de 2012**
K: KitKat (v4.4)	**31 de octubre de 2013**
L: Lollipop (v5.0/v5.1)	**12 de noviembre de 2014**
M: Marshmallow (v6.0)	**28 de mayo de 2015**

Estructura fundamental de una aplicación

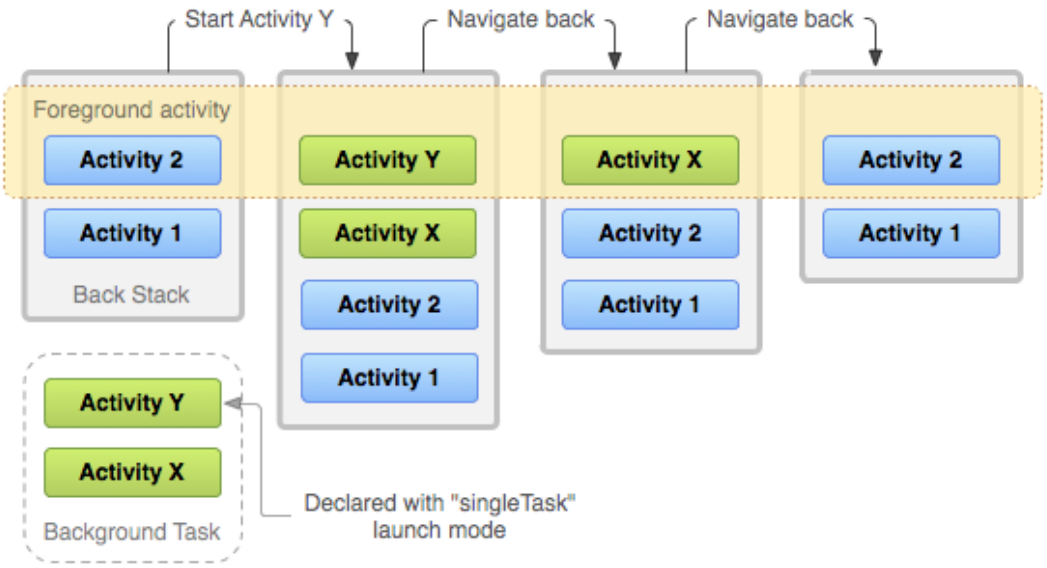
Antes de empezar con los conceptos técnicos debemos ponernos en contexto, saber cómo funciona internamente una aplicación Android, cuáles son sus componentes principales, entre estos podemos destacar los siguientes **(Véase** [4]**):**

Activity (Actividades)

Se trata del componente principal de la aplicación, permite definir las acciones que se van a realizar en cada una de las pantallas que se presentarán, se podría decir que es el elemento análogo a las vistas (ventanas).

Generalmente el concepto de actividad se extiende a más de una ya que es muy probable que a cada vista le corresponda una ventana, siempre teniendo en cuenta que debe haber una actividad principal que deberá ser especificada como tal.

El intercambio de ventanas que se realiza mediante Intent (de los que hablaremos a posteriori) se almacenan en una pila, cada actividad por la que pasamos se detiene a la espera de volver a ser llamada, por ello, esta pila se denomina "pila de retroceso". **(Véase** [5]**)**

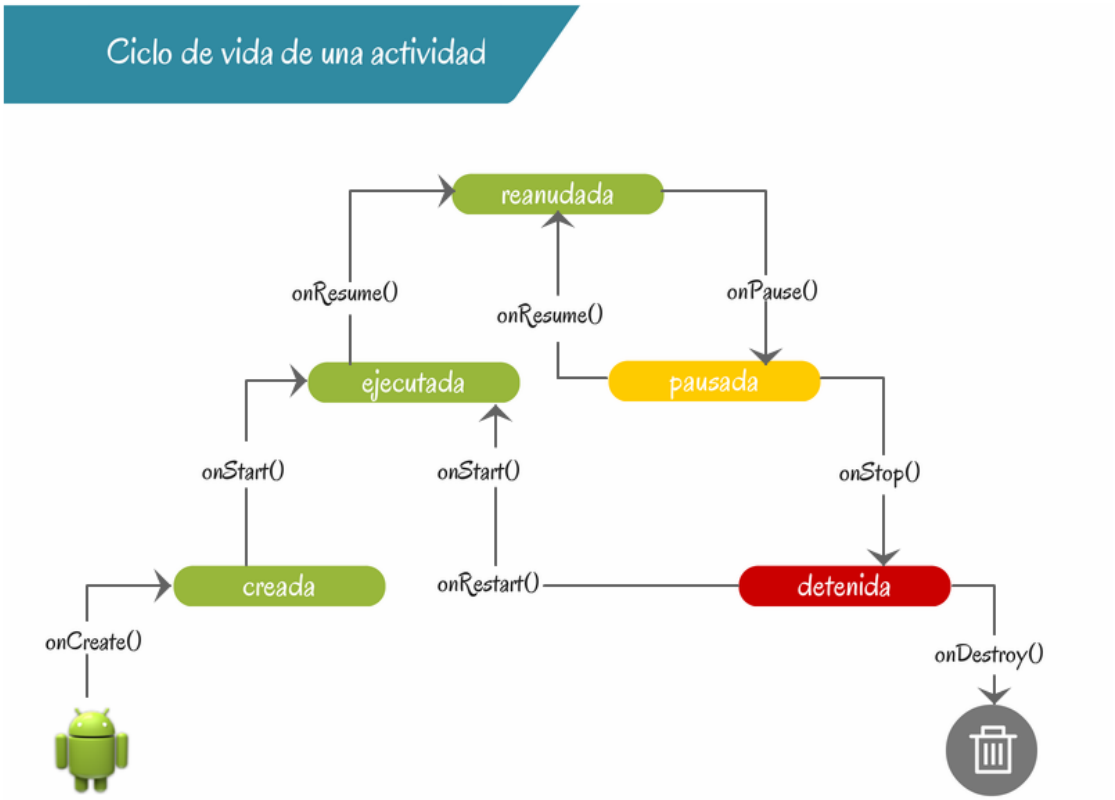


A cada actividad le corresponde un estado que se relaciona con dicha pila **(Véase** [6]**)**:

- **Running (Activo):** La actividad no ha sido llevada a un segundo plano, se trata de la actividad que se está mostrando actualmente, está en la cima de la pila.
- **Paused (Pausado):** La actividad está en segundo plano, otra actividad está por encima de ella, si otra actividad más es llevada al primer plano esta se oculta por completo pasando al siguiente estado.
- **Stopped (Parada):** La aplicación no está mostrando dicha pantalla, se debe guardar su estado para recuperarla posteriormente.
- **Destroyed (Destruida):** La actividad desparece definitivamente, ha sido desechada y no es recuperable ya que no está en la pila correspondiente.

Por este motivo cada actividad puede definir las acciones que se realizan en caso de producirse uno de los eventos anteriores mediante las siguientes funciones:

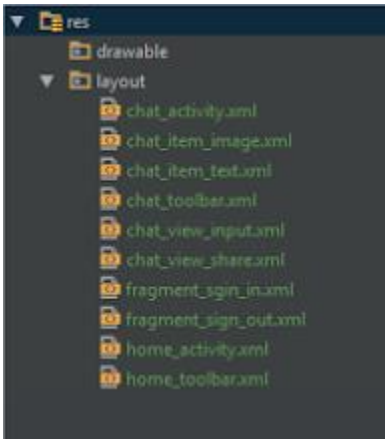
- **onCreate()**: Se invoca una vez por cada actividad, siempre al iniciarla.
- **onRestart():** Se invoca si la actividad vuelve al primer plano.
- **onStart():** Se invoca cuando la actividad se inicia (se hace visible al usuario).
- **onResume()**: Siempre que el usuario vuelve a interactuar con la vista.
- **onPause():** Se invoca cuando otra actividad se pone delante de la actual.
- **onStop():** Se invoca siempre que la actividad pasa a no está visible.
- **onDestroy():** Se produce cuando la actividad está finalizando, bien porque el método onFinish() ha sido llamada o bien para liberar espacio en memoria.



Podemos ver en el siguiente diagrama el ciclo de vida de una actividad, cada estado está representando de un color, los verdes son procesos por los que pasa una actividad que se encuentra en ejecución o va a ser ejecutada, por otro lado, amarillo o rojo identifican que se encuentra inactiva, bien porque ha sido destruida o bien porque esta pausada.

Views (Vistas)

Son el elemento fundamental junto con las actividades, definen toda la parte visual de la aplicación, su estructura se divide en forma de árbol pudiendo agrupar tantos componentes (botones, cajas de texto, calendarios selectores) como se quieran en un contenedor.



Intent

El intent permite la comunicación entre dos actividades de una aplicación Android, se trata del envío de mensajes o peticiones entre los distintos componentes del sistema, se asemejan a los enlaces en páginas web, permiten el intercambio de información por paquete (Bundles).

Desde un Intent se puede iniciar una aplicación, un servicio (como la localización), enviar mensajes o incluso interconectar aplicaciones. Un Intent se compone de:

- **Una acción:** Que deseamos hacer de entre las descritas arriba.
- **Envío de datos:** Que datos queremos que la otra parte (actividad-servicio) reciba.

3.2. SQLite

SQLite será el motor de base de datos que se utilizará durante el proyecto, esto nos permitirá almacenar la información relativa a los usuarios que utilicen la aplicación de manera sencilla, se trata de una base de datos relacional de código abierto. **(Véase** [7]**)**

La principal ventaja de este motor es que precisa de poca configuración para funcionar y no requiere de un servidor ya que utiliza archivos almacenados internamente en disco, aunque parezca lo contrario dichos archivos ocupan muy poco y son muy fáciles de gestionar.

Además este motor de base de datos dispone de un paquete nativo en Android que nos facilitará su desarrollo, Android ofrece de serie soporte total para la creación y administración de esta base de datos a través del paquete "android.database.sqlite", aunque sí que deberemos crear y gestionar los datos que esta se almacenen.

A diferencia de los demás sistemas gestores de bases de datos SQLite no es un proceso independiente del programa o aplicación que lo ejecuta, esto nos da cierta velocidad a la hora de manipular los datos, la biblioteca SQLite está directamente enlazada no es necesario llamar a servicios externos que introduzcan un retardo en las consultas, todo programa que utiliza SQLite realiza llamadas a procedimientos y subrutinas.

En sus últimas versiones SQLite permite almacenamiento de datos de hasta 2 Terabytes, además debido a su pequeño tamaño, SQLite es muy adecuado para los sistemas integrados, como pueden ser los de los dispositivos móviles, en este caso, Android. Dichos sistemas pueden hacer uso de la base de datos en lectura desde tantos hilos como se desee, mientras que en el caso de la escritura solo uno puede acceder a la vez a una misma columna.

Respecto a la gestión que realiza de los datos su sistema se basa en transacciones ACID (Atomicidad, consistencia, aislamiento y durabilidad), usando el estándar SQL, por otro lado, en lugar de asignar un tipo a una columna SQLite lo asigna directamente a cada valor, lo que permite tener columnas con valores de Enteros rellenas con valores Cadena, ya que SQLite tratara de acomodarlo a su formato, si bien esto puede ser una desventaja también se puede ver como una innovación.

Aunque parezca lo contrario, SQLite es un sistema muy ligera empleado por empresas conocidas para el almacenamiento de datos de sus usuarios, por ejemplo, Skype almacena los datos relativos a las conversaciones en archivos de este tipo, aunque también podemos hablar de navegadores como Mozilla Firefox u OpenOffice.

3.3. Kits de desarrollo de realidad aumentada

Como elemento fundamental para el desarrollo de una aplicación de este tipo, será necesario disponer de servicios de posicionamiento o reconocimiento de objetos entre otros.
(Vease [8]**)**

Se han valorado distintos kits de desarrollo que serán expuestos a continuación, todos ellos disponen de estos servicios comentados, pero no todos se adaptan a las necesidades de este proyecto, lo que más se ha valorado es que este posicionamiento sea lo más preciso y sencillo posible, además la cámara deberá ser capaz de reconocer los objetos que deseemos.

Wikitude

Wikitude se trata de un kit para desarrolladores que incluye todas las funcionalidades necesarias para una aplicación de realidad aumentada, como son el reconocimiento y seguimiento de elementos 3D, superposición de videos o localización basada en Realidad aumentada.

Para la realidad aumentada basada en la localización, la posición de los objetos en la pantalla del dispositivo móvil se calcula utilizando la posición del usuario (mediante GPS o Wi-Fi), la dirección en la que se enfrenta el usuario (mediante el uso de la brújula) y el acelerómetro.

Desde agosto de 2012, Wikitude también cuenta con tecnologías de reconocimiento de imagen que permiten imágenes de seguimiento para activar la tecnología de realidad aumentada dentro de la aplicación.

El contenido en el Wikitude World Browser es principalmente generado por el usuario. El contenido puede ser añadido por una interfaz web, por KML, y ARML. Además, los servicios Web están disponibles para registrar la entrega de datos dinámicos. Wikitude es un miembro del W3C y OGC y está trabajando para desarrollar aún más ARML como parte de un proyecto W3C ARML

Los servicios que ofrece Wikitude son gratuitos siempre y cuando la marca de agua de la aplicación aparezca sobre la cámara, también dispone de versiones comerciales por las que habría que pagar.

DroidAR

DroidAR se trata de un framework de desarrollo de realidad aumentada exclusivo para Android, permite localizar al usuario y marcar su posición mediante estos mecanismos entre otros servicios que ofrece.

Dicho framework de desarrollo se puede utilizar en muchos escenarios diferentes, la mayoría de ellos se encuentran representados en las demos que ofrece la página, este framework se puede descargar gratuitamente bajo una licencia GNU GPL desde un repositorio Github o bien desde la propia página de los desarrolladores.

La realidad aumentada basa en localización y la misma basada marcador son posibles. Este marco se puede utilizar para muchos escenarios diferentes, ya existen algunas aplicaciones de demostración disponibles.

- Reconocimiento de pasos, Realidad aumentada en interiores.
- Juegos con realidad aumentada y proyecto crowfunding.

Desde la creación inicial del prototipo de una herramienta de desarrollo de la realidad aumentada en 2010, se ha mejorado de forma continua y se añadieron nuevas características. La herramienta de desarrollo creada (SDK) permite a otros programadores integrar la realidad aumentada en sus propias aplicaciones.

Ellos proporcionan el SDK de forma gratuita con una licencia de código abierto para proyectos no comerciales. Se ha desarrollado desde entonces en uno de los más utilizados de origen aumentada SDK de realidad abierta para la plataforma Android.

Este software es lo suficientemente flexible como para ofrecer las posibilidades de una aplicación de realidad aumentada desde lo más básico a los más complejo.

Mixare

Mixare (Mix Augmented Reality) se trata de un navegador de realidad aumentada publicado bajo licencia GPLv3, dispone de las características fundamentales y sirve tanto para dispositivos Android como iOS.

Se trata de un kit para desarrolladores muy versátil, solo tiene un inconveniente y este es de los más importantes ya que la documentación es un elemento fundamental para los desarrolladores, cosa de la que mixare carece, lo que dificultaría su desarrollo posterior.

Podemos elegir Mixare como nuestro SDK de desarrollo dados los siguientes puntos:

1. Mixare es una aplicación autónoma, que (por el momento) muestra puntos de interés de los alrededores mediante Wikipedia.
2. Mixare se puede acceder mediante un enlace en una página HTML, por lo que la fuente de datos se transfiere a la aplicación.
3. Mixare se puede acceder por el propio lanzador de la aplicación, la fuente de datos se transfiere a la aplicación.
4. Mixare es extensible libremente e incluso se puede modificar en una aplicación individual (GPLv3).

Layar

Este parece el más versátil de todos, pero es de pago, y no dispone de versión gratuita, Layar permite a los desarrolladores una amplia gama a de posibilidades para publicar capas de realidad aumentada, este kit de desarrollo emplea REST (Representational State Transfer) para facilitar la integración de todas las aplicaciones que la utilicen.

El navegador hace uso de los siguientes:

- Acelerómetro
- Cámara integrada
- Brújula
- GPS

Estos se utilizan en conjunto para identificar la ubicación del usuario y el campo de visión. Desde la posición geográfica, las diversas formas de datos se colocan sobre la vista de la cámara como la inserción de una capa adicional.

Los datos en el navegador vienen en la forma de capas. Las capas son servicios web REST que sirven puntos geolocalizados de interés en el entorno del usuario. Las capas son desarrolladas y mantenidas por terceros mediante una API libre, Layar como empresa es responsable de la validación en el proceso de publicación. Proyectos de terceros que utilizan Layar pueden ser programas que requieren etiquetas QR, o juegos, tales como Far-Play.

ARToolKit

ARToolkit se trata de una librería GNU open Source que se encontraba disponible para sistemas operativos iOS pero que actualmente ha sido migrada también para dispositivos móviles Android, al tratarse de una actualización reciente el soporte que se ofrece para este último sistema es muy limitado.

ARToolKit es una biblioteca que permite la creación de aplicaciones de realidad aumentada, en las que se superponen imágenes virtuales al mundo real. Para ello, utiliza las capacidades de seguimiento de vídeo, con el fin de calcular, en tiempo real, la posición de la cámara y la orientación relativa a la posición de los marcadores físicos. Una vez que la posición de la cámara real se sabe, la cámara virtual se puede colocar en el mismo punto y modelos 3d son sobrepuestos exactamente sobre el marcador real.

Actualmente se mantiene como un proyecto de código abierto alojado en SourceForge con licencias comerciales disponibles en ARToolWorks. ARToolKit es un AR muy utilizado seguimiento de la biblioteca con más de 160.000 descargas desde el año 2004.

Como inconveniente, los foros de soporte de este kit de desarrollo están llenos de dudas respecto a la implementación y problemas que surgen utilizando ciertas clases que se encuentran en estado deprecado.

Tabla comparativa

Previo a la elección de cualquiera de los kits de desarrollo arriba expuestos se decide realizar una comparativa de cada uno de ellos para ver hasta qué punto nos pueden ser de utilidad en el desarrollo del proyecto.

Gracias a **Danie Jooste, Victoria Rautenbach y Serena Coetzee** por su evaluación de los diferentes SDK expuesta en la página [9].

Esta evaluación se realizará atendiendo a los criterios a continuación expuestos y que nos permitirán realizar una tabla comparativa para cada uno de los kits, se ha optado por dividir estos criterios en diferentes secciones de las que constará este apartado y son las siguientes:

Criterios generales (Lenguaje de programación, plataforma), criterios funcionales (Tipo de datos, formato de los mismos), por último, criterios no funcionales (Usabilidad).

Criterios generales

		MixAR	ARToolKit	DroidAR	Layar	Wikitude
Plataforma	Android	X	X	X	X	X
	iOS	X	X	X	X	X
	BlackBerry				X	
	Otros					X
Lenguaje	Java	X	X	X	X	X
	Objetive-C		X		X	X
	Otros				X	X
Licencia	OpenSource	X	X	X		
	Freeware				X	X
	Propietario				X	X
Estándar de implementación	OGC ARLM				X	X
	OGB Web Services					
	Codificado Estándar	X	X	X	X	X
Disponibilidad Offline	Si/No	NO	SI	SI	NO	NO

Estándares de implementación: OGC (Open Geospatial Consortium) ARLM se trata de un estándar exclusivo de realidad aumentada que permite utilizar sistemas XML para describir y posicionar objetos dentro del mapa empleando un sistema específico ECMA que permite acceder a los datos de un objeto en concreto, por otro lado, el estándar OGB aunque no se implementa en ninguna habría sido útil debido a su versatilidad con sistemas JSON y conexiones a bases de datos.

Disponibilidad Offline: Es una de las características más demandadas ya que muchos usuarios no suelen disponer de altas cantidades de datos de red, lo que supone un problema para el uso de este tipo de aplicaciones que tienen que estar constantemente conectadas a internet. ARToolkit y DroidAR son sistemas openSource por lo que podemos implementar las cargas de datos directamente desde el dispositivo sin disponer de conectividad a internet (SQLite).

Criterios funcionales

		MixAR	ARToolKit	DroidAR	Layar	Wikitude
Formato de datos	Web Service	X	X	X	X*	X*
	QR					
	Base de datos	X	X	X		X
	Código Nativo		X	X		X
	Otros					
Presentación de los datos	¿La visualización es alterable?		X	X	X	X
	¿El usuario puede elegir que datos se le muestran?	X	X	X	X	X
Licencia	OpenSource	X	X	X		
	Freeware				X	X
	Propietario				X	X
Eventos en objetos	¿Se pueden disparar eventos en objetos?		X	X	X	X
¿El radio de visualización es alterable?	Si/No	SI	SI	SI	SI	SI
Amplitud visual	Photo				X	X
	LiDAR				X	X
	Otros (3D)	X	X	X	X	X

* *Uso de Web Services propietarios*

Data Source: En todos los SDK se emplean Web Services como sistema nativo de carga de datos, las otras formas son carga de datos mediante bases de datos o código nativo, Layar accede a los datos mediante un Web Service propietario, funcionando como caja negra.

Todos los SDK exceptuando Layar permite a los usuarios utilizar código nativo para acceder a los datos, sin embargo, esto en el software propietario no es posible.

Presentación de los datos: La superposición de información por realidad aumentada se puede dar por muchos medios (texto, imágenes, video).

Para ser más exactos, la presentación de texto es fundamental debido al formato de las bases de datos, esto se puede lograr con todos los frameworks arriba expuestos, sin embargo, existen limitaciones sobre el software propietario.

ARToolkit se basa en la UIKit para ajustar la representación visual de los objetos. Todos los marcos implementan comportamientos en objetos en forma de eventos o disparadores.

Radio de aplicación: Se trata del campo visual del SDK a la hora de utilizar la cámara, todos los frameworks evaluados disponen de esta funcionalidad, sin embargo, si el radio es muy pequeño se puede sobre poblar la zona que la cámara ve con puntos de interés.

Amplitud Visual: Permite no solo confiar en la proximidad al objeto sino también en la definición de ciertas pautas como una imagen significativa o una representación 3D.

DroidAR no es compatible con el reconocimiento de imágenes (es decir, fotos) o LiDAR, pero es compatible con el reconocimiento de gestos que se puede utilizar para desarrollar aplicaciones de realidad virtual para complementar las aplicaciones de realidad aumentada.

Por otro lado, Wikitude ofrece todo tipo de posibilidades para identificar objetos, desde LiDAR o imagen a reconocimiento por Google Glasses.

Criterios no funcionales

		MixAR	ARToolKit	DroidAR	Layar	Wikitude
¿Fácil integración con otras Apps GIS?	¿Incorporados otros framework de GIS?					
Facilidad de extensión	¿El software es extensible?		X	X		
Usabilidad	¿El framework es fácil de instalar?		X		X	X
	¿En general, es fácil de utilizar?	X	X	X	X	X
Documentación y Soporte	Documentación clara y al día	X	X	X	X	X
	Buena comunicación con la comunidad y el soporte	X	X	X	X	X
	Atención al usuario				X	X

Integración con apps GIS: En la actualidad, ninguno de los marcos ha incorporado la integración con otros productos, tales como ArcGIS, QuantumGIS o PostGIS. Aunque esto no es esencial, la integración de estos podría ser interesante a la hora de posicionar los objetos por medio de coordenadas especiales.

Facilidad de extensión: En general no es fácil de ampliar los SDK de propietarios, ya que el código no está disponible y restricciones de licencia prohíbe al usuario de la ampliación del marco. ARToolkit y DroidAR son las únicas aplicaciones que pueden hacerlo fácilmente, ya que su código es completamente abierto.

Facilidad de uso: Se encontró que todos los marcos eran fáciles de utilizar. Todos ellos o bien utilizan el Android Studio con bibliotecas adicionales o un SDK independiente que podría ser instalado usando un instalador de un solo clic.

Documentación y soporte: Todos los marcos, excepto DroidAR, proporcionan una variedad de vías de apoyo. Proporcionan una amplia y bien estructurada y actualizada documentación con numerosos ejemplos y fragmentos de código.

Discusión de los resultados

El objetivo es identificar y evaluar los marcos de desarrollo existentes que podrían utilizarse para el desarrollo de una aplicación móvil que muestra los objetivos en la realidad aumentada.

Por el momento, sólo dos marcos de código abierto (ARToolkit y DroidAR) satisfacen los requisitos relativos a las necesidades. Otros marcos de código abierto, como Mixare, no cumplen con los requisitos.

Un marco de código abierto también permitirá a los usuarios integrar más fácilmente la aplicación con otros productos, que es precisamente lo que se pretende con este trabajo.

La disponibilidad de los datos y aplicaciones de manera offline, es crítica cuando se trabaja en un área rural o de una situación de socorro donde la conectividad es limitada.

Una limitación actual de todos los marcos de propiedad es extensa programación adicional requerida para eludir el método de adquisición de datos.

Atendiendo a los criterios anteriormente expuestos se ha considerado desarrollar la aplicación con el framework **DroidAR** ya que se trata de un kit exclusivo para Android con las funcionalidades cuidadas para este tipo de dispositivos, además ofrece la posibilidad de no disponer de conexión a internet para realizar las operaciones.

Los resultados de la evaluación muestran que todas los SDK seleccionados podrían ser utilizados para desarrollar una aplicación de realidad aumentada, sin embargo, cada uno tiene sus propias limitaciones, siempre teniendo en cuenta la más importante, la del GPS y sus limitaciones en cuanto a precisión (aproximadamente 5 metros horizontales).

4. Análisis y diseño

En este apartado de tratar de definir con claridad los objetivos expuestos en apartados anteriores en un lenguaje más específico, por lo tanto, en el mismo de presentaran los elementos para proceder con el desarrollo de las aplicaciones descritas.

Previamente se ha realizado una toma de decisiones en cuanto al material y elementos necesarios para completar el trabajo, expuesto en el Estado del arte, a partir de este punto se realizará un análisis del diseño de las aplicaciones, organización, técnicas y metodología que se van a emplear (para el desarrollo y la ejecución del proyecto).

Para realizar dicho análisis tomaremos los requisitos de las aplicaciones y mostraremos un pequeño diagrama explicativo para cada una de ellas, por último, de dichos requisitos se extraerá una interpretación para el desarrollo definitivo.

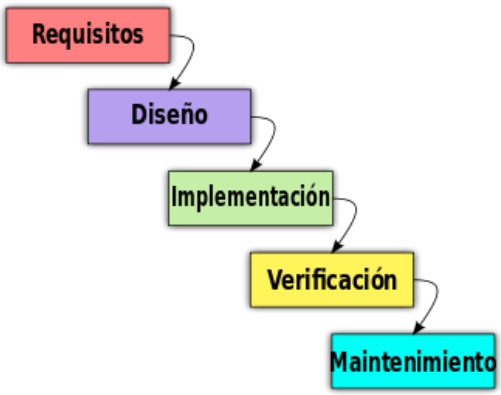
4.1. Metodología

La metodología de trabajo será progresiva, el primer paso y fundamental es la descripción de los requisitos que van caracterizar la aplicación y a partir de los cuales va a ser desarrollada, estos requisitos fueron desarrollados de acuerdo a las necesidades de este tipo de aplicaciones para cada uno de los modelos desarrollados y se detallaran en el apartado correspondiente.

"La metodología de desarrollo se trata de un marco de trabajo para estructurar, planificar, y controlar el proceso de desarrollo y mantenimiento del software"

Tras establecer estos primeros requisitos se procedió a establecer una serie de entregables en fases a modo de prototipos para evaluar la viabilidad de los mismos, siempre de acuerdo al diseño especificado en la fase correspondiente.

Por lo tanto, y con la idea clara de que el primer y fundamental paso es la definición de requisitos el ciclo de vida software que más se acoge a estos parámetros es el de cascada que recoge las fases que a continuación se detallan **(Véase** [10]**):**



La metodología en cascada es un método secuencial de desarrollo en el que los pasos son vistos a través de las fases de *análisis de requisitos, diseño e implementación, verificación y mantenimiento*. Para que se dé el inicio de la siguiente en este proceso se debe haber aprobado la anterior, lo que conlleva que tras la finalización de la etapa de desarrollo se debe verificar que el software está listo para continuar dicho ciclo.

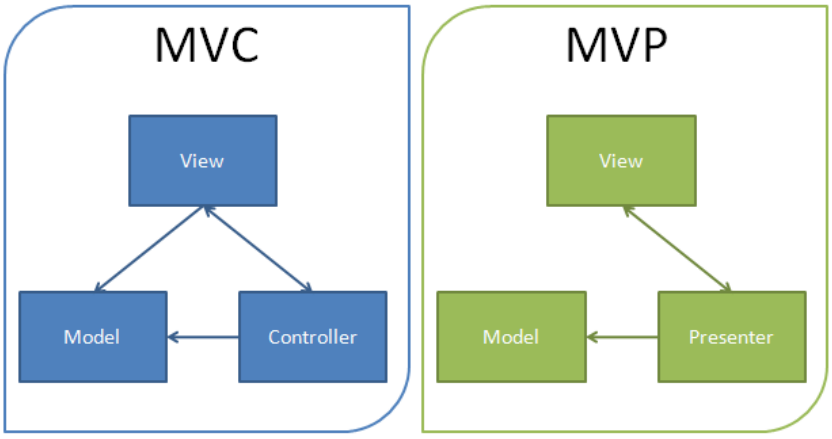
Durante la fase de toma de requerimientos se determinan todas las funcionalidades del sistema intentando cubrir todas las necesidades posibles, del cual se extrae un documento de requisitos, el cual se especificará posteriormente.

Las dos fases posteriores van de la mano se tratan del diseño y la implementación, durante el diseño se organizan los elementos del sistema, de la cual deberemos extraer la especificación de las partes del sistema, existen dos diseños el de alto nivel y el detallado. Para la implementación deberemos codificar todos los elementos extraídos del diseño mediante código fuente, en este caso Java (Android), especificando los algoritmos necesarios para llegar a la solución del problema especificado en los requisitos.

El último paso es realizar las pruebas oportunas sobre el software para asegurarse que los elementos descritos están correctamente codificados y el sistema funciona como debería. Si se comete algún error durante el desarrollo de la aplicación este debe ser notificado en la etapa de verificación lo que implica una revisión del software previa a su venta o distribución, esta metodología es idónea para proyectos de este tipo.

4.2. Técnicas de desarrollo y organización

Cada proyecto debe definir claramente las técnicas de desarrollo que se van a utilizar, cada recurso emplea técnicas acordes al mismo, si bien siempre hay que tener en cuenta dos estándares y que en este caso deben ser empleados sin ninguna duda y estos son la *reusabilidad del código y la facilidad de integración* sin que afecte de manera directa a otros módulos ya integrados, ya que la aplicación se trata de un modelo.



Para facilitar esta laborar se van a emplear en medida de lo posible los patrones de diseño a los que podremos asociar las clases de nuestras aplicaciones, los patrones no son más que estructuras y técnicas de desarrollo para garantizar estos dos elementos fundamentales anteriormente mencionados (reusabilidad y facilidad de integración). Especialmente Android ofrece esta posibilidad internamente en alguna de sus clases y aprovecharemos dicha ventaja.

En cuanto a la estructuración y organización del proyecto vamos a basarnos en un patrón **MVP** (**M**odelo **V**ista-**P**resentador), se trata de una evolución del patrón modelo vista controlador que permite el desarrollo de interfaces (presentadores), ya que al ser nuestra aplicación un modelo lo visual es lo que debería destacar a modo de ejemplo. **(Véase** [11]**)**

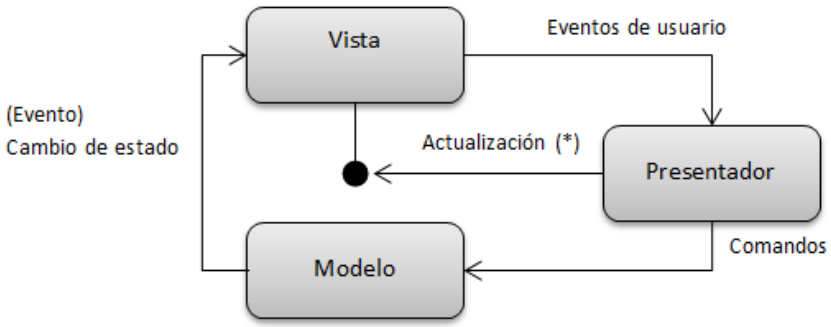
El Modelo Vista-Presentador es un patrón arquitectónico de interfaces de usuario diseño para facilitar pruebas de unidad automatizadas y mejorar la separación en la lógica. **El modelo** se trata de una interfaz que define los datos que se mostrarán, **el presentador** recupera los datos de los repositorios (directamente del teléfono en nuestro caso) y los formatea para presentarlos, por último, **la vista** es la interfaz pasiva que presenta datos relativos al modelo y da órdenes al usuario mediante eventos.

En términos generalmente este patrón utiliza la vista de implementación para instanciar objetos de presentador que se desean utilizar en cada momento, aunque el presentador no establezca claramente un grado de control sobre la vista, ya que dependiendo de este grado de control existen variaciones en el modelo.

A grandes rasgos existen pues dos variantes de este modelo uno supervisado (El cual utilizaremos nosotros) y otro pasivo **(Véase** [12]**):**

MVP Supervisado

En el primero de estos el presentador no gestiona la información que se muestra, sino que es la vista la que define la lógica y el formateo de los datos a partir de los controladores. En estos casos, el presentador únicamente actúa en las acciones más complejas para facilitar el trabajo a la vista en ese momento.

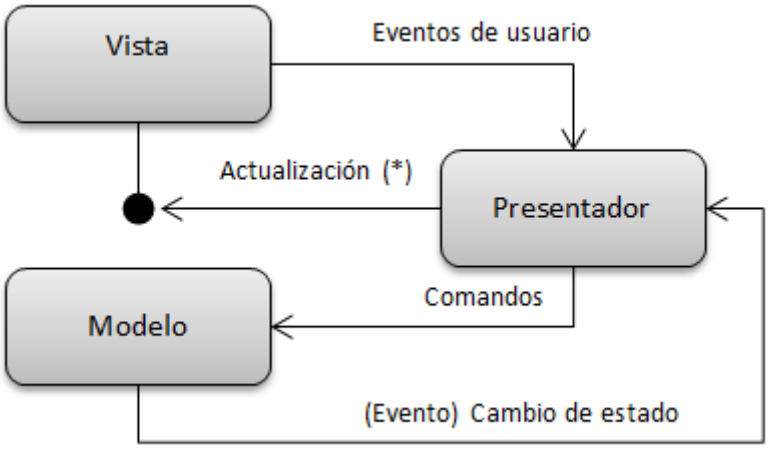


Cuando la vista recibe un evento de teclado (pulsar en pantalla en nuestro caso), delega el control al presentador, en ese momento este realiza operaciones relacionadas con las vistas, como el control de estado en controladores y a posteriori llamar al comando que corresponda, comunicándose con el modelo que corresponda, entonces el modelo realiza cambio de estado que generan eventos sobre la vista.

Con esta lógica conseguimos un modelo similar al modelo vista-controlador, aunque este enfoque permite utilizar técnicas avanzadas de enlazado de datos (Data Binding).

MVP pasivo

Aunque no será el modelo que empleemos conviene tener una referencia de este para comentar las diferencias que existen, en este modelo el presentador gestiona totalmente la información que la vista presenta. Es decir, la vista solo gestiona los datos que le llegan externamente y los presenta (pasiva).



En este caso cuando el usuario realiza una operación sobre la vista (interfaz de usuario), la vista delega todo el trabajo en el presentador, cuando los datos le son devueltos se encarga de cargarlos y presentarlos. El presentador sigue haciendo la función de comunicador con el modelo utilizando los comandos correspondientes, en ese momento el modelo envía un evento de cambio de datos si es necesario y el presentador se lo comunica a la vista para su cambio.

4.3. Requerimientos SW y HW

En el apartado del estado del arte se han seleccionado las tecnologías que se van a emplear en el desarrollo de las aplicaciones, en esta sección vamos a desglosar los elementos software y hardware que han sido necesarios para el desarrollo de la aplicación.

4.3.1. Software

En este apartado se van a desglosar todos los requerimientos en cuanto al software de desarrollo, entre los cuales se encuentran el entorno de desarrollo, los SDK y el sistema operativo empleado.

Entorno de desarrollo

Se barajaron dos IDEs a escoger, entre Android Studio y Eclipse, el segundo de ellos se escogió como alternativa a Android Studio ya que las dependencias y bibliotecas que íbamos a usar estaban desarrolladas en su completitud para eclipse.

Se comenzó a desarrollar la aplicación en Eclipse, aunque no se veía viable para su migración a Android Studio debido al uso de las dependencias de la librería escogida, además de que Google informó el año pasado que dejaría de dar soporte a Eclipse para el desarrollo de aplicaciones móviles, por lo que Eclipse pasó a ser una alternativa inviable.

Por otro lado, tenemos en entorno de desarrollo por excelencia (Android Studio) el cual es el que definitivamente usamos, este maneja las dependencias mediante tecnologías gradle que van actualizándose a medida que pasa el tiempo, si bien, nuestro SDK de desarrollo no utilizaba este tipo de tecnologías por lo que tuvimos que añadirlo como una dependencia que se explicará en el anexo correspondiente.

Android Studio se trata de un IDE basando en IntelliJ Idea, desarrollado y destinado exclusivamente para el tratamiento de aplicaciones Android, por lo que es la alternativa más eficiente y correcta.

Sistema Operativo

El sistema operativo elegido es, evidentemente, Android ya que la aplicación debería estar destinada a Modelo de aplicaciones de realidad aumentada en dispositivos Android, teniendo en cuenta todo lo comentado en el Estado del arte, Android es uno de los sistemas operativos más empleados actualmente y el idóneo para lanzar al mercado aplicaciones de este tipo.

Se consideraron otros sistemas operativos como iOS, pero no se disponía del material adecuado para el desarrollo de estas aplicaciones, ya que el uso de estas aplicaciones es exclusivo de los dispositivos Apple, material del cual no se disponía.

Para iniciar el desarrollo de este tipo de aplicaciones es necesario un SDK de Android que viene previamente instalado en Android Studio, por lo que, al instalar el paquete entero estaremos listo para el desarrollo de nuestra aplicación.

SDKs

Los SDKs o Starter Developer Kit se tratan de bibliotecas de desarrollo que permiten cierta flexibilidad para el implementar determinadas aplicaciones, en este apartado comentaremos el SDK de realidad aumentada y las propias librerías de Google.

Bibliotecas Android

Las bibliotecas de Android son imprescindibles para el desarrollo de aplicaciones móviles para estos dispositivos, gracias a Android Studio estas bibliotecas vienen incorporadas por defecto en todas las aplicaciones que desarrollamos con dicho IDE.

Para asegurarnos de que estas bibliotecas están correctamente referenciadas deberemos irnos al Gradle correspondiente de la aplicación (app).

app ×

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 23
    buildToolsVersion "24.0.0"

    defaultConfig {
        applicationId "com.uah.artext"
        minSdkVersion 15
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.4.0'
    compile project(':droidAR')
```

SDK DroidAR

El SDK (Starter Developer Kit) de DroidAR está desarrollado íntegramente para Eclipse por lo que su soporte y manuales de instalación están completamente destinados a dicho entorno de desarrollo, a pesar de ello se decidió trabajar con Android Studio por lo que dicho SDK se migró a Android Studio gracias a las facilidades que esta herramienta ofrece.

Para ser más exactos, el SDK de DroidAR se incorpora al proyecto como si fuera un módulo en vez de una dependencia ya que estas son tecnologías Gradle recientes, cosa que no integraba el proyecto (Ya que es de 2012, momento en el que Eclipse estaba en auge para el desarrollo de este tipo de aplicaciones).

Por otro lado, esto nos ofrece una ventaja y es que al ser de código abierto y poder integrar el proyecto como un módulo, se pueden editar todas y cada una de las clases que este contiene para adaptarse a las necesidades de cada proyecto.

En el anexo correspondiente se va a explicar cómo se instala el módulo de realidad aumentada en el proyecto, ya que el código de eclipse debe migrarse a Android Studio.

Versión del SDK Android

Para el correcto desarrollo de la aplicación se debe establecer una versión mínima del SDK, es decir, desde que dispositivo funcionará esta aplicación y hasta cual, además de especificar la versión de compilado para determinar que funcionalidades estarán disponibles y cuáles de ellas no.

```
android {
    compileSdkVersion 23
    buildToolsVersion "24.0.0"

    defaultConfig {
        applicationId "com.uah.artext"
        minSdkVersion 15
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
```

Para nuestro proyecto el SDK elegido es el 15 (API 4.0.3 Kit Kat), y compilaremos con la última de las versiones de Android hasta la fecha API 23 (Marshmallow).

Por ultimo resaltar, que en Android Studio es necesario que estas versiones se encuentren en el Gradle correspondiente al módulo de la aplicación, además de en el típico Manifest.

Desglose de permisos

Para que la aplicación funciones correctamente son necesarios ciertos permisos que nos den acceso a funcionalidades del hardware del teléfono, entre las que se encuentran la posición GPS, el uso de la brújula o magnetómetro y el acceso a internet.

Todos estos permisos deben estar declarados íntegramente en el Manifest de Android, para que, a la hora de instalar una aplicación, el usuario sepa que permisos está concediendo a la aplicación, todos los permisos usan el tag **<uses permission Android:name= …/>**

Permisos de localización: Necesarios para acceder al posicionamiento GPS del usuario que va a utilizar la aplicación, dos tipos la exacta y la aproximada.

- **`android.permission.ACCESS_COARSE_LOCATION`**: Actualización de la posición por el procesador de RED (WIFI o 3G/4G).
- **`android.permission.ACCESS_FINE_LOCATION`**: Actualización de la posición por señal GPS/GPRS.

Permisos de cámara: Permiso necesario para acceder a la cámara para visualizar el mundo exterior, y presentar en él los elementos virtuales.

- **`android.permission.CAMERA`**: Acceso directo al dispositivo interno de cámara trasera.

Permisos de internet: Permiso necesario para acceder a la cámara para visualizar el mundo exterior, y presentar en él los elementos virtuales.

- **`android.permission.INTERNET`**: Permite acceder directamente a Internet mediante los Sockets de conexión.

Permisos de vibración y almacenamiento: Permisos necesarios para acceder a las funcionalidades de la vibración o el almacenamiento interno.

- **`android.permission.VIBRATE`**: Permite al dispositivo vibrar cuando se pulsa sobre algún elemento de la cámara.

- **`android.permission.WRITE_EXTERNAL_STORAGE`**: Este permiso permite escribir datos en el almacenamiento del sistema, principalmente para SQLite.

4.3.2. Hardware

En este apartado describiremos el hardware que ha sido empleado para el desarrollo de las aplicaciones y sin el cual no podríamos haber probado y completado las tareas encomendadas en los requerimientos.

Por lo tanto, vamos a explicar concretamente cuales fueron las restricciones hardware que empleamos, estas son el PC y el móvil de pruebas empleado.

Móvil de pruebas

El móvil de pruebas empleados es el Hardware más importante en términos de desarrollado, puede utilizarse cualquier móvil que disponga de servicios de posicionamiento GPS, el hardware interno no es necesario que sea demasiado potente, ya que estas aplicaciones son modelos y no desarrollos complejos.

El móvil de pruebas empleados ha sido un Samsung Galaxy A5, equipado con dispositivos GPS, acelerómetro y sensores de orientación para la brújula. Este dispositivo cuenta con la versión 5.0 de Android (Lollipop).

Se probaron con otros dispositivos Android más antiguos utilizados previamente para otros desarrollos como puede ser el LG G2 Mini, aunque surgían problemas de localización GPS por lo que se desestimó el uso de este dispositivo.

Naturalmente, este dispositivo se emplea ya que la interacción con un dispositivo móvil es mucho más eficiente que sobre un emulador de Android, además de la posibilidad de desplazar el GPS para localizar los objetos con posiciones sin simular, para dar una experiencia mejor al usuario de desarrollo.

Por otra parte, este dispositivo es necesario ya que el emulador imposibilita la opción de mover la cámara para asegurar el posicionamiento de la brújula.

Equipo de desarrollo

El equipo de desarrollo empleado es aquel en el que se han instalado las necesidades Software arriba expuestas, se trata del ordenador utilizado para el desarrollo de las aplicaciones, este dispositivo con un año de antigüedad tiene las siguientes características.

El PC es un ordenador portátil de marca ASUS con las siguientes descripciones técnicas:

Caché	
4MB	
Modelo de procesador	
Intel® Core™ i7	
Núcleos del procesador	
2	
Velocidad	
2.4GHz (Hasta 3GHz con Turbo)	
Serie	
Intel Core i7 5500	

Memoria máxima	
4GB	
Tipo	
DDR3L a 1600MHz	
RAM	
4GB	

4.4. Arquitectura del sistema

Tras haber definido las técnicas que utilizaremos para el desarrollo de la aplicación podemos definir la arquitectura del sistema, es decir, como se comunicaran los diferentes módulos entre sí.

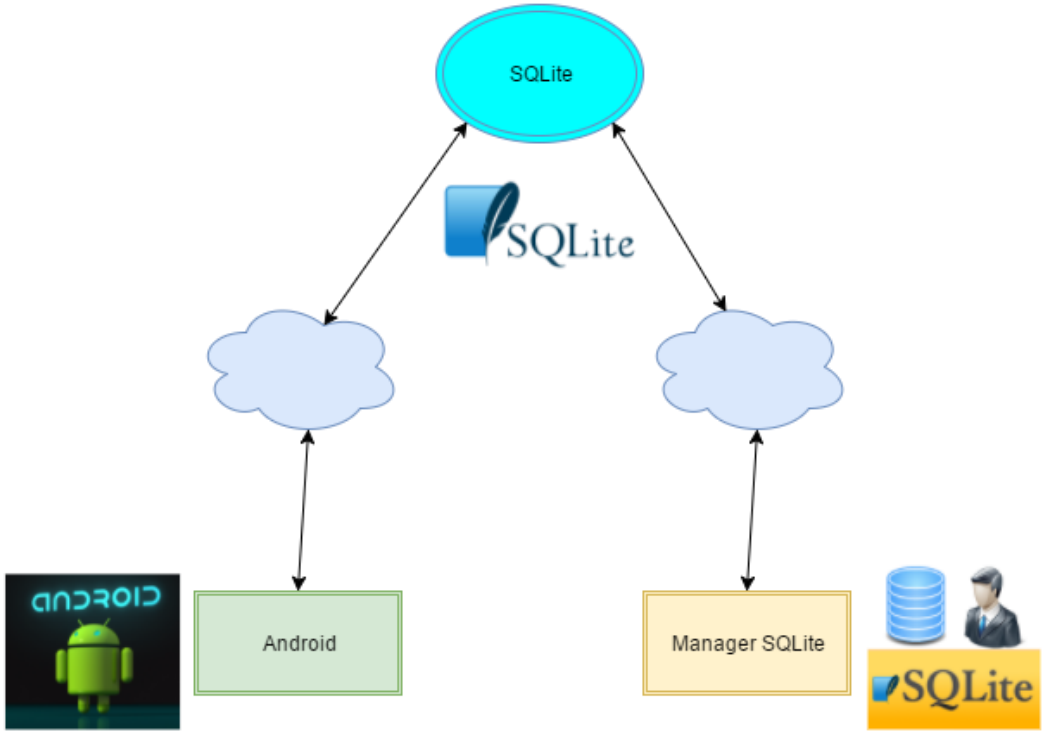
Como se ha decidido anteriormente en el Estado del arte, la arquitectura del sistema dependerá exclusivamente del teléfono ya que utilizaremos la base de datos interna de las aplicaciones ya que se trata de una aplicación modelo, lo más recomendable sería migrar estos datos a unas bases de datos más sostenibles, pero esto ya lo discutiremos en apartados posteriores.

Todas las aplicaciones que desarrollaremos por tanto atacaran al mismo sistema de base de datos SQLite, con la ventaja de que necesitaremos internet para acceder o modificar estos datos, se podría decir que el teléfono "ejerce de servidor" aunque esto no sea exactamente así ya que los datos almacenados se cargan en un archivo de teléfono en el teléfono.

Por otro lado, el cliente será también el mismo teléfono que hará la comunicación cuando corresponda mediante los controladores proporcionados por Google para el desarrollo de este tipo de bases de datos.

Para la administración de la base de datos se puede editar directamente el archivo de texto, aunque existen otros editores interactivos como si se tratase de una base de datos como tal, por ejemplo, **SQLite Manager** [13] desde el cual puedes realizar consultas directamente a la base de datos.

Aunque finalmente se decidió no utilizar ningún gestor, ya que solo se deseaba almacenar los datos de manera temporal a modo de modelo, de igual manera esta sería la arquitectura resultante de esta aplicación, siempre y cuando esta se comunique con el ordenador.



4.5. Problemas encontrados

Durante el desarrollo de nuestras aplicaciones nos hemos encontrado con ciertos problemas algunos de los cuales han podido ser solventados, aunque otros no, la mayoría de estos han sido derivados de la falta de documentación sobre el SDK.

Teniendo en cuenta que el SDK de desarrollo se lanzó en el 2012 y que solo ha sido actualizado un par de veces en los últimos años esto propicia una dificultad al intentar comprender los elementos que este dispone, sin embargo, este problema pudo ser resuelto gracias a la gran variedad de ejemplos que este dispone además de los videos orientativos que se publicaron en su momento.

Por otro lado, la base de datos escogida no es la más correcta para la visualización de datos, SQLite dispone de un motor de consultas interno, es decir, dentro de la aplicación puedes ver los datos, pero si quieres extraerlos necesitas de herramientas externas además de tener permisos para acceder a ciertos directorios, los cuales no están disponibles para todo dispositivo.

Finalmente, el primer dispositivo de pruebas, cuyo GPS no funcionaba correctamente lo cual ralentizó el desarrollo de la aplicación. Se recomienda usar un dispositivo con buena señal GPS.

5. Planteamiento y desarrollo

En este apartado vamos a tratar de desglosar las necesidades de una aplicación modelo y cuáles serán las funcionalidades idóneas para que posteriormente se pueda retomar el trabajo integrando los procedimientos que desarrollaremos.

Por un lado, se podría desarrollar una sola aplicación con cada una de las funcionalidades especificadas dentro de un botón en la interfaz y según se seleccione uno u otro abriría una u otra demostración, es decir, el trabajo sería desarrollar un solo producto que nos permita modelizar los trabajos próximos, sin embargo, se decidió dedicar una sola aplicación a cada una de las funcionalidades útiles del SDK comentadas en el Estado del arte, así por ejemplo, los textos flotantes formarían su propia aplicación con su documentación pertinente.

El sistema por tanto estará formado por las siguientes aplicaciones:

- **Aplicación modelo texto:** La siguiente aplicación deberá presentar textos con diferentes formatos de letras tamaños de texto, además se deberá presentar una funcionalidad que desplace el texto en el espacio vectorial.

- **Aplicación minijuego:** La siguiente aplicación consiste en una pantalla con un temporizador que va disminuyendo a casa segundo, el usuario tiene un tiempo para caza el mayor número de diamantes.

- **Aplicación información emergente:** La última aplicación y la más compleja de todas permite al usuario desplazarse por el espacio utilizando su cámara, cuando se tope con una posición GPS concreta se le mostrará un elemento emergente, que al pulsarlo realizara una acción concreta, por ejemplo, mostrar una página web.

Teniendo en cuenta los criterios anteriores se sabe que el desarrollo se desglosará por aplicación, por tanto, y atendiendo a este criterio en este apartado describiremos cada una de las funcionalidades de estas aplicaciones, lo que no servirá para posteriormente para especificar correctamente los requisitos, es decir, para que el código de la aplicación sea legible para una futura reutilización en este apartado vamos a procurar dejar lo más claro posible todas las partes de las que se compone la aplicación, basándonos en unos requisitos y funcionalidades.

Por lo tanto, podemos decir que desde este momento comienza la fase de análisis correspondiente del ciclo de vida software en cascada sobre el cual nos estamos basando a la hora desarrollar la aplicación.

Por último, en este apartado entraremos en detalle del desarrollo de cada una de las aplicaciones descritas subdividido por aplicación como se comentaba antes, para cada una de ellas se explicará el desarrollo en su conjunto dando detalles de implementación para cada parte fundamental de la aplicación usando la documentación expuesta en [14].

5.1. Prototipo 1: Aplicación Textos

En este apartado se van a describir las restricciones y funcionalidades correspondientes a la aplicación de textos, además hablaremos de la implementación del sistema de texto flotantes, podríamos dividir el desarrollo en dos secciones la del diseño de la aplicación, en la cual se incluirían las imágenes de la pantalla de presentación, los colores, tipos de letra, imágenes de la brújula etc., y por otro lado la parte programática que es la que se encarga de realizar las acciones correspondientes a cada funcionalidad.

5.1.1. Funcionalidades

Las funcionalidades para la aplicación de textos van definidas de acuerdo a la siguiente tabla:

Funcionalidad	Explicación
Texto dinámico	El usuario debe ser capaz de modificar el texto que se le presenta de manera dinámica, el campo de texto tendrá las siguientes restricciones: • El campo de texto contendrá infinitos caracteres.
Texto móvil en el plano	El texto debe desplazarse en el plano durante la ejecución de la aplicación, no será necesario emplear ninguna acción.
Acción al pulsar sobre el texto	Si el usuario pulsa sobre la cámara en el punto en el que se encuentra se debe lanzar una acción, la acción podría ser: • El texto se desplaza en el plano XYZ a una posición aleatoria.
Posicionamiento con brújula	La aplicación debe ser capaz de localizar la orientación del dispositivo, alternativamente se presentan la siguiente característica: • La imagen de la brújula debe ser personalizable.
Pantalla de presentación	Pequeño SplashScreen a modo de presentación de la aplicación.
Botones de vista pájaro	Al usuario se le deben presentar unos botones que le permitan reducir o aumentar la vista en el eje Z.
Presentación de la cámara	El software debe ser capaz de presentar la cámara del móvil para representar los objetos en el espacio de la misma.

5.1.2. Requisitos

Como se comentaba en el apartado anterior (**Funcionalidades)** vamos a basarnos en estas funcionalidades solicitadas para evitar que los requisitos hayan sido mal capturados, de esta manera no aseguramos que la fase de revisión todo este en su sitio.

Los requerimientos se establecerán de acuerdo a las tablas expuestas a continuación:

Requisitos funcionales

Número del requisito	RF1.1
Nombre del requisito	Presentación de cámara
Prioridad	Alta
Descripción	La aplicación debe ser capaz de presentar al usuario la vista actual de su cámara para evaluar los posibles objetos en el espacio.

Número del requisito	RF2.1
Nombre del requisito	Pantalla de presentación
Prioridad	Media
Descripción	La aplicación debe mostrar una pequeña pantalla de presentación (Splash)

Número del requisito	RF3.1
Nombre del requisito	Orientación brújula
Prioridad	Alta
Descripción	En todo momento debe estar presente una brújula que sea capaz de orientar en los puntos cardinales al usuario.

Número del requisito	RF4.1
Nombre del requisito	Posicionamiento en base de datos
Prioridad	Media/Baja
Descripción	Cada vez que el usuario pulse sobre el texto flotante se debe almacenar en la base de datos la hora y la posición GPS del mismo.

Número del requisito	RF5.1
Nombre del requisito	Texto flotante móvil
Prioridad	Alta
Descripción	Al usuario se le debe presentar un texto móvil, es decir, que cuando se le pulse se mueva en el plano.

Número del requisito	RF6.1
Nombre del requisito	Modificar texto
Prioridad	Alta
Descripción	Al usuario se le mostrará una caja de texto que cambiará el texto por el deseado.

Número del requisito	RF7.1
Nombre del requisito	Formato de texto
Prioridad	Alta
Descripción	Para la programación del texto se debe poder elegir un tipo de texto y un tamaño de letra.

Número del requisito	RF8.1
Nombre del requisito	Color de texto
Prioridad	Alta
Descripción	Para la programación del texto se debe poder elegir un tipo de color.

Requisitos no funcionales

Número del requisito	RNF1.1
Nombre del requisito	Base de datos
Prioridad	Media/Baja
Descripción	La base de datos elegida para almacenar los datos será SQLite.

Número del requisito	RNF2.1
Nombre del requisito	Sistema operativo
Prioridad	Alta/Imprescindible
Descripción	El sistema operativo elegido para el desarrollo será Android.

Número del requisito	RNF3.1
Nombre del requisito	GPS Móvil
Prioridad	Imprescindible
Descripción	Para que la aplicación funcione correctamente el móvil debe tener los sistemas GPS activos.

Número del requisito	RN4.1
Nombre del requisito	Espacio de almacenamiento
Prioridad	Alta
Descripción	La aplicación desarrollada no debe ocupar más de 20 Mb en el disco del móvil.

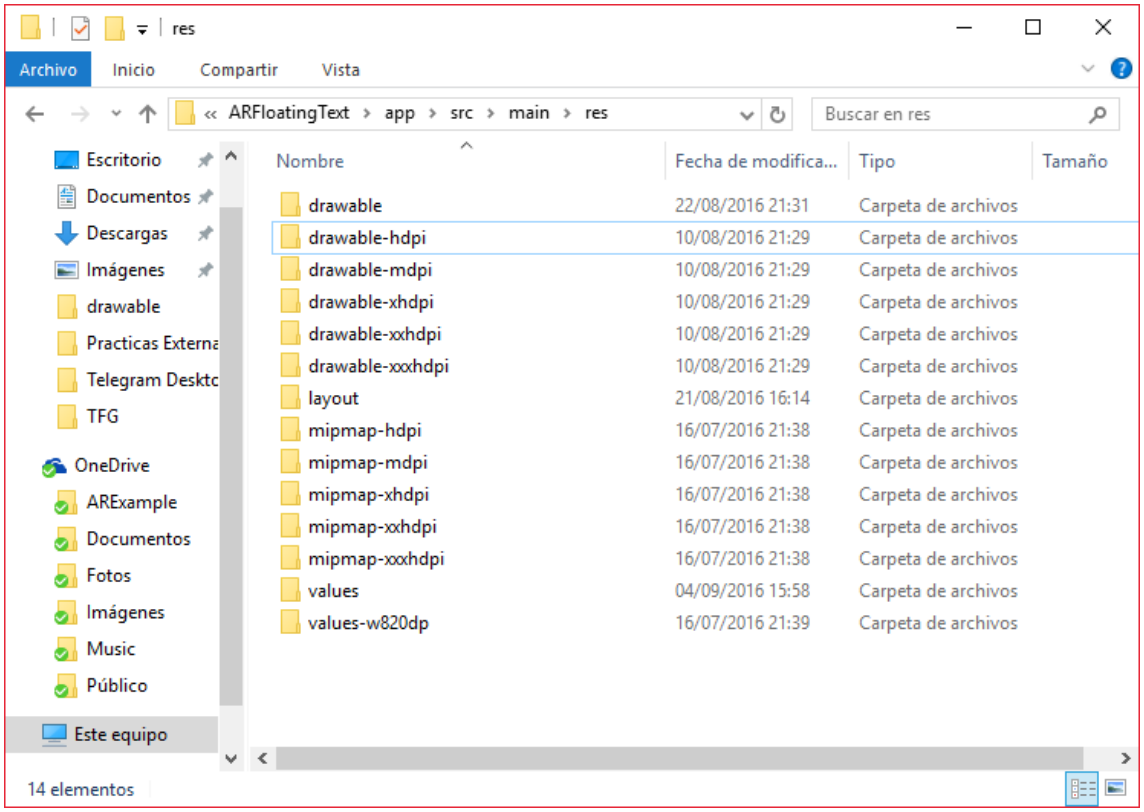
5.1.3. Diseño artístico

Para cada una de las pantallas se diseñó un concepto artístico, es decir, la pantalla de presentación tiene diferentes elementos que la pantalla de textos, ya que esta esencialmente muestra la cámara del usuario.

Comenzando por la pantalla de presentación se decidió utilizar el logo de la Universidad de Alcalá a modo de presentación seguido de un texto orientativo que determina la aplicación sobre la que estamos trabajando. El fondo de esta aplicación se trata de un degradado obtenido de internet que permite utilizarse como plantilla para los diferentes tamaños de pantalla que Android posee, de acuerdo a su documentación los tamaños son los siguientes: **(Véase** [15]**)**

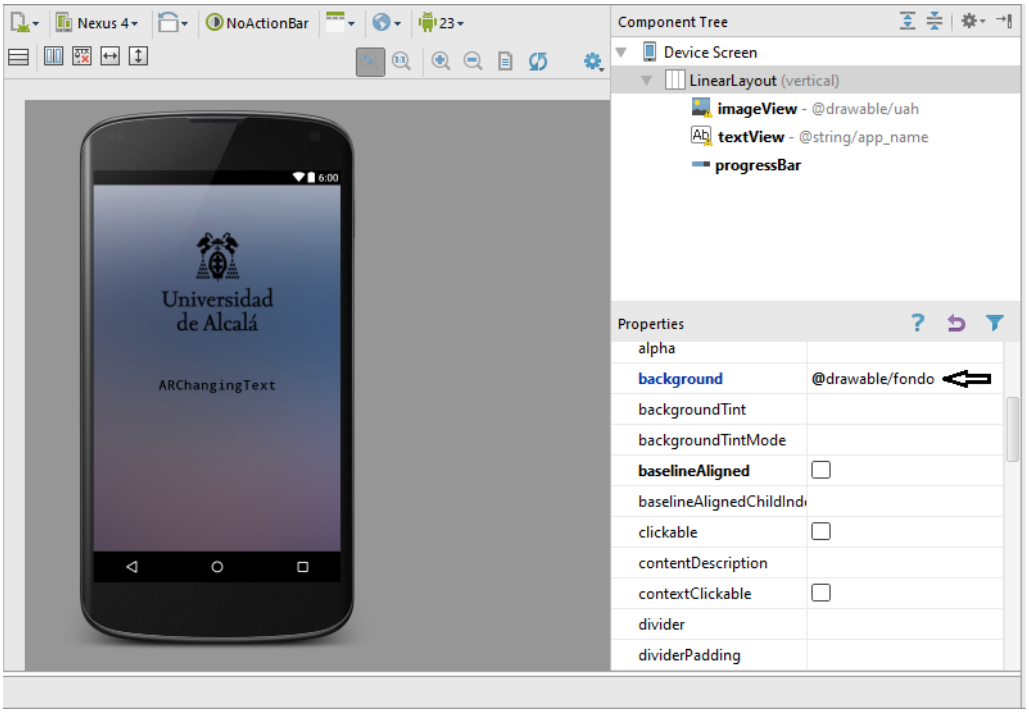
```
xxxhdpi: 1280x1920 px
xxhdpi: 960x1600 px
xhdpi: 640x960 px
hdpi: 480x800 px
mdpi: 320x480 px
ldpi: 240x320 px
```

Por lo tanto, deberemos distribuir por las diferentes carpetas de recursos, por ejemplo, res/drawable hdpi las diferentes imágenes de fondo para la pantalla de presentación.



De esta manera nos aparecerán automáticamente como recursos disponibles para la programación de las aplicaciones, para acceder a estos datos utilizaremos directamente la clase de recursos Android (R), aunque esto lo comentaremos en la parte programática.

Este fondo se seleccionará directamente desde la interfaz de Android, dándole una imagen como propiedad a una de las capas o Layouts.

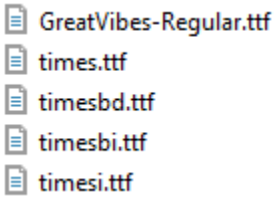


Por último, cabe destacar que los elementos correspondientes a la interfaz de realidad Aumentada se deben programar de manera autónoma ya que la pantalla de la cámara se trata de una vista diseñada por el creador del propio SDK de **DroidAR**.

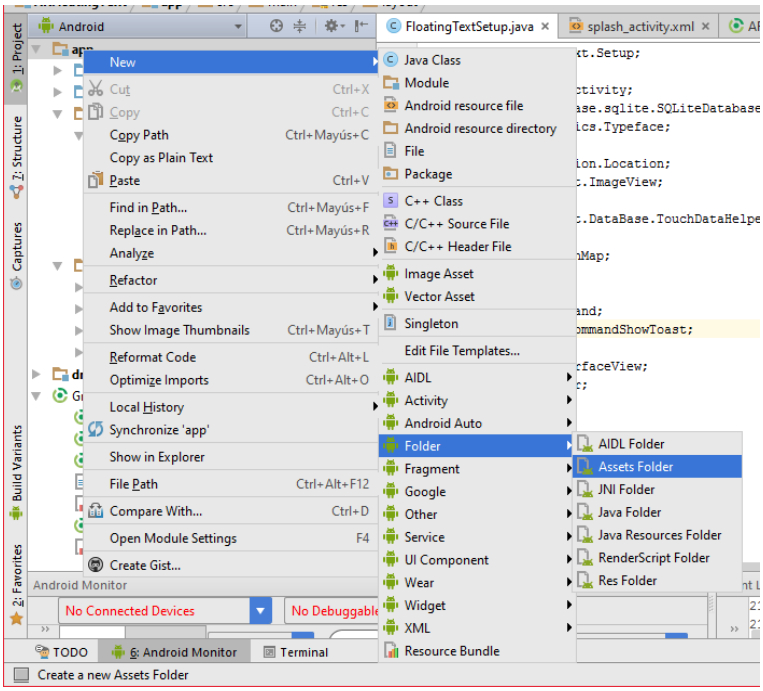
Los elementos que se han tenido que programar son los siguientes:

- Brújula: La imagen de la brújula se ha tenido que introducir manualmente sobre la vista predeterminada de DroidAR, gracias a las funciones que ofrece podemos introducir cualquier componente de Android de manera programática. Esto lo explicaremos en el apartado correspondiente.

- Tipo de letra para el texto: El tipo de letra por defecto de Android es el utilizado para presentar los textos virtuales, sin embargo, esto no es lo que decían los requerimientos, por lo tanto, deberemos tener un tipo de texto con extensión ttf.



Para la introducción de este tipo de textos es necesario agregar la carpeta assets al proyecto, para ello, podemos utilizar la interfaz de Android Studio, deberemos situarnos sobre el modulo correspondiente a la aplicación (app), pulsar click derecho y buscar la opción New -> Folder -> Assets, este proceso nos creará una carpeta para almacenar todo aquello que no sean imágenes, textos, animaciones, etc. [16]



Posteriormente deberemos añadir una carpeta fonts, desde la que accederemos con la programación, ahí almacenaremos nuestras fuentes.

5.1.4. Implementación

Una vez que hemos descrito todos los elementos no programáticos podemos proceder con la descripción técnica del proyecto, esta es la parte de programación, en este apartado desglosaremos las funciones imprescindibles para el funcionamiento correcto de la aplicación, entre estas está la gestión de la base de datos, la gestión de la brújula o la clase de SplashScreen de presentación.

SplashScreen

Para el desarrollo de la pantalla de presentación utilizaremos la codificación clásica de un SplashScreen al cual habrá que introducirle un retardo mediante el cual programaremos una tarea que se ejecutará cuando este tiempo pase. **(Véase** [17]**)**

```
TimerTask task = () → {

        ArActivity.startWithSetup(SplashActivity.this,
                new FloatingTextSetup(compass, Typeface.createFromAsset(getAssets(),"fonts/times.ttf")));

        // Close the activity so the user won't able to go back this
        // activity pressing Back button
       SplashActivity.this.finish();
};

// Simulate a long loading process on application startup.
Timer timer = new Timer();
timer.schedule(task, SPLASH_SCREEN_DELAY);
```

El tiempo estándar normalmente se sitúa en los 5 segundos, esto se especifica en la variable SPLASH_SCREEN_DELAY, la cual debe ser final y estática, ya que se trata de una constante.

La tarea que se ejecuta cuando se termina el Timer es la de finalizar la actividad para que no se pueda retornar a ella e iniciar una nueva actividad con ArActivity y el Setup que hemos programado para los textos, este recibe una brújula y un tipo de texto.

```
image = new ImageView(this);

image.setImageDrawable(getResources().getDrawable(R.drawable.pointer));
image.setLayoutParams(new ActionBar.LayoutParams(ActionBar.LayoutParams.WRAP_CONTENT, ViewGroup.LayoutParams.WRAP_CONTENT))
```

Por último, esta clase debe introducir una clase Compass que realiza las funciones de brújula mediante la orientación del dispositivo, aunque esta no es la configuración que se usa actualmente, a esta clase se le pasa una imagen que será aquella que se presente en la pantalla, y por lo tanto deberemos generarla. La imagen se obtiene de la clase R correspondiente a los recursos expuestos para el proyecto (*R.drawable.pointer*).

Compass

La clase Compass permite especificar la imagen que se utilizará para presentarla en la pantalla, además debe implementar el SensorListener, para determinar cuándo se ha cambiado la posición del dispositivo. **(Véase** [18]**)**

```
public class Compass implements SensorEventListener{

    // record the compass picture angle turned
    private float currentDegree = 0f;
    // device sensor manager
    private final SensorManager mSensorManager;
    // compass arrow to rotate
    public ImageView arrowView = null;

    public Compass(Context context, ImageView arrowView) {
        // initialize your android device sensor capabilities
        mSensorManager = (SensorManager) context.getSystemService(Context.SENSOR_SERVICE);
        this.arrowView = arrowView;
    }
```

La clase implementará los métodos, en estos deberemos determinar la posición de la brújula utilizando animaciones de rotación, esto se determina en la función onSensorChange. La implementación ha sido la siguiente:

```
@Override
public void onSensorChanged(SensorEvent event) {
    // get the angle around the z-axis rotated

    float degree = Math.round(event.values[0]);

    // create a rotation animation (reverse turn degree degrees)
    RotateAnimation ra = new RotateAnimation(currentDegree,-degree,
            Animation.RELATIVE_TO_SELF, 0.5f,
            Animation.RELATIVE_TO_SELF,
            0.5f);

    // how long the animation will take place

    ra.setDuration(210);

    // set the animation after the end of the reservation status

    ra.setFillAfter(true);

    // Start the animation

    arrowView.startAnimation(ra);

    currentDegree = -degree;
```

Por último, tenemos los métodos que nos permitirán iniciar y parar los sensores para permitir ahorrar batería, estos son los métodos start() y stop() que deberán ser invocados en el onStart() y onStop() de la actividad que los implemente. También hemos introducido una función que nos devuelve la imagen de la flecha que indica la posición.

```
public void start() {

    // for the system's orientation sensor registered listeners
    mSensorManager.registerListener(this,
mSensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
SensorManager.SENSOR_DELAY_GAME);

}


public void stop() {

    // to stop the listener and save battery

    mSensorManager.unregisterListener(this);

}

public ImageView getArrowView() {
    return arrowView;
}
```

TouchDataBaseHelper

La siguiente clase permite la gestión de la base de datos para ello deberemos extender de la clase SQlite OpenHelper, este se encarga de abrir nuevos archivos de bases de datos para su creación, por lo tanto, en esta clase especificaremos los campos de la base de datos y su nombre.

```
public class TouchDataHelper extends SQLiteOpenHelper {

    String sqlCreate = "CREATE TABLE Location (codigo INTEGER, texto TEXT, lat LONG, lon LONG)";

    public TouchDataHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) { db.execSQL(sqlCreate); }


    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        //NOTA: Por simplicidad del ejemplo aquí utilizamos directamente la opción de
        //      eliminar la tabla anterior y crearla de nuevo vacía con el nuevo formato.
        //      Sin embargo lo normal será que haya que migrar datos de la tabla antigua
        //      a la nueva, por lo que este método debería ser más elaborado.

        //Se elimina la versión anterior de la tabla
        db.execSQL("DROP TABLE IF EXISTS Location");

        //Se crea la nueva versión de la tabla
        db.execSQL(sqlCreate);
    }
}
```

Los dos métodos que se emplean son onUpdate y onCreate, estos dos métodos permiten crear una base de datos por primera vez y actualizarla entre versiones, en nuestro caso si hay una actualización simplemente crearemos de nuevo la base de datos, ya que no planeamos modificar los campos que esta contiene. **(Véase** [19]**)**

FloatingTextSetup

Esta clase es la que hace todo el trabajo de la realidad aumentada, a la actividad **ArActivity**, predefinida del SDK de desarrollo hay que introducirle un Setup o configuración, para ello creamos una clase que extienda de **Setup**, si bien, este no es nuestro caso, ya que deseamos implementar una actividad que espere por la precisión del GPS, esto lo podemos conseguir extendiendo de la clase **DefaultARSetup.**

En esta clase tenemos una serie de métodos que permiten situar los textos o cualquier elemento sobre el plano cuando el GPS haya adquirido la suficiente exactitud, para evitar que las lecturas de la posición sean incorrectas.

```
 * Este metodo se utiliza para esperar a que el GPS obtenga toda la exactitud necesaria.
 * @param renderer El elemento de renderizado de GL.
 * @param world El mundo al que añadir los objetos.
 * @param objectFactory Para generar objetos de OpenGL.
 */
public void addObjectsTo(GL1Renderer renderer, World world, GLFactory objectFactory) {
    // Aqui se añade el texto al mundo, con la referencia al hashmap y la camara que lo dibuja.
    text = new GLText("Texto de ejemplo!!!", myTargetActivity, textMap,
            getCamera(),letra);
    // Para añadir cualquier elemento de realidad virtual este debe ser un objeto Obj, lo creamos.
    Obj o = new Obj();
    o.setComp(text);
    o.setPosition(new Vec(0,5,0));
    // Finalmente lo añadimos.
    world.add(o);
    text.setOnClickCommand(() -> {
            SQLiteDatabase dbo = db.getWritableDatabase();
            Location loc = camera.getGPSLocation();
            //Hacemos el insert en la base de datos de nuestro texto actual y su localizacion.
            dbo.execSQL("INSERT INTO Location (codigo,texto,lat,lon)" +
                    "VALUES (" + counter +",\""+ text.getText() + "\"," + loc.getLatitude() + "," + loc.getLongitude() +");")
            text.setPosition(Vec.getNewRandomPosInXYZ(camera.getPosition(),1,3));
            CommandShowToast.show(getActivity(),"Insertada posicion y texto en base de datos. :D");
            counter++;
            dbo.close();
            return false;
```

Una vez generado el texto debemos introducirle una acción al pulsarlo, en este caso es que se posicione aleatoriamente en otro sitio de la cámara entre 1 y 3 metros, además utilizamos la base de datos para insertar los datos que contenía y en qué posición nos encontramos actualmente, además deberemos añadir un paso más al contador para evitar conflictos en la base de datos, ya que ese identificador es único.

La posición aleatoria del texto se consigue mediante las funciones expuestas del SDK, utilizando la clase estática **Vec** y accediendo a una de sus funciones **getNewRandomPosition**, si bien esto también se puede cambiar manualmente especificando un nuevo vector x,y,z cuyas posiciones son relativas a la cámara.

La base de datos, el contador y el texto serán unas variables de la clase que se crearán de la siguiente manera y algunas de las cuales deberán ser inicializadas en el constructor de la clase correspondiente, si bien también disponemos de un método **initFieldsIfNecessary** mediante el cual podemos inicializar cualquier variable que no haya sido posible durante la creación de la clase para su uso, ahorrando de esta manera memoria.

```
/* El tipo de letra deseado. */
private final Typeface letra;
/* La clase de la base de datos. */
private TouchDataHelper db;
/* La imagen para la brujula. */
private ImageView image;
/* El texto flotante que se desea agregar a la camara. */
private GLText text;

/* La estructura de datos que contiene los textos almacenados */
private HashMap<String, MeshComponent> textMap = new HashMap<>();
/* El contador de veces que pulsamos para usarlo de codigo. */
private int counter;
```

Como podemos ver también será necesario inicializar la letra y la imagen de la brújula que extraeremos de la clase correspondiente (*Compass*).

En nuestro caso, y para evitar conflictos de apertura y cierre de la base de datos, utilizaremos el método expuesto por el SDK para abrir la conexión con la base de datos, lo que nos generara un archivo en el almacenamiento interno del teléfono.

Antes de pasar a explicar los métodos internos de la clase DefaultARSetup vamos a explicar cómo se introducen elementos de Android en la pantalla de la cámara, el SDK expone cuatro posiciones para colocar los elementos, estos son: izquierda, derecha, arriba y abajo. Estas posiciones son paneles que se han colocado sobre la vista de la cámara y a los cuales programáticamente se le añaden los elementos.

```
/**
 * En este metodo se tiene que añadir la brujula y el texto de la localización.
 * @param guiSetup variable que permite añadir objetos Android al mundo.
 * @param activity this is the same activity you can get with
 *            {@link Setup#myTargetActivity} but its easier to access this
 */
public void _e2_addElementsToGuiSetup(GuiSetup guiSetup, Activity activity) {
    super._e2_addElementsToGuiSetup(guiSetup,activity);
    // Alineamos a la derecha y metemos los componentes.
    guiSetup.setTopViewAllignRight();
    guiSetup.setRightViewAllignBottom();
    guiSetup.addViewToTop(image);
    guiSetup.addSearchbarToView(guiSetup.getBottomView(), new Command() {

        @Override
        public boolean execute() { return false; }

        @Override
        public boolean execute(Object transfairObject) {
            if (transfairObject instanceof String) {
                String s = (String) transfairObject;
                if (text != null)
                    text.changeTextTo(s);
            }
            return true;
        }
    }, "Escribe algo y pulsa enter");
```

Arriba se describen las líneas de código empleadas para añadir la barra de escritura y la brújula, además tenemos una llamada a la clase padre mediante super, esto permite además añadir las flechas de posición sobre el eje Z para poder ver correctamente el texto una vez aparezca.

Además, la caja de texto introduce una acción al enviar los datos, esto lo que hace es comprobar el texto que se ha introducido mediante el Objeto que realiza la acción execute, finalmente si el texto contiene algo se cambia por el deseado.

GLText

Al ser un SDK de código abierto podemos realizar todos los cambios que veamos convenientes sobre el código, en este caso nos han sido necesarias dos modificaciones, ya que el texto virtual no introducía la posibilidad de utilizar otros tipos de letras, pero sí que introducía la posibilidad de ponerlo en negrita un procedimiento muy similar al cambio de letra.

Para ello creamos un constructor con los valores por defecto, además añadimos el campo TypeFace, que permite elegir el tipo de texto, cuando no se le introduce ningún valor se utiliza la letra por defecto de Android, cuando se le introduce un valor si la letra está en la carpeta assets se cambia por la letra seleccionada.

```
/**
 * @param text
 *            The text that should be displayed. Can be changed with
 *            {@link GLText#changeTextTo(String)}
 * @param context
 * @param textMap
 *            hte already created characters will be stored in here,
so pass a new {@link HashMap} or maybe also a already initialized
 * @param glCamera
 *            to allow the text to face the camera
 * @param textStyle
 *             to change the text typography.
 */
public GLText(String text, Context context,
          HashMap<String, MeshComponent> textMap, GLCamera
glCamera, Typeface textStyle) {
   super(null);
   myText = text;
   myContext = context;
   myTextMap = textMap;
   myCamera = glCamera;
   type = textStyle;
}
```

Finalmente, se añadió un método para recoger el texto y poder almacenarlo en la base de datos, ya que esta clase solo ofrecía la posibilidad de cambiar el texto, el texto por otra parte, es tratado como una imagen para poder cargarlo sobre la cámara. También cambiamos el color de texto mediante el método **setTextColor**, especificando un color de Android, aunque siempre se puede elegir un color de la gama de colores del proyecto.

```
/**
 * @param s El texto a generar
 * @author Javier Palomares
 **/
protected Bitmap generateText(String s) {
    TextView v = new TextView(myContext);
    v.setTextColor(android.graphics.Color.BLUE);
    // Cambia el tipo al correcto.
    v.setTypeface(type, Typeface.BOLD);
    v.setText(s);
    return util.IO.loadBitmapFromView(v);
}

public void changeTextTo(String s) {
    myText = s;
    clearChildren();
    textLoaded = false;
}
```

DefaultARSetup

De esta clase nos interesan los métodos que permiten determinar si el GOS tiene suficiente exactitud para poder continuar con la ejecución de la aplicación, esto se introduce en el método **addActionsToEvents** y se trata de una acción de **LocationChanged**, que se llama por primera vez para comprobar la posición GPS del usuario.

Cuando se llega a la posición GPS exacta o al menos un 75% de exactitud se llama al método **addObjectToWorld**, el cual habíamos implementado en nuestra configuración.

```
minAccuracyAction = new ActionWaitForAccuracy(getActivity(), 24.0f,
10) {
   @Override
   public void minAccuracyReachedFirstTime(Location l,
                                    ActionWaitForAccuracy a) {
      callAddObjectsToWorldIfNotCalledAlready();
      if (!eventManager.getOnLocationChangedAction().remove(a)) {
         Log.e(LOG_TAG,
               "Could not remove minAccuracyAction from the
onLocationChangedAction list");
      }
   }
};
eventManager.addOnLocationChangedAction(minAccuracyAction);
```

Por último, y como se comentaba esta clase añade al mundo los botones que reducen o aumentan el objetivo en el eje Z, en 5 puntos.

```
@Override
public void _e2_addElementsToGuiSetup(GuiSetup guiSetup, Activity activity) {
    guiSetup.setRightViewAllignBottom();

    guiSetup.addViewToTop(minAccuracyAction.getView());

    guiSetup.addImangeButtonToRightView(R.drawable.arrow_up_float,
            () -> {
                    camera.changeZPositionBuffered(+ZDELTA);
                    return false;
            });
    guiSetup.addImangeButtonToRightView(R.drawable.arrow_down_float,
            () -> {
                    camera.changeZPositionBuffered(-ZDELTA);
                    return false;
            });

}
```

En el apartado correspondiente del **Manual de usuario** podremos ver cómo funcionan estos métodos dentro del móvil, y como cambian los tipos de texto, el texto y la letra junto con su color, además de almacenar en la base de datos el texto junto con la localización para su posterior consulta con el gestor si es lo que se desea, aunque no es el caso.

5.2. Prototipo 2: Aplicación minijuego

En este apartado se van a describir las restricciones y funcionalidades correspondientes a la aplicación minijuego, además vamos a hablar de la implementación de la misma, idéntico a la anterior, dividiremos el desarrollo en diseño de la aplicación, en la cual se incluirían las imágenes y recursos empleado en la creación de la misma, y por otro lado la parte programática que es la que se encarga de realizar las acciones.

5.2.1. Funcionalidades

Las funcionalidades para la aplicación de textos van definidas de acuerdo a la siguiente tabla:

Funcionalidad	Explicación
Temporizador	En la parte superior de la pantalla deberá aparecer un temporizador, que irá corriendo para indicar el tiempo restante. • El tiempo debe ser variable.
Figuras en el plano	Sobre la cámara deberán aparecer las distintas figuras que el usuario deberá capturar, de diferentes características.
Puntuación variante	La puntuación que el jugador obtiene debe variar en función de las características de la figura capturada: • El color es este factor.
Pantalla de presentación	Pequeño SplashScreen a modo de presentación de la aplicación.
Presentación de la cámara	El software debe ser capaz de presentar la cámara del móvil para representar los objetos en el espacio de la misma.
Dialogo informativo	Al finalizar la partida (se acaba el temporizador) se debe mostrar un dialogo en el que se debe introducir un nombre y en el cual se le mostrara la puntuación al usuario.

5.2.2. Requisitos

Como se comentaba en el apartado anterior (**Funcionalidades)** vamos a basarnos en estas funcionalidades solicitadas para evitar que los requisitos hayan sido mal capturados, de esta manera no aseguramos que la fase de revisión todo este en su sitio.

Los requerimientos se establecerán de acuerdo a las tablas expuestas a continuación:

Requisitos funcionales

Número del requisito	RF1.1
Nombre del requisito	Presentación de cámara
Prioridad	Alta
Descripción	La aplicación debe ser capaz de presentar al usuario la vista actual de su cámara para evaluar los posibles objetos en el espacio.

Número del requisito	RF2.1
Nombre del requisito	Pantalla de presentación
Prioridad	Media
Descripción	La aplicación debe mostrar una pequeña pantalla de presentación (Splash)

Número del requisito	RF3.1
Nombre del requisito	Puntuación en base de datos
Prioridad	Media/Baja
Descripción	Cada vez que el usuario termine de jugar al minijuego, es decir, se ha agotado el tiempo se debe almacenar un nombre y una puntuación.

Número del requisito	RF4.1
Nombre del requisito	Diamantes flotantes
Prioridad	Alta
Descripción	Alrededor del usuario deberán aparecer diversos diamantes que deberá capturar para obtener puntuaciones, cada diamante tendrá su puntuación.

Número del requisito	RF5.1
Nombre del requisito	Dialogo final
Prioridad	Alta
Descripción	Tras acabarse el temporizador deberá aparecer un dialogo en el que el usuario deberá introducir su nombre y se le guardará la puntuación en la base de datos.

Número del requisito	RF6.1
Nombre del requisito	Temporizador variables
Prioridad	Alta
Descripción	Para la programación del temporizador se debe poder elegir el tiempo que este dura, para que el minijuego tenga un tiempo variable.

Requisitos no funcionales

Los requisitos no funcionales de esta aplicación son muy similares, ya que este tipo de aplicaciones requiere las mismas capacidades, estas dependen de las funciones expuestas.

Número del requisito	RNF1.1
Nombre del requisito	Base de datos
Prioridad	Media/Baja
Descripción	La base de datos elegida para almacenar los datos será SQLite.

Número del requisito	RNF2.1
Nombre del requisito	Sistema operativo
Prioridad	Alta/Imprescindible
Descripción	El sistema operativo elegido para el desarrollo será Android.

Número del requisito	RNF3.1
Nombre del requisito	GPS Móvil
Prioridad	Imprescindible
Descripción	Para que la aplicación funcione correctamente el móvil debe tener los sistemas GPS activos.

Número del requisito	RN4.1
Nombre del requisito	Espacio de almacenamiento
Prioridad	Alta
Descripción	La aplicación desarrollada no debe ocupar más de 20 Mb en el disco del móvil.

5.2.3. Diseño artístico

Similar a la anterior aplicación para cada pantalla se diseñan los elementos artísticos respecto de un concepto, estas pantallas tienen diferentes estilos, que se almacenan en las carpetas de Android como recursos.

De la misma manera que en la aplicación de textos emergentes se diseñó una pantalla de presentación para indicar al usuario la aplicación que está utilizando, los tamaños del fondo se pueden consultar en el apartado correspondiente de la aplicación de textos.

El fondo se selecciona de igual manera, colocándolo directamente sobre el Layout que representa a la pantalla del teléfono móvil (el root), teniendo en cuenta las diferentes distribuciones de pantalla (xdpi, hdpi, mdpi, etc.).

Además, debemos introducir los elementos que aparecerán sobre la interfaz de la cámara, estos elementos se diseñan en archivos XML y se incorporan de manera programática sobre la pantalla, en este caso, será el dialogo de introducir el nombre de usuario y los colores que en este se almacenan, ya que todos los demás elementos gráficos (diamantes) y colores son elementos directamente introducidos por el SDK de **DroidAR**.

Si nos fijamos en el dialogo, este debe tener un título y una descripción, que deberán ser introducidos directamente sobre el XML ya que no van a variar nunca, además deberemos tener una caja de texto para poder introducir nuestro nombre para el Hall de la fama, esto se consigue haciendo un LinearLayout vertical sobre el que introducimos cada elemento, que deberá ser de un tamaño acorde a la pantalla de ancho y de alto lo que ocupe el texto (WrapContent).

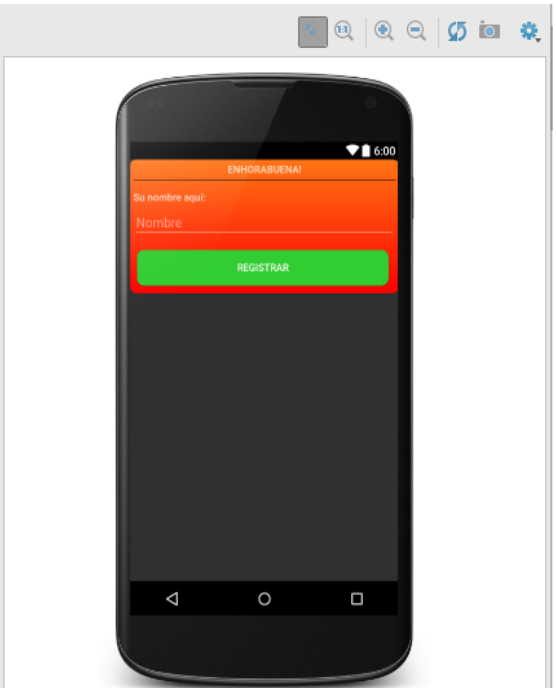
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:background="@drawable/corner_dialog">

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="ENHORABUENA!"
            android:id="@+id/title"
            android:gravity="center"
            android:layout_marginTop="5dp"
            android:textStyle="bold" />

        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:background="@android:color/black"
            android:layout_margin="5dp" />

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
```



El diseño del dialogo implica utilizar otros diseños de colores que deben ser introducidos en la carpeta drawable, como podemos ver la izquierda el fondo corner_dialog del primer Layout o el color y fondo redondeados del botón registrar. A la derecha tenemos la vista previa del dialogo tal y como se mostrará al usuario al abrir terminar el juego.

Los botones redondeados se definan en su parte en otro XML independiente, para el verde, la definición sería la que sigue **(Ver** [20]**):**

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding="10dp">
    <!-- you can use any color you want I used here gray color-->
    <solid android:color="#32CD32"/> <!-- rojo -->
    <corners android:radius="10dp"/>
</shape>
```

La forma del botón deberá por tanto ser rectangular, con un padding interno de 10 puntos, el color interno, definido por el parámetro solid deberá ser rojo oscuro, en RGB (#32CD32), como queremos que el botón sea redondeado, le introducimos un valor de radio de 10 puntos.

Por último, la forma del dialogo que utilizaremos durante aplicaciones sucesivas y que es fundamental para que este aparezca redondeado una vez que se presente por pantalla debe seguir la siguiente especificación XML. Teniendo en cuenta que el dialogo es un degradado, deberemos introducirle una forma rectangula con un angulo determinado, en este caso 90 y con dos colores uno de fin y otro de principio, naranja y rojo en nuestro caso, además deberemos introducirle un radio de 9 para los bordes.

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle" >

    <gradient
        android:angle="90"
        android:endColor="#FF7514"
        android:startColor="#FE0000"
        android:type="linear" />

    <corners
        android:radius="9dp"/>

</shape>
```

5.2.4. Implementación

Una vez que hemos descrito todos los elementos no programáticos podemos proceder con la descripción técnica del proyecto, de la misma manera que para las anteriores aplicaciones en este apartado describiremos todos los elementos programáticos y que son imprescindibles para que el sistema funcione, van desde la presentación del Splash la cual es idéntica a las anteriores y por lo cual no nos detendremos mucho, hasta el desarrollo de los puntos de intereses o los botones para desplazarse al punto.

SplashScreen

Poco que comentar de este apartado, sigue la misma lógica que en la aplicación de los textos, solo que esta vez hemos prescindido de la brújula, el tiempo de duración del minijuego debe especificarse sobre la configuración del minijuego, empleando el constructor correspondiente.

```
TimerTask task = () → {

        // Configuramos el juego para participar un solo minuto.
        ArActivity.startWithSetup(SplashActivity.this,
                new MiniGameSetup(60000));

        // Close the activity so the user won't able to go back this
        // activity pressing Back button
       SplashActivity.this.finish();
};
```

La tarea que se ejecuta cuando se termina el **Timer** es la de finalizar la actividad para que no se pueda retornar a ella e iniciar la actividad de realidad aumentada con nuestra configuración y el tiempo que queremos que dure el minijuego en milisegundos, en nuestro caso, 60 segundos o un minuto, por lo que le ponemos **60000 milisegundos**.

InfoDialog

Esta clase permite la implementación del dialogo informativo en código, este dialogo debe aparecer cuando se termina la partida, para ello creamos una clase que genere el dialogo con una actividad concreta, la cual deberá finalizar cuando se registre nuestro usuario, además deberemos inflar el Layout correspondiente al dialogo personalizado. **(Véase** [21]**)**

```
customDialog = new Dialog(context);
//deshabilitamos el título por defecto
customDialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
//obligamos al usuario a pulsar los botones para cerrarlo
customDialog.setCancelable(false);
//establecemos el contenido de nuestro dialog
customDialog.setContentView(R.layout.info_dialog);

customDialog.getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));

final TextView texto = (TextView) customDialog.findViewById(R.id.nombre);
```

La vista se carga mediante el uso de setContentView, tras realizar esta carga deberemos hacer referencia al cuadro de texto en el que se va introducir el nombre del participante, para ello deberemos recoger el elemento por el id que se le dio en el diseño artístico (R.id.nombre), posteriormente también deberemos hacer referencia al botón para añadirle la acción correspondiente a guardar la puntuación en base de datos.

Para crear el dialogo solo tenemos que introducirle una actividad como parámetro en el constructor y posteriormente llamar al método mostrar con la puntuación obtenida. El constructor por otro lado deberá crear una base de datos de puntuaciones con el contexto de la aplicación como referencia para guardar el nombre del paquete.

```
/**
 * Constructor de la clase, para incializar el contexto.
 * @param context El contexto de la app.
 */
public InfoDialog(Activity context) {
    this.context = context;
    db = new PuntuationDataHelper(context,"DBGame",null,1);
}

/**
 * Muestra la informacion en el dialogo.
 * @param puntuacion La puntuacion final.
 */
public void mostrar(final int puntuacion) {
```

Por último, deberemos describir la acción correspondiente al botón, este botón lo que hace es comprobar si el nombre ha sido introducido, si es así, guarda la puntuación en base de datos, informa al usuario y cierra la ventana, de lo contrario informa al usuario para que introduzca un nombre que será guardado en la base de datos junto con su puntuación.

```
// Añadimos una accion al boton
((Button)
customDialog.findViewById(R.id.online)).setOnClickListener(new
View.OnClickListener() {

    @Override
    public void onClick(View view) {
        // SI el texto es vacio se notifica
        if(texto.getText().toString().isEmpty()){
            new CommandShowToast(context,"Introduzca un nombre para
el hall de la fama").execute();
        }else{
            // Si no se añade puntuacion a la base de datos y
cerramos la app.
            SQLiteDatabase dbo = db.getWritableDatabase();
            dbo.execSQL("INSERT INTO Puntuation (puntos, nombre)" +
                    "VALUES
("+puntuacion+",\""+texto.getText().toString()+"\");");
            customDialog.dismiss();
            new CommandShowToast(context,"Gracias por jugar " +
texto.getText().toString() + ", " +
                    "tu puntuación " + puntuacion + " sera
almacenada.").execute();
            context.finish();
        }
    }
});
```

PuntuationDataHelper

El funcionamiento de esta clase es idéntico a la del texto, en este caso cambiamos los parámetros para recibir un texto que indica el nombre de usuario y un integer que se trata de la puntuación que se ha obtenido finalmente.

```
/**
 * Created by Lea on 04/09/2016.
 * Esta clase permite manejar una base de datos simple de puntuaciones
 */
public class PuntuationDataHelper extends SQLiteOpenHelper {

    String sqlCreate = "CREATE TABLE Puntuation (puntos INTEGER, nombre TEXT)";

    public PuntuationDataHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) { db.execSQL(sqlCreate); }


    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        //NOTA: Por simplicidad del ejemplo aquí utilizamos directamente la opción de
        //      eliminar la tabla anterior y crearla de nuevo vacía con el nuevo formato.
        //      Sin embargo lo normal será que haya que migrar datos de la tabla antigua
        //      a la nueva, por lo que este método debería ser más elaborado.

        //Se elimina la versión anterior de la tabla
        db.execSQL("DROP TABLE IF EXISTS Puntuation");

        //Se crea la nueva versión de la tabla
        db.execSQL(sqlCreate);
    }
}
```

MiniGameSetup

Este Setup no tiene que esperar por la localización del usuario ya que no la utilizamos directamente para actualizar la posición de los diamantes, si no que utilizamos la de la cámara, que se sitúa sobre una posición determinada cuando se inicia, posteriormente utilizamos la posición relativa de los diamantes para cambiarlos de posición.

```
/**
 * El constructor principal de la clase.
 * @param milis Los milisegundos que durará el juego.
 */
public MiniGameSetup(long milis) { this.milis = milis; }
```

Esta configuración tiene que heredar de la clase **Setup** de la que hablaremos posteriormente en la **aplicación de información emergente**, generalmente estas configuraciones tienen unos métodos que deben ser implementados para que funcione correctamente el diseño de la aplicación, esto permite renderizar objetos en el mundo.

Esto se consigue en el método **addWorldsToRender(),** en el cual deberemos añadir todos los objetos que sean necesarias al mundo y posteriormente renderizarlo, para ello generamos un diamante por cada color mediante la función **addDiamon(Color)**.

```
@Override
/**
 * Esta funcion permite renderizar el mundo, debemos inicializar en
esta clase la camara y el mundo.
 * @author Javier Palomares
 * @param glRenderer
 *            here you should add your world(s)
 * @param objectFactory
 *            you could get this object your self wherever you want
by
 *            getting the singleton-instance of {@link GLFactory}
 * @param currentPosition
 *           La posición actual para posicionar objetos en el mapa.
 */
public void _b_addWorldsToRenderer(GL1Renderer renderer,
      GLFactory objectFactory, GeoObj currentPosition) {

   // Colocamos la camara a una altitud de 10, para que sea mas
facil capturar los diamantes.
   camera = new GLCamera(new Vec(0, 0, 10));
   world = new World(camera);

   // Añadimos los diamantes al mapa.
   addDiamond(Color.blue());
   addDiamond(Color.green());
   addDiamond(Color.red());

   renderer.addRenderElement(world);
}
```

Arriba se muestra como se crea la cámara y el mundo, que perfectamente como hemos visto en otras ocasiones pueden inicializare en el método **initFieldsIfNecessary** que veremos más adelante, además generamos un diamante de cada color con su parámetro correspondiente y renderizamos el mundo.

Los diamantes por su parte se generan mediante la función **addDiamond** de la cual hemos implementado dos, utilizando una de ella únicamente, aunque debido a la necesidad de reutilización se pensó en la necesidad de que cada diamante tuviese su propio comando por lo cual se implementó ese segundo método.

La estructura de ambas es muy similar cambiando la cabecera de la función y que en vez de crear el comando directamente en código se le coloca el que se recibe como parámetro.

```
/**
 * La siguiente funcion permite añadir un diamante con efecto pulso.
 * @param color El color especifico del diamante.
    */
private void addDiamond(Color color)
```

```
/**
 * Permite añadir un diamante con un color y comando especifico.
 * @param color El color del diamante
 * @param com El comando para el diamante.
    */
private void addDiamond(Color color, Command com)
```

Si nos fijamos más detenidamente en el método correspondiente al del color, debemos crear primero un diamante de un color (parámetro) y a una altura en el eje Z de cuatro puntos, escalando el objeto a un punto en cada eje y con una animación de pulso.

Posteriormente agregamos este ítem a una forma y ponemos un comando cuando se le pulsa, el cual aumenta la puntuación según el color.

```
/**
 * La siguiente funcion permite añadir un diamante con efecto pulso.
 * @param color El color especifico del diamante.
   */
private void addDiamond(Color color) {
   // Creamos el diamante con un color, para el primer grupo
   final MeshComponent arrow =
GLFactory.getInstance().newDiamond(color);
   arrow.setPosition(new Vec(0,0,4));
   arrow.setScale(new Vec(1,1,1));
   arrow.addAnimation(new AnimationPulse(2, new Vec(0, 0, 0),new
Vec(1,1,1), 0.2f));

   // Creamos una nueva figura que contenga al diamante parpadeante.
   final MeshComponent itemMesh = new Shape();
   itemMesh.addChild(arrow);
   // Le ponemos una posicion aleatoria alredor de la camara.

itemMesh.setPosition(Vec.getNewRandomPosInXYPlane(camera.getPosition
(),
         1, 5));

   // Le añadimos el comando.
   itemMesh.setOnClickCommand(new Command() {
      @Override
      public boolean execute() {
         // Si el color es azul, 20 puntos y cambiamos a verde.
         if (arrow.getColor().equals(Color.blue())) {
            arrow.setColor(Color.green());
            puntuation+=20;
         // Si el color es verde, 50 puntos y cambiamos a rojo.
         }else if(arrow.getColor().equals(Color.green())){
            arrow.setColor(Color.red());
            puntuation+=50;
         // Si el color es rojo, 100 puntos y cambiamos a azul.
         }else if(arrow.getColor().equals(Color.red())){
            arrow.setColor(Color.blue());
            puntuation+=100;
         }
         // Cambiamos la posición entre 0.5 y 2 puntos, desde donde
estaba.

itemMesh.setPosition(Vec.getNewRandomPosInXYZ(itemMesh.getPosition()
,.5f,2));
         return false;
      }
   });

   // Añadimos el item al mundo.
   world.add(itemMesh);
}
```

Por otro lado, los elementos que van a ser utilizados durante el desarrollo de la aplicación deben ser inicializados en la función **initFieldIfNecessary**, de la misma manera que se hacía en aplicaciones anteriores. En este caso deberemos el dialogo informativo para mostrarlo posteriormente, el temporizador **(Ver** [22]**)** con los milisegundos y el texto indicativo del tiempo.

```
@Override
/**
 * En este metodo deberemos inicializar todos los campos necesarios que no se hayan inicializado en constructor.
 */
public void _a_initFieldsIfNecessary() {
    // Creamos el dialogo informativo, el textView y el contador.
    info = new InfoDialog(getActivity());
    tiempo = new TextView(getActivity());
    // El contador disminuye cada segundo 1000 milisegundos, y tiene milis de tiempo.
    new CountDownTimer(milis, 1000) {
        public void onTick(long millisUntilFinished) {
            tiempo.setText("Segundos restantes: " + millisUntilFinished / 1000);
        }

        public void onFinish() {
            // Cuando se acaba el temporizador paramos el juego y pedimos el nombre.
            info.mostrar(puntuation);
        }

    }.start(); // Inicializamos el temporizador.
}
```

Además, el texto que habíamos inicializado debemos añadirlo a la cámara mediante **addElementsToGuiSetup**, cambiando el posicionamiento de esta vista a la derecha y en la parte superior de la misma.

```
@Override
/**
 * En este metodo se tiene que añadir la brujula y el texto de la localización.
 * @param guiSetup variable que permite añadir objetos Android al mundo.
 * @param activity this is the same activity you can get with
 *            {@link Setup#myTargetActivity} but its easier to access this
 */
public void _e2_addElementsToGuiSetup(GuiSetup guiSetup, Activity context) {
    guiSetup.setTopViewAllignRight();
    guiSetup.addViewToTop(tiempo);
}
```

Por último, las acciones presentes en la cámara deben implementarse en el método **addActionsToEvents**, de manera que la cámara se mueva junto con el objetivo.

```
@Override
/**
 * En este metodo debemos inicializar las acciones como los toques sobre el texto o la pantalla.
 * La funciones de movimiento de camara son siempre las mismas.
 * @param eventManager
 *            El gestor de los eventos.
 * @param arView
 *            La vista a la que añadimos los eventos.
 * @param updater
 *            Por si hay que actualizar alguna posicion.
 */
public void _c_addActionsToEvents(EventManager eventManager,
        CustomGLSurfaceView arView, SystemUpdater updater) {
    ActionMoveCameraBuffered move = new ActionMoveCameraBuffered(camera, 5,
            25);
    // Eliminado para hacer pruebas, esto permite mover la camara con el dedo.
    // arView.addOnTouchMoveAction(move);
    eventManager.addOnTrackballAction(move);
    Action rot = new ActionRotateCameraBuffered(camera);
    updater.addObjectToUpdateCycle(rot);
    eventManager.addOnOrientationChangedAction(rot);
    eventManager.addOnLocationChangedAction(new ActionCalcRelativePos(
            world, camera));
}
```

La ventana informativa por otro lado es idéntica a la de la aplicación de texto, aparecerá al principio del todo para informar al usuario de los pasos que este debe seguir para que la aplicación funcione correctamente.

```
@Override
/**
 * Mediante esta funcion presentamos al usuario un pequeño display con informacion.
 * @param infoScreenData
 *          La pantalla que contiene los datos.
 */
public void _f_addInfoScreen(InfoScreenSettings infoScreenData) {
    infoScreenData
            .addText("MiniGame Setup!");
    infoScreenData
            .addTextWithIcon(
                    R.drawable.infoboxblue,
                    "Van a aparecer diamantes cerca de tu posicion, pusla sobre estos para ganar puntos! Suerte!");
}
```

5.3. Prototipo 3: Aplicación información emergente

En este apartado se van a describir las restricciones y funcionalidades correspondientes a la aplicación de información emergente, además vamos a hablar de la implementación del sistema de información emergente, de la misma manera que para las previas aplicaciones dividiremos el desarrollo en diseño de la aplicación, en la cual se incluirían las imágenes de la pantalla de presentación, los colores, tipos de letra, imágenes de la brújula etc., y por otro lado la parte programática que es la que se encarga de realizar las acciones.

5.3.1. Funcionalidades

Las funcionalidades para la aplicación de textos van definidas de acuerdo a la siguiente tabla:

Funcionalidad	Explicación
Figuras en puntos de interés	El usuario visualizará en puntos concretos del globo, representados por latitud y longitud los elementos que representarán puntos de interés, los elementos podrán ser: • Figuras geométricas. • Botones flotantes. • Imágenes flotantes.
Acción al pulsar sobre puntos	Si el usuario pulsa sobre la cámara en el punto de interés se debe lanzar una acción que dependerá del mismo: • Pop-up informativo • Presentación de Web • Información por pantalla
Botones posición	A modo de demostración se va a falsear la posición actual de la cámara empleando botones que cambiaran la cámara de sitio para localizar los puntos de interés.
Posicionamiento con brújula	La aplicación debe ser capaz de localizar la orientación del dispositivo, alternativamente se presentan la siguiente característica: • La imagen de la brújula debe ser personalizable.
Botones de vista pájaro	Al usuario se le deben presentar unos botones que le permitan reducir o aumentar la vista en el eje Z.
Pantalla de presentación	Pequeño SplashScreen a modo de presentación de la aplicación.
Presentación de la cámara	El software debe ser capaz de presentar la cámara del móvil para representar los objetos en el espacio de la misma.

5.3.2. Requisitos

Como se comentaba en el apartado anterior (**Funcionalidades)** vamos a basarnos en estas funcionalidades solicitadas para evitar que los requisitos hayan sido mal capturados, de esta manera no aseguramos que la fase de revisión todo este en su sitio.

Los requerimientos se establecerán de acuerdo a las tablas expuestas a continuación:

Requisitos funcionales

Número del requisito	RF1.1
Nombre del requisito	Presentación de cámara
Prioridad	Alta
Descripción	La aplicación debe ser capaz de presentar al usuario la vista actual de su cámara para evaluar los posibles objetos en el espacio.

Número del requisito	RF2.1
Nombre del requisito	Pantalla de presentación
Prioridad	Media
Descripción	La aplicación debe mostrar una pequeña pantalla de presentación (Splash)

Número del requisito	RF3.1
Nombre del requisito	Orientación brújula
Prioridad	Alta
Descripción	En todo momento debe estar presente una brújula que sea capaz de orientar en los puntos cardinales al usuario.

Número del requisito	RF4.1
Nombre del requisito	Posicionamiento en base de datos
Prioridad	Media/Baja
Descripción	Cada vez que el usuario pulse sobre el texto flotante se debe almacenar en la base de datos la hora, la posición GPS del mismo y el cual fue el elemento que pulsó.

Número del requisito	RF5.1
Nombre del requisito	Puntos de interés
Prioridad	Alta
Descripción	Al usuario se le deben presentar ciertos elementos sobre la cámara al alcanzar cierta posición GPS.

Número del requisito	RF6.1
Nombre del requisito	Botones posición
Prioridad	Alta
Descripción	Se deben presentar unos botones que devuelven al usuario a la posición que desee, para así alcanzar los puntos de interés.

Número del requisito	RF8.1
Nombre del requisito	Acción al pulsar
Prioridad	Alta
Descripción	Para la programación de los objetos se debe hacer diferentes cosas, como abrir el navegador o mostrar un popup.

Requisitos no funcionales

Los requisitos no funcionales de esta aplicación serán todos muy similares, ya que las restricciones no funcionales son idénticas en este tipo de aplicaciones, y dependen mucho de las funcionalidades que presentes.

Número del requisito	RNF1.1
Nombre del requisito	Base de datos
Prioridad	Media/Baja
Descripción	La base de datos elegida para almacenar los datos será SQLite.

Número del requisito	RNF2.1
Nombre del requisito	Sistema operativo
Prioridad	Alta/Imprescindible
Descripción	El sistema operativo elegido para el desarrollo será Android.

Número del requisito	RNF3.1
Nombre del requisito	GPS Móvil
Prioridad	Imprescindible
Descripción	Para que la aplicación funcione correctamente el móvil debe tener los sistemas GPS activos.

Número del requisito	RN4.1
Nombre del requisito	Espacio de almacenamiento
Prioridad	Alta
Descripción	La aplicación desarrollada no debe ocupar más de 20 Mb en el disco del móvil.

5.3.3. Diseño artístico

De la misma manera que las anteriores aplicaciones para cada pantalla se diseñó un concepto, cada una de las pantallas tiene diferentes estilos, que deben ser introducidos en la carpeta de recursos de Android.

Como en todas las aplicaciones anteriores tenemos una pantalla de presentación que determina cual modelo estamos ejecutando, el diseño es el mismo que las anteriores, mirar el apartado artístico de la aplicación de texto, los tamaños de los fondos no varían ya que son un estándar en Android.

El fondo del SplashScreen se selecciona de la misma manera que anteriormente, directamente sobre el layout como background, para ello deberemos introducir los respectivos fondos en sus carpetas de recursos por pantallas (xdpi, dpi, xxdpi, …).

Por otro lado, volvemos a los elementos que se deben introducir directamente sobre la interfaz que expone en SDK **DroidAR**, esto se hace de manera programática y en este caso solo serán botones que nos "teletrasporten" a la localización del objeto, además de las imágenes que se presentarán según el elemento al que nos acerquemos, tenemos el museo, la iglesia o el reloj (representando la puerta del sol).

Es decir, los elementos que se han tenido que programar son los siguientes:

- Brújula: La imagen de la brújula no ha variado respecto de otros proyectos, pero está presente como uno de los elementos del sistema y por tanto hay que mencionarla.

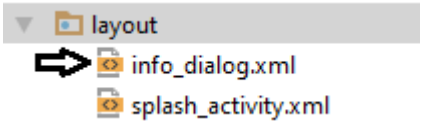
- Iglesia: Se trata de la imagen que se ha empleado para representar la iglesia del pueblo de Torrejón de Ardoz, la imagen se representa como un elemento más de la realidad aumentada, el aspecto final es el siguiente.

- Museo: De la misma manera que la anterior, en este caso la imagen es de un museo, y es representativo de la casa de Cervantes, la imagen es también un elemento de realidad aumentada, su aspecto es el siguiente.

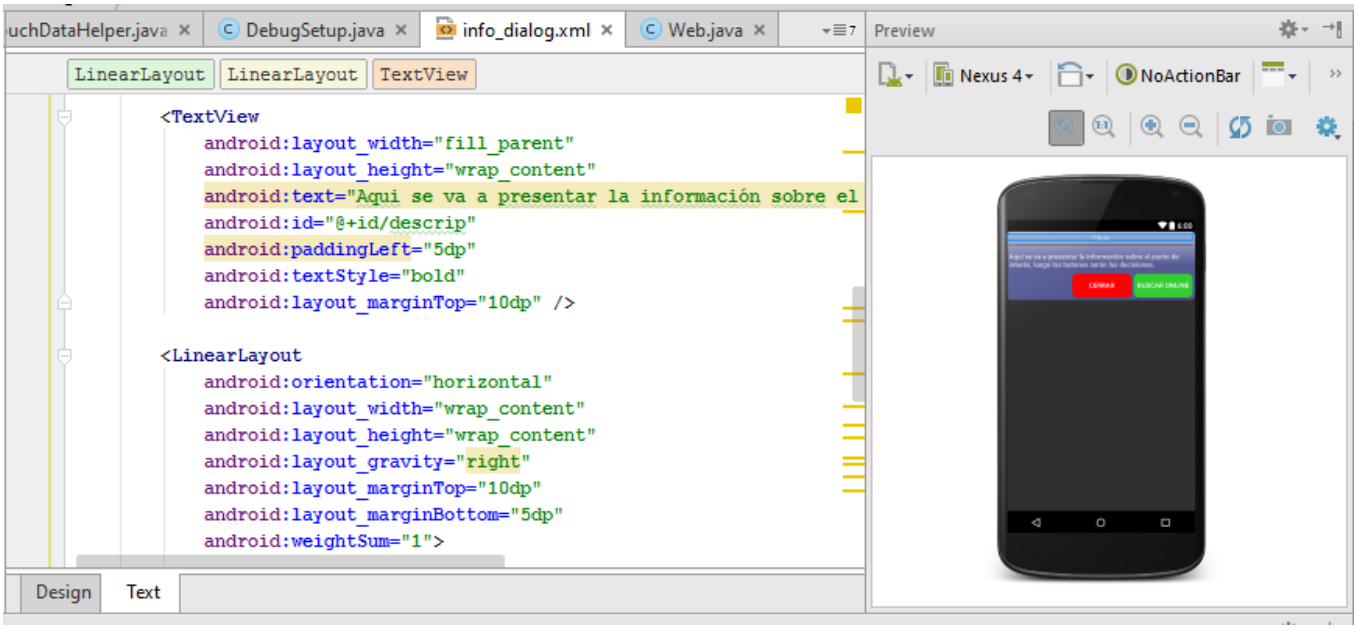
- Reloj: El reloj, representativo de la puerta del sol, igual que las anteriores se debe introducir de manera programática (como explicaremos a continuación). Su aspecto final es el siguiente.

Como se ha comentado con anterioridad, todos estos elementos deben estar almacenados en la carpeta de recursos del proyecto, concretamente en la de los "dibujables" o drawables, al ser iconos de un tamaño único no es necesario distribuirlos en carpetas de tamaño, aunque sí que deben seguir un estándar de Android.

Por último, los diseños de los diálogos emergentes y los botones del mismo, estos elementos se diseñan directamente con valores XML sobre la interfaz, para ello deberemos definir una plantilla para el dialogo emergente que deberá almacenarse en la carpeta layout de los recursos de Android.



Los componentes que forman parte de esta plantilla son dos capas lineares en formato vertical, la primera de ellas representa a la pantalla la segunda al dialogo, dentro la segunda deberemos introducir todos los elementos que deseemos, por ejemplo, el título, el divisor, la descripción y los dos botones, de los que hablaremos más adelante.



A la izquierda podemos ver los valores XML y la derecha el resultado final, como podemos ver los textos son componentes internos de Android (TextView), el divisor es una vista personalizadas con un tamaño 5 de grosor (View), los botones son Button con background definido por un XML como comentaremos a continuación, el fondo del dialogo y los bordes redondeados también son otro XML.

Los botones redondeados se definan en su parte en otro XML independiente, uno por cada botón para permitir el cambio de color, entre el rojo y el verde, la definición es la que sigue:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding="10dp">
    <!-- you can use any color you want I used here gray color-->
    <solid android:color="#FE0000"/> <!-- rojo -->
    <corners android:radius="10dp"/>
</shape>
```

La forma del botón deberá por tanto ser rectangular, con un padding interno de 10 puntos, el color interno, definido por el parámetro solid deberá ser rojo oscuro, en RGB (#FE0000), como queremos que el botón sea redondeado, le introducimos un valor de radio de 10 puntos.

La implementación para el botón verde es idéntica y sigue la siguiente estructura:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding="10dp">
    <!-- you can use any color you want I used here gray color-->
    <solid android:color="#32CD32"/> <!-- verde -->
    <corners android:radius="10dp"/>
</shape>
```

Por otro lado, también es necesaria la implementación de la lista emergente con datos relativos al museo de cervantes, esto se consigue mediante un XML de datos, la implementación normal es una capa del teléfono junto con una ListView de Android, los parámetros son lo de menos.

Para poder implementar la lista necesitamos un modelo de celda al que le almacenaremos posteriormente los datos, para ello debemos definir otro Layout independiente que rellenaremos posteriormente.

Este es el XML **(Véase** [23]**):**

```
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
   <ImageView
          android:id="@+id/photo"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
      android:paddingRight="15dp"
      android:paddingTop="10dp" />
   <TextView android:id="@+id/name"
      android:textSize="14sp"
      android:textStyle="bold"
      android:textColor="#FFFF00"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_toRightOf="@id/photo"/>
   <TextView android:id="@+id/itemDescription"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@id/name"
      android:layout_toRightOf="@id/photo" />
   <TextView android:id="@+id/price"
      android:textSize="19sp"
      android:textStyle="bold"
      android:textColor="#32cd32"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@id/itemDescription"
      android:layout_toRightOf="@id/photo" />
</RelativeLayout>
```

Para terminar con la implementación estética de la aplicación el dialogo redondeado sigue una estructura similar a la de los botones, si bien, en vez de un color solido se le introduce un color de degradado esta vez en tres tonos, pero siguiendo la estética y lógica de apartados anteriores.

El ángulo debe ser de 90 grados para que el degradado se reparta por toda la superficie, además se introducen las esquinas redondeadas con el habitual parámetro radius.

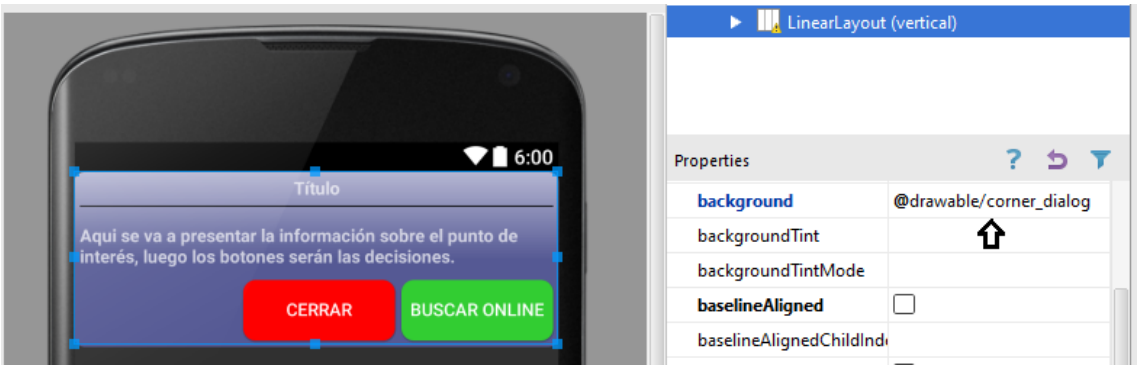
```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle" >

    <gradient
        android:angle="90"
        android:centerColor="#555994"
        android:endColor="#b5b6d2"
        android:startColor="#555994"
        android:type="linear" />

    <corners
        android:radius="9dp"/>

</shape>
```

Finalmente, de la misma manera que a los fondos del Splash, se le debe agregar a la capa correspondiente al dialogo (Linear Layout 2) el fondo del dialogo, como a continuación.



5.3.4. Implementación

Una vez que hemos descrito todos los elementos no programáticos podemos proceder con la descripción técnica del proyecto, de la misma manera que para las anteriores aplicaciones en este apartado describiremos todos los elementos programáticos y que son imprescindibles para que el sistema funcione, van desde la presentación del Splash la cual es idéntica a las anteriores y por lo cual no nos detendremos mucho, hasta el desarrollo de los puntos de intereses o los botones para desplazarse al punto.

SplashScreen

El SplashScreen no tiene variación respecto de las anteriores aplicaciones, se trata de una pequeña pantalla introductoria que indica el nombre de la aplicación, el tiempo de espera es idéntico, de cinco segundos.

```
TimerTask task = () → {

        Compass comp = new Compass(getApplicationContext(),arrow);
        comp.start();
        ArActivity.startWithSetup(SplashActivity.this, new PointsOfInterestSetup(comp));

        // Close the activity so the user won't able to go back this
        // activity pressing Back button
       SplashActivity.this.finish();
};
```

La tarea que se ejecuta cuando se termina el Timer es la de finalizar la actividad para que no se pueda retornar a ella e iniciar una nueva actividad con ArActivity y el Setup que hemos programado para los puntos de interés, este recibe una brújula por parámetro.

Para variar en la manera de programar el Splash esta vez iniciamos y creamos la brújula una vez finaliza el timer para ahorrar batería en el momento en que no se esté usando.

Compass

La clase relativa a la brújula es idéntica a la de las aplicaciones anteriores y no tienen ninguna variación, sus métodos principales son start y stop que permiten el ahorro de batería, además de implementar el SensorListener.

TouchDataBaseHelper

El funcionamiento de esta clase es idéntico a las anteriores aplicaciones, en este caso además introducimos un nuevo parámetro a la base de datos para indicar a qué hora se ha realizado el toque sobre uno de los elementos de la realidad aumentada.

Para una mejor gestión a posteriori debería cambiarse el campo hora de text a timestamp o algo similar, ya que las consultas complejas de la base de datos no permiten comparar fechas como texto, pero para simplificarlo, el campo actualmente es un texto.

```
public class TouchDataHelper extends SQLiteOpenHelper {

    String sqlCreate = "CREATE TABLE Location (codigo INTEGER, lugar TEXT, lat LONG, lon LONG, hora TEXT)";

    public TouchDataHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) { db.execSQL(sqlCreate); }


    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        //NOTA: Por simplicidad del ejemplo aquí utilizamos directamente la opción de
        //      eliminar la tabla anterior y crearla de nuevo vacía con el nuevo formato.
        //      Sin embargo lo normal será que haya que migrar datos de la tabla antigua
        //      a la nueva, por lo que este método debería ser más elaborado.

        //Se elimina la versión anterior de la tabla
        db.execSQL("DROP TABLE IF EXISTS Location");

        //Se crea la nueva versión de la tabla
        db.execSQL(sqlCreate);
    }
```

De la misma manera que las anteriores veces, los dos métodos que se emplean son onUpdate y onCreate, permiten crear una base de datos por primera vez y actualizarla entre versiones.

PointsOfInterestSetup

Esta es la clase que implementa los métodos de las configuraciones, de la misma manera que las anteriores, para ello introducimos un constructor al que le pasaremos los datos de la brújula y el cual inicializará los puntos de interés.

```
/**
 * Constructor por defecto de la clase, genera 3 puntos.
 * @param compass La brujula.
 */
public PointsOfInterestSetup(Compass compass) {
    // Iglesia torrejon
    posA = new GeoObj(40.459085, -3.5488911);
    // Casa natal cervantes.
    posB = new GeoObj(40.48223, -3.4371712,12);
    // Puerta del sol.
    posC = new GeoObj(40.4169262,-3.7735684,12);
    image = compass.getArrowView();
}
```

Para la implementación de estos puntos de interés no es necesario esperar por la localización GPS, aunque sí que sería recomendable, por lo que extenderemos directamente de la clase Setup utilizando los métodos de renderizado y actualización.

En este caso, el más importante es el de renderizado ya que vamos a implementar los 3 puntos de intereses y en ese momento le añadiremos las acciones al pulsar sobre estos.

Esto se consigue en el método **addWorldsToRender(),** en el cual deberemos añadir todos los objetos que sean necesarias al mundo y posteriormente renderizarlo, para ello generamos un comando para cada objeto.

```
@Override
public void _b_addWorldsToRenderer(GL1Renderer renderer,
      GLFactory objectFactory, GeoObj currentPosition) {

   // Le añadimos una acción.
   Command iglesia = new Command() {
      @Override
      public boolean execute() {
         SQLiteDatabase dbo = db.getWritableDatabase();
         Location loc = camera.getGPSLocation();
         Date fecha = new Date();
         dbo.execSQL("INSERT INTO Location
(codigo,lugar,lat,lon,hora)" +
               "VALUES (" + counter +",\""+ "Iglesia" + "\"," +
loc.getLatitude() + "," + loc.getLongitude() + ",\"" +
fecha.toString() +"\");");
         dbo.close();
         counter++;
         getActivity().runOnUiThread(new Runnable() {
            @Override
            public void run() {
               info.mostrar("La iglesia de Torrejon",
                     "La iglesia de torrejon se trata de un
monumento histórico de gran valor para el municipio.",
                     "http://www.ayto-torrejon.es/");
            }
         });
         return false;
      }
   };
```

Arriba se muestra como se implementa un comando para ponerlo en cada componente, los tres comandos se generan de la misma forma, solo que cada uno realiza una opción diferente (Pagina web, abrir lista de objetos o abrir dialogo), posteriormente se les indica una posición, el objeto, su imagen y se le aplica el comando mediante el método **spawnObj().**

```
/**
 * Se añaden objetos al mundo.
 * @param pos La posicion en latitud longitud.
 * @param com El comando que se le introduce al mesh component.
 * @param resourceId El id del recurso (imagen).
 * @param id El id de la imagen para el mesh.
 */
private void spawnObj(final GeoObj pos, Command com, int resourceId, String id) {
    // Se crea un nuevo objeto a partir de la posición.
    GeoObj x = new GeoObj(pos);

    // Creamos un conjunto de objetos con la imagen que deseamos.
    MeshComponent mesh = GLFactory.getInstance()
            .newTexturedSquare(
                    id,
                    IO.loadBitmapFromId(myTargetActivity,
                            resourceId));
    mesh.setScale(new Vec(10, 10, 10));
    // Le ponemos el comando.
    mesh.setOnClickCommand(com);

    // Se le crea una nueva posicion a partir del objeto, desplazado 5 en el eje Y.
    mesh.setPosition(new Vec(0,5,0));
    x.setComp(mesh);
    CommandShowToast.show(myTargetActivity, "Punto de interés en: "
            + x.getLatitude()+ "," + x.getLongitude());
    world.add(x);
}
```

Por otro lado, los elementos que van a ser utilizados durante el desarrollo de la aplicación deben ser inicializados en la función **initFieldIfNecessary**, de la misma manera que se hacía en aplicaciones anteriores. En este caso deberemos inicializar la base de datos y el dialogo informativo para mostrarlo posteriormente.

```
/**
 * En este metodo se inicializan las variables.
 */
@Override
public void _a_initFieldsIfNecessary() {
    camera = new GLCamera();
    world = new World(camera);
    gpsAction = new ActionCalcRelativePos(world, camera);
    info = new InfoDialog(getActivity());
    db = new TouchDataHelper(getActivity(),"DBLocation",null,1);
}
```

Además, los botones de cambio de posición GPS deben añadirse en el método **addElementsToGuiSetup**, desde el cual añadiremos los botones de la cámara (idénticos a los de aplicaciones anteriores) y los de cambio de posición GPS (Los cuales solo funcionan si no disponemos de una localización concreta), cuya acción se implementa con métodos internos del SDK **DroidAR**.

```
guiSetup.addButtonToBottomView(new Command() {

    @Override
    public boolean execute() {
        gpsAction.resetWorldZeroPositions(camera.getGPSLocation());
        return false;
    }
}, "To GPS");

guiSetup.addButtonToBottomView(new
DebugCommandPositionEvent(gpsAction,
        posA), "To pos A");
guiSetup.addButtonToBottomView(new
DebugCommandPositionEvent(gpsAction,
        posB), "To pos B");
guiSetup.addButtonToBottomView(new
DebugCommandPositionEvent(gpsAction,
        posC), "To pos C");
```

Por último, las acciones presentes en la cámara deben implementarse en el método **addActionsToEvents**, de manera que la localización de la cámara cambie con la posición GPS y que la cámara se mueva junto con el objetivo.

```
/**
 * En este metodo se añaden los eventos al mundo.
 * @param eventManager El manejador de eventos, para crear y añadir
 * @param arView La vista a la que se añaden los objetos.
 * @param updater El actualizador de los datos.
 */
@Override
public void _c_addActionsToEvents(EventManager eventManager,
        CustomGLSurfaceView arView, SystemUpdater updater) {
    /*arView.addOnTouchMoveListener(new ActionMoveCameraBuffered(camera, 5,
            25));*/

    // Añadimos todas las acciones necesarias para el cambio de localizacion y movimiento de la camara.
    ActionRotateCameraBuffered rot = new ActionRotateCameraBuffered(camera);
    updater.addObjectToUpdateCycle(rot);
    eventManager.addOnOrientationChangedAction(rot);

    eventManager.addOnTrackballAction(new ActionMoveCameraBuffered(camera,
            5, 25));
    eventManager.addOnLocationChangedAction(gpsAction);
}
```

Setup

La clase Setup es la clase del que deben heredar todas las configuraciones que queremos que emplen los métodos para presentar objetos en la cámara, es imprescindible que implementen todas las funciones que se encargan del renderizado y la carga de datos en el mapa, sin las cuales no sería posible el desarrollo de estas aplicaciones.

```
public Setup() { this(true); }

public Setup(Activity target, SetupListener listener,
        boolean useAccelAndMagnetoSensors) {
    this(useAccelAndMagnetoSensors);
    mySetupListener = listener;
}

// TODO remove boolean here and add to EventManager.setListeners..!
public Setup(boolean useAccelAndMagnetoSensors) {
    this.useAccelAndMagnetoSensors = useAccelAndMagnetoSensors;
}

/**
 * Default initialization is {@link Surface#ROTATION_90}, use landscape on
 * default mode if the initialization does not work
 *
 * @return {@link Surface#ROTATION_0} or {@link Surface#ROTATION_180} would
 *         mean portrait mode and 90 and 270 would meen landscape mode
 *         (should be the same on tablets and mobile devices
 */
public static int getScreenOrientation() {
    if (screenOrientation == null) {
        Log.e(LOG_TAG, "screenOrientation was not set! Will asume"
                + " default 90 degree rotation for screen");
        return Surface.ROTATION_90;
    }
    return screenOrientation;
}
```

Esta clase además se encarga de gestionar todos los datos relativos a la cámara y los sensores de Android que son necesarios para el funcionamiento de las aplicaciones como, por ejemplo, el magnetómetro o el acelerómetro.

Además, introduce la posibilidad de implementar un menú de opciones en la barra de navegación de Android, aunque en este caso no ha sido necesario y no se ha implementado.

```
public boolean onCreateOptionsMenu(Menu menu) {
    if (myOptionsMenuCommands != null) {
        return fillMenuWithCommandsFromCommandgroup(menu,
                myOptionsMenuCommands);
    }
    return false;
}

/*
 * is used by the GuiSetup class to add elements to the options menu
 */
public void addItemToOptionsMenu(Command menuItem, String menuItemText) {
    if (myOptionsMenuCommands == null) {
        myOptionsMenuCommands = new CommandGroup();
    }
    menuItem.getInfoObject().setShortDescr(menuItemText);
    myOptionsMenuCommands.add(menuItem);
}

public boolean onMenuItemSelected(int featureId, MenuItem item) {
    if (featureId == Window.FEATURE_OPTIONS_PANEL) {
        if (myOptionsMenuCommands != null) {
            return myOptionsMenuCommands.myList.get(item.getItemId())
                    .execute();
        }
    }
    return false;
}
```

Los métodos que se utilizan para el renderizado de los datos y que deben ser heredados en las clases hijas para las configuraciones son los siguientes:

```
public abstract void _a_initFieldsIfNecessary();
public abstract void _b_addWorldsToRenderer(GL1Renderer glRenderer,
      GLFactory objectFactory, GeoObj currentPosition);
public abstract void _c_addActionsToEvents(EventManager
eventManager,
      CustomGLSurfaceView arView, SystemUpdater updater);
public abstract void _d_addElementsToUpdateThread(SystemUpdater
updater);

public abstract void _e2_addElementsToGuiSetup(GuiSetup guiSetup,
      Activity activity);
```

Cada uno de ellos con su respectiva documentación por parte del equipo de **DroidAR.**

InfoDialog

Esta clase se trata de la implementación del dialogo informativo en código, este dialogo debe aparecer cuando se pulsa sobre la iglesia, para ello creamos una clase que genere el dialogo con un contexto de aplicación, además deberemos inflar el Layout correspondiente a nuestro dialogo.

```
customDialog = new Dialog(context);
//deshabilitamos el título por defecto
customDialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
//obligamos al usuario a pulsar los botones para cerrarlo
customDialog.setCancelable(false);
//establecemos el contenido de nuestro dialog
customDialog.setContentView(R.layout.info_dialog);

customDialog.getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));

// Referenciamos todos los contenidos.
TextView title = (TextView) customDialog.findViewById(R.id.title);
title.setText(titulo);

TextView contenido = (TextView) customDialog.findViewById(R.id.descrip);
contenido.setText(info);
```

Una vez que tenemos la vista cargada, podemos hacer referencia a los elementos que en esta se encuentran para poder modificar el título del dialogo y las acciones de los botones, para ello la función mostrar deberá recibir el título y la descripción, además de la url que queremos que se cargue cuando pulsamos sobre "*ir en línea*".

```
/**
 * Muestra la informacion en el dialogo.
 * @param titulo El titulo del dialogo
 * @param info La informacion del dialogo
 * @param url La url que se abrira.
 */
public void mostrar(String titulo, String info, final String url) {
```

Por último, deberemos describir la acción correspondiente al botón, este lo que hace es abrir una intención sobre nuestra clase Web que explicaremos a continuación para así poder abrir la página web desde la misma aplicación. Si esto falla, se abrirá desde el navegador predeterminado. Deberemos almacenar la Url como un **extra** de información en la intención.

```
((Button)
customDialog.findViewById(R.id.online)).setOnClickListener(new
View.OnClickListener() {

    @Override
    public void onClick(View view) {
        // Creamos una intención, que recibira como parametro la
url.
        Intent i = new Intent(context,Web.class);
        i.putExtra("url",url);
        // Iniciamos la actividad.
        context.startActivity(i);
        customDialog.dismiss();
    }
});
```

Web

Esta clase se trata de una actividad que tiene un XML asociado, este XML contiene un elemento WebView que permite cargar páginas web directamente sobre la aplicación y sin necesidad de acceder al navegador.

```
/**
 * Created by Lea on 07/09/2016.
 * La clase que permite visualizar la web desde la aplicación.
 */
public class Web extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.setContentView(R.layout.web_load);
        WebView wb = (WebView) findViewById(R.id.webView);
        String url;
        if(savedInstanceState != null)
            url = (String) savedInstanceState.get("url");
        else
            url = getIntent().getStringExtra("url");
        wb.loadUrl(url);
    }
}
```

Para ello se rescata la URL recibida como extra dentro del **Intent** esto se consigue o bien con el **savedInstance** (Siempre y cuando ya se haya estado en la actividad) o bien con el **Intent** si no se había visitado con anterioridad.

ListViewActivity

La siguiente clase se utiliza para mostrar información sobre el museo de cervantes, puede ser reutilizada para mostrar cualquier información, cambiando el modelo de celda explicado en el apartado de diseño artístico y modificando los datos que se almacenan en la lista para el adaptador, es decir, el método, GetInfo();

```
/**
 * Devuelve la informacion en un array para el adaptador.
 * @return EL array list con la información.
 */
private ArrayList<ItemDetails> GetInfo(){
    ArrayList<ItemDetails> results = new ArrayList<>();


    ItemDetails item_details = new ItemDetails();
    item_details.setName("Información");
    item_details.setItemDescription("El Museo Casa Natal de Cervantes");
    item_details.setPrice("Gratis");
    item_details.setImageNumber(1);
    results.add(item_details);

    item_details = new ItemDetails();
    item_details.setName("Como llegar");
    item_details.setItemDescription("C/ Mayor, 48, Alcalá de Henares");
    item_details.setPrice("");
    item_details.setImageNumber(2);
    results.add(item_details);
```

Finalmente, en el método onCreate se implementa la lista y se le añade el adaptador del que hablaremos posteriormente, luego seleccionamos las celdas que devuelven información y las celdas que te mandan a una página web mediante la variable posición.

```
// Referenciamos el listView y le añadimos un adaptador
      final ListView lv1 = (ListView) findViewById(R.id.listview);
      lv1.setAdapter(new ItemListBaseAdapter(this, image_details));

      lv1.setOnItemClickListener(new OnItemClickListener() {
       @Override
       public void onItemClick(AdapterView<?> a, View v, int
position, long id) {
          //Si pulsamos sobre web, nos la abre.
          if(position == 3){
         // Creamos una intención, que recibira la url.
         Intent i = new
Intent(ListViewImagesActivity.this,Web.class);

i.putExtra("url","http://www.museocasanataldecervantes.org/");
         // Iniciamos la actividad.
         startActivity(i);
      }else{
         Object o = lv1.getItemAtPosition(position);
         ItemDetails obj_itemDetails = (ItemDetails)o;
         Toast.makeText(ListViewImagesActivity.this, "Ha tocado
sobre: " + obj_itemDetails.getName(), Toast.LENGTH_LONG).show();
      }
   }
```

ListItem

Se trata del modelo que almacena los datos de interés para la lista desplegable de información, tiene los métodos correspondientes para cambiar los valores y para obtenerlos. Los datos son, titulo, descripción y precio (si lo hubiese).

```
/* El nombre principal. */
private String name ;
/* La descripción. */
private String itemDescription;
/* El precio si lo hubiese. */
private String price;
/* El resource id */
private int imageNumber;
```

ItemListBaseAdapter

Esta clase es la encargada de manejar los datos que contiene la lista y presentarlos en la pantalla, para cada uno de los elementos los va describiendo en función de la posición y el dato almacenado en la lista que se le pasa al adaptador, para poder rellenar estos datos utilizamos el método getView.

```
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder;
    // Para evitar que Android cargue continuamente objetos. Se almacenan estos en una
    // convertView, si este existe se recupera por su tag, si no se recarga.
    if (convertView == null) {
        convertView = l_Inflater.inflate(R.layout.item_details_view, null);
        holder = new ViewHolder();
        holder.txt_itemName = (TextView) convertView.findViewById(R.id.name);
        holder.txt_itemDescription = (TextView) convertView.findViewById(R.id.itemDescription);
        holder.txt_itemPrice = (TextView) convertView.findViewById(R.id.price);
        holder.itemImage = (ImageView) convertView.findViewById(R.id.photo);

        convertView.setTag(holder);
    } else {
        holder = (ViewHolder) convertView.getTag();
    }

    // Una vez que tenemos recogidos los campos, los rellenamos con su respectivo codigo.
    holder.txt_itemName.setText(itemDetailsrrayList.get(position).getName());
    holder.txt_itemDescription.setText(itemDetailsrrayList.get(position).getItemDescription());
    holder.txt_itemPrice.setText(itemDetailsrrayList.get(position).getPrice());
    holder.itemImage.setImageResource(imgid[itemDetailsrrayList.get(position).getImageNumber() - 1]);

    return convertView;
}
```

Cabe destacar que para esta clase debemos tener una estructura de datos Holder, además de los métodos implementados del baseAdapter.

```
/**
 * Estructura de datos que contiene la informacion de cada celda.
 */
static class ViewHolder {
   TextView txt_itemName;
   TextView txt_itemDescription;
   TextView txt_itemPrice;
   ImageView itemImage;
}
```

6. Presupuestos

El presupuesto del proyecto trata de desglosar todos los gastos materiales y de obra utilizado en el proyecto y por ello se divide en Mano de obra y material.

6.1. Coste de la mano de obra

Concepto	Horas	Coste/Hora	Coste total
INGENIERIA			
Jefe de proyecto	35	40	1400
Analista	84	20	1680
Programador	640	12	7680
SECRETARIA			
Secretaria	120	10	1200
Total			**11960**

Como se puede ver en la tabla adjunta los conceptos han sido desglosados por secciones, a continuación, explicamos que incluye cada uno.

Concepto de ingeniería:

- Diseño de la aplicación.
- Diseño de la interfaz de usuario.
- Pruebas realizadas sobre las aplicaciones.
- Escritura de la memoria.

Concepto de secretaria:

- Generar documentos e informes del proyecto.

6.2. Coste del material

Coste Software

Descripción	Licencias	Coste	TOTAL
Windows 10	1	135	135
Android Studio	1	Gratis	0
SQLite	1	Gratis	0
Total			**135**

Coste Hardware

Descripción	Cantidad	Coste	TOTAL
PC Asus	1	550	550
LG G2 Mini	1	250	120
Galaxy A5	1	250	250
Total			**920**

Total de costes

Descripción	TOTAL
Software	135
Hardware	920
Total	**1055**

6.3. Coste total del proyecto

Este apartado engloba el computo completo de todos los costos en los apartados anteriores, es la suma total de cada gasto implicado en el proyecto.

Además, a este cómputo se le sumará un 10% de la mano de obra de acuerdo a gastos en dietas, desplazamientos o material de oficina el cual denominaremos gasto genérico.

Descripción	TOTAL
Mano de obra	11960
Material	1055
Gasto genérico	1196
Total	**14211**

Por lo tanto, el coste total del proyecto asciende a **catorce mil doscientos once euros.**

7. Conclusiones y trabajos futuros

Tras desarrollar nuestras aplicaciones se debe realizar una reflexión sobre cómo se ha desarrollado el proyecto, o sobre que líneas de investigación se va a trabajar en un futuro.

7.1. Conclusiones

Con el desarrollo de estas aplicaciones podemos concluir que los objetivos principales de nuestro proyecto han sido resueltos satisfactoriamente, el SDK (Kit de desarrollo) elegido ha sido el apropiado para este tipo de cometido, ya que este se trata de un OpenSource al cual le podemos introducir cualquier mejora que creamos necesaria.

Por otro lado, la falta de documentación sobre este kit de desarrollo puede llevar a confusión al principio, además de no ofrecer todas las necesidades de este tipo de aplicaciones en la parte integrada, ya que la mayoría de las mejoras que se han ido incorporando con el tiempo se encuentran en módulos separados.

A pesar de las consideraciones anteriores, podemos concluir que no han sido utilizadas ni la mitad de las funcionalidades que expone el kit de desarrollo de DroidAR y, por lo tanto, este trabajo puede ser completado perfectamente en un futuro con mayor variedad de funcionalidades de entre todas las que dispone el SDK.

Como consideración final destacar que este tipo de kits de desarrollo de código abierto son idóneos para desarrollos de poco calibre, por ejemplo, para trabajos de investigación, pero si se quiere llegar a algo más elaborado es recomendable evaluar el resto de SDKs que proporcionan mayor variedad, aunque su versión completa sea de pago.

7.2. Trabajos futuros

Las líneas de trabajo futuras siguen el cauce del desarrollo actual y permiten completar el desarrollo de estas aplicaciones en un futuro, de acuerdo a esto podríamos definir las siguientes tareas como necesarias en cuanto a desarrollo:

- **Inclusión de otros modulos:** Muchas de estas aplicaciones utilizan funcionalidades que no están directamente en el módulo principal del proyecto DroidAR, por ejemplo, el mapa para la visualización de los elementos que se encuentran cerca o bien para guiar al usuario por la ciudad en la que se encuentra.

- **Versionado superior:** Durante nuestro desarrollo Android ha lanzado al mercado dos versiones superiores a las que se han empleado para testear estas aplicaciones, lo que quiere decir que algunas de las opciones ya no sean compatibles.

Respecto a la integración realizada del SDK:

- **Mejoras en el SDK:** Al tratarse de un SDK de código abierto, a este se le pueden realizar todos los cambios que se crean convenientes, como en nuestro caso los textos, por ello se podría dar una vuelta de hoja al mismo y ver que se podría mejorar.
- **Resolver Warnings Gradle:** Cuando se compila con el kit de desarrollo de DroidAR aparecen warnings de deprecación que podrían ser resolubles, aunque no afectan a la compilación.

Respecto a la base de datos sería recomendable una de las dos posibilidades:

A) **Acceder a los datos de la base de datos:** Esto requiere un rooteo del móvil o buen manejo de la consola de Android disponible en Android Studio, ya que los datos se encuentran en carpetas inaccesibles del dispositivo.

B) **Cambiar la base de datos:** Evaluar la posibilidad de migrar las conexiones a un sistema externo que no dependa del estado del dispositivo MySQL sería una buena idea, por el hecho de ser de código abierto.

Por otro lado, y siguiendo el criterio de investigación:

- **Integrar otros SDK OpenSource:** A pesar de que nuestra investigación asegura haber elegido el mejor kit de desarrollo, esta se trata de una opinión personal y puede variar respecto del desarrollador, existen otros kits de desarrollo de código abierto como **ARToolkit** que son perfectamente compatibles, aunque llevan menos tiempo.
- **Valorar la posibilidad de un SDK de pago:** Existen multitud de kits de desarrollo de pago, los cuales disponen de una versión gratuita de evaluación, que coloca una marca de agua sobre la cámara, al tratarse de código cerrado no es posible eliminarla a no ser que pagues por el uso del mismo.

8. Bibliografía

[1] «Wikipedia Realidad Aumentada,» [En línea]. Available: https://es.wikipedia.org/wiki/Realidad_aumentada.

[2] Profeta, «La biblia del programador,» [En línea]. Available: http://labibliadelprogramador.blogspot.com.es/2012/09/estructura-de-android.html.

[3] Francisco, «Android Zone,» [En línea]. Available: http://androidzone.org/2013/05/historia-de-android-la-evolucion-a-lo-largo-de-sus-versiones/.

[4] Universidad Politécnica de Valencia, «Android Cursos,» [En línea]. Available: http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/149-componentes-de-una-aplicacion.

[5] «ngb,» [En línea]. Available: http://www.ngb-tvos.cn/apihtml/guide/topics/fundamentals/tasks-and-back-stack.html.

[6] James, «Hermosa programación,» [En línea]. Available: http://www.hermosaprogramacion.com/2014/08/android-app-componentes/.

[7] «Wikipedia SQLite,» [En línea]. Available: https://es.wikipedia.org/wiki/SQLite.

[8] N. Davis, «Social Compare,» [En línea]. Available: http://socialcompare.com/es/comparison/augmented-reality-sdks.

[9] V. R. S. C. Danie Jooste, «Research Gate,» [En línea]. Available: https://www.researchgate.net/publication/283451573_Results_of_an_Evaluation_of_Augmented_Reality_Mobile_Development_Frameworks_for_Addresses_in_Augmented_Reality.

[10] «Wikipedia desarrollo cascada,» [En línea]. Available: https://es.wikipedia.org/wiki/Desarrollo_en_cascada.

[11] «Wikipedia MVP,» [En línea]. Available: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93presentador.

[12] O. Arrivi, «The art of the left foot,» [En línea]. Available: http://theartoftheleftfoot.blogspot.com.es/2010/10/el-patron-modelo-vista-presentador-mvp.html.

[13] Firefox, «Mozilla Addons,» [En línea]. Available: https://addons.mozilla.org/es/firefox/addon/sqlite-manager/.

[14] S. Heinen, «GitHub,» [En línea]. Available: https://github.com/bitstars/droidar/wiki/_pages.

[15] «StackOverflow,» [En línea]. Available: http://stackoverflow.com/questions/13024272/android-background-image-size-in-pixel-which-support-all-devices.

[16] S. XL, «Trucos Android Studio,» [En línea]. Available: http://trucosandroidstudio.blogspot.com.es/2015/03/como-cambiar-la-fuente-de-nuestra-app.html.

[17] Amatellanes, «Amatellanes Blog,» [En línea]. Available: https://amatellanes.wordpress.com/2013/08/27/android-crear-un-splash-screen-en-android/.

[18] iutinvg, «GitHub,» [En línea]. Available: https://github.com/iutinvg/compass.

[19] sgoliver, «Sgoliver Blog,» [En línea]. Available: http://www.sgoliver.net/blog/bases-de-datos-en-android-i-primeros-pasos/.

[20] «StackOverflow,» [En línea]. Available: http://es.stackoverflow.com/questions/9068/botones-redondeados-en-android.

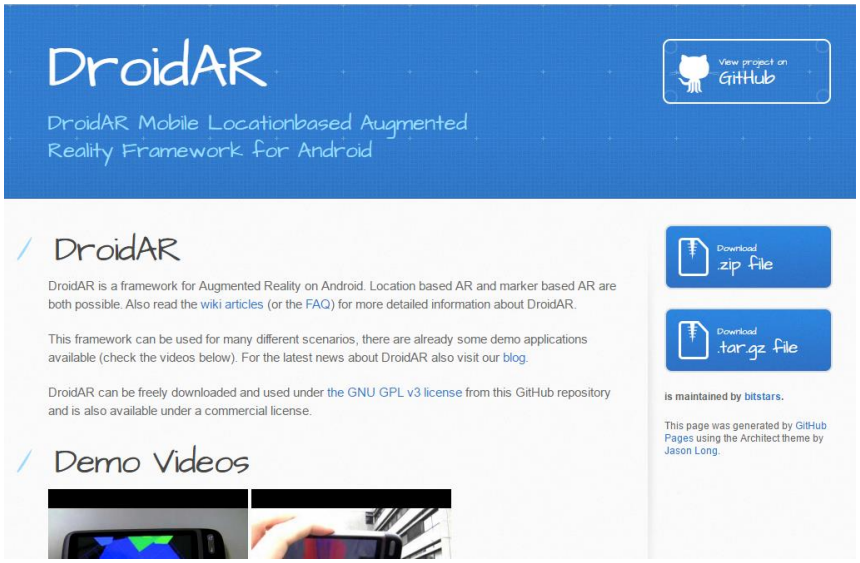
[21] Google, «Android Developers,» [En línea]. Available: https://developer.android.com/guide/topics/ui/dialogs.html.

[22] Google, «Android Developers,» [En línea]. Available: https://developer.android.com/reference/android/os/CountDownTimer.html.

[23] C. Wijesinghe, «Java Code Geeks,» [En línea]. Available: https://www.javacodegeeks.com/2012/10/android-listview-example-with-image-and.html.

Webgrafía

Destacamos de entre todas las paginas la que ha sido fundamental para el desarrollo de las aplicaciones, aquella dedicada en exclusiva al SDK y de la cual se ha extraído toda la información sobre el desarrollo de estas aplicaciones. https://bitstars.github.io/droidar/ **(Wiki en [14])**



Esta página ofrece la posibilidad de descargar el código además de los links de los videos que se han seguido para la elaboración de la aplicación, además explica un poco el funcionamiento del SDK y todas las funcionalidades de las que dispone actualmente.

9. Anexo

En este anexo se van a introducir todos aquellos puntos que no hayan sido abarcados directamente en el desarrollo de la aplicación, aunque, sin embargo, han sido necesarios para el desarrollo de la mismo, y, por tanto, es importante hacerles mención.

El anexo constará del manual de usuario, así como las instalaciones que han sido llevadas a cabo para la programación de las aplicaciones, es decir, como se integra el SDK DroidAR y como se instala Android Studio para su completo desarrollo.

I. Anexo: Manual de usuario

En este apartado vamos a realizar una pequeña introducción al funcionamiento de las aplicaciones que hemos introducido, sin entrar en mucho detalle ya que estas aplicaciones son un modelo de futuras aplicaciones del mismo tipo, se hará una pequeña referencia a la funcionalidad de cada elemento.

Además, por otro lado, permitirá analizar la utilidad de estas aplicaciones para usos futuros o bien a modo de demostración en base a las necesidades de este tipo de aplicaciones.

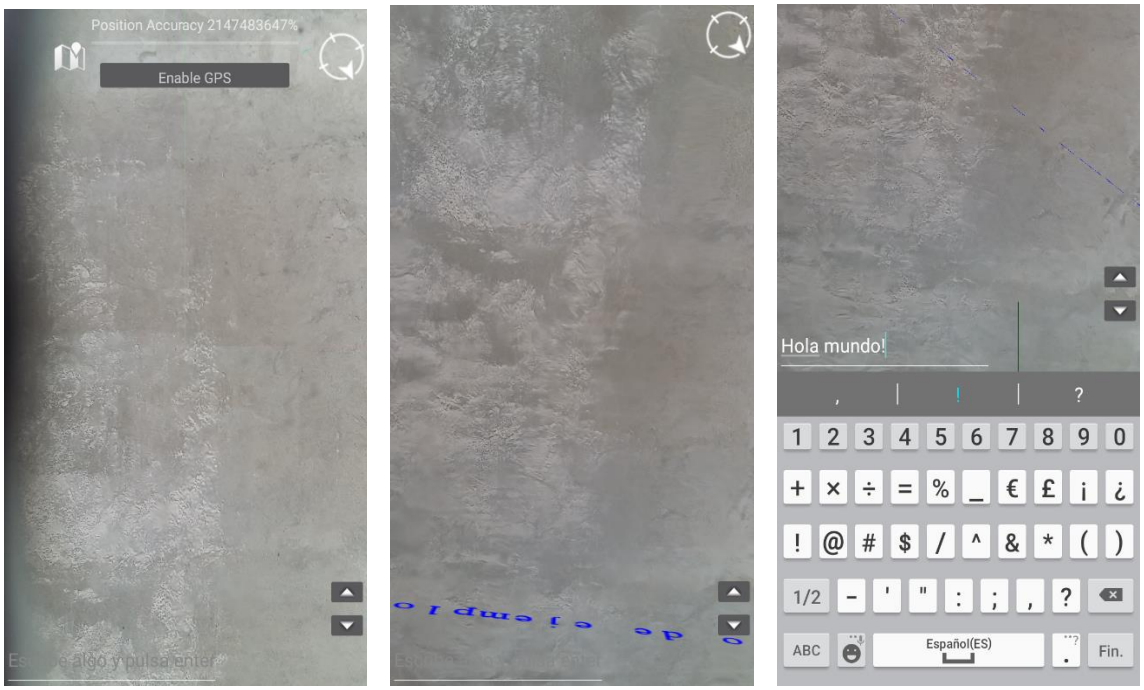
El apartado por tanto irá dividido en 3 subsecciones correspondientes a cada aplicación.

A. Aplicación textos

Como podíamos ver en el apartado correspondiente al desarrollo de la aplicación está destinada a mostrar el uso de los textos sobre la cámara, a los cuales podíamos cambiar el tamaño de letra, el color o la posición.

Esta aplicación permite mostrar fehacientemente el uso típico de la realidad aumentada para mostrar información sobre la pantalla de manera virtual, además permite introducir al usuario en el funcionamiento de la cámara, añadiendo movimiento sobre el texto al pulsar sobre el mismo.

El uso es muy sencillo, tras esperar por la localización GPS deberá aparecer frente al usuario un texto flotante en color azul, el cual, tras pulsar sobre el mismo, este aparecerá en otro punto de la cámara.

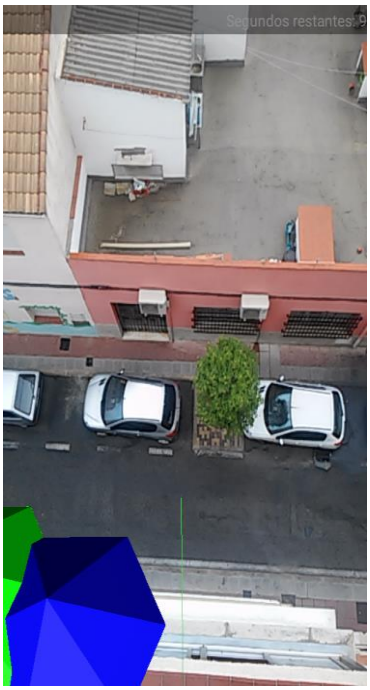


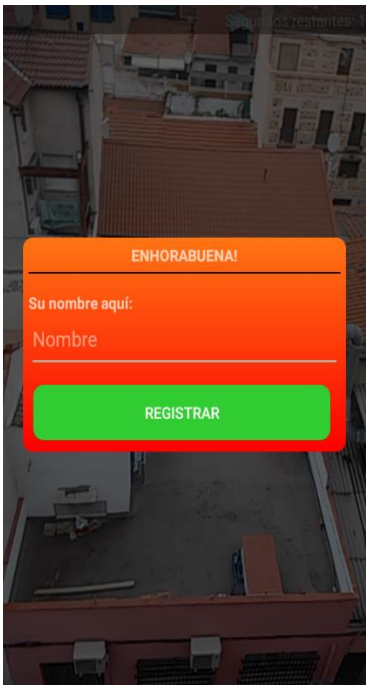
B. Aplicación minijuego

La aplicación minijuego, como podemos ver en el apartado correspondiente al desarrollo de la aplicación está destinada a mostrar la posibilidad de implementar un minijuego con realidad aumentada, además de mostrar los efectos visuales.

Al igual que los textos esta aplicación permite mostrar fehacientemente el uso similar de la realidad aumentada para implementar un minijuego sencillo con los elementos que ofrece el SDK, además permite introducir al usuario en el funcionamiento de la cámara y de la posición de los objetos respecto de un vector.

El uso de esta aplicación es muy sencillo, deberemos capturar el mayor número de diamantes posible en el tiempo que aparecerá en la parte superior, tras terminar se nos solicitará un nombre para guardarlo en la base de datos interna.





C. Aplicación información emergente

Por último, la aplicación de información emergente la cual está destinada a mostrar las múltiples posibilidades de implementación en una aplicación de Realidad aumentada, por ejemplo, para el turismo.

Como todas las demás aplicaciones permite mostrar fehacientemente el uso similar de la realidad aumentada para implementar una aplicación más elaborada, en la que se abarca desde diálogos emergentes con información hasta al desarrollo de una actividad con información en forma de lista, hasta búsqueda de información en internet.

Esta aplicación funciona de dos maneras, o bien, vamos directamente al sitio sobre el que se sitúa el punto de interés o bien desactivamos la ubicación y empleamos los botones de la parte inferior de la pantalla.

Supongamos que no disponemos de un GPS y queremos probar la aplicación, para ello solo deberemos pulsar sobre los diferentes botones, por ejemplo, el de la ubicación A que nos llevaría a la iglesia de Torrejón. Posteriormente, deberemos pulsar sobre el icono de la iglesia para poder acceder al menú emergente, si bien, la imagen capturada de directamente de la localización.

Por último, si accedemos a la posición B o bien andando hacia el punto de localización o bien con los botones, podemos abrir una lista desplegable de precios e información del museo de cervantes.

II. Anexo: Instalación Android Studio

Para el desarrollo de nuestras aplicaciones ha sido necesaria la instalación de uno de los entornos de desarrollo más conocidos en el último año para el desarrollo de aplicaciones, este entorno trata de unificar todas las herramientas de desarrollo necesarias para crear aplicaciones en el sistema operativo móvil Android.

La compañía Google decidió dejar de dar soporte a Eclipse para pasar a su propia plataforma de desarrollo invitando a los desarrolladores a la migración con su sencilla herramienta de integración de proyecto, el entorno Android Studio está basado en entornos de desarrollo de IntelliJ Idea.

La herramienta Android Studio se encuentra de manera totalmente gratuita en la página del proyecto Google Android para desarrolladores, para ello deberemos acceder al siguiente enlace: https://developer.android.com/studio/index.html

Si pulsamos sobre el botón, nos aparecerá un mensaje indicándonos la política de privacidad del proyecto, la cual deberemos aceptar para posteriormente descargar el archivo, esto nos redirigirá directamente a las instrucciones de instalación.

Una vez descargado el archivo procedemos a ejecutarlo, si no tenemos instalado el JDK de previos desarrollos en Java convendría instalarlo ya que se nos va a solicitar el directorio de java tal y como indica la documentación:

1. Descarga el JDK de Java si no lo tienes instalado **(Salta al paso 4, si ya dispones del SDK apropiado).** El SDK puede obtenerse desde la página de Oracle. http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

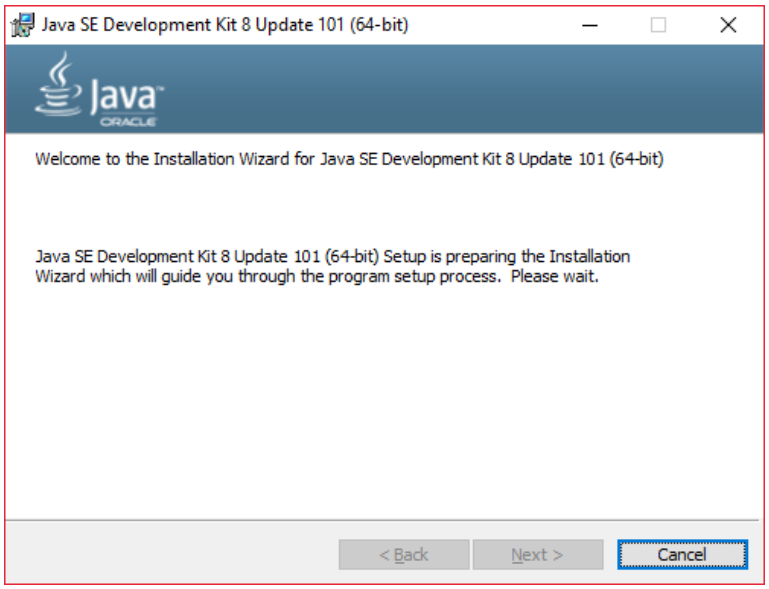
Java SE Development Kit 8u101

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.77 MB	jdk-8u101-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.72 MB	jdk-8u101-linux-arm64-vfp-hflt.tar.gz
Linux x86	160.28 MB	jdk-8u101-linux-i586.rpm
Linux x86	174.96 MB	jdk-8u101-linux-i586.tar.gz
Linux x64	158.27 MB	jdk-8u101-linux-x64.rpm
Linux x64	172.95 MB	jdk-8u101-linux-x64.tar.gz
Mac OS X	227.36 MB	jdk-8u101-macosx-x64.dmg
Solaris SPARC 64-bit	139.66 MB	jdk-8u101-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	98.96 MB	jdk-8u101-solaris-sparcv9.tar.gz
Solaris x64	140.33 MB	jdk-8u101-solaris-x64.tar.Z
Solaris x64	96.78 MB	jdk-8u101-solaris-x64.tar.gz
Windows x86	188.32 MB	jdk-8u101-windows-i586.exe
Windows x64 ←	193.68 MB	jdk-8u101-windows-x64.exe

2. Una vez descargado ejecuta el .exe descargado.

3. Sigue las indicaciones del asistente de configuración.



4. Ejecuta el archivo .exe correspondiente a Android Studio que descargaste.

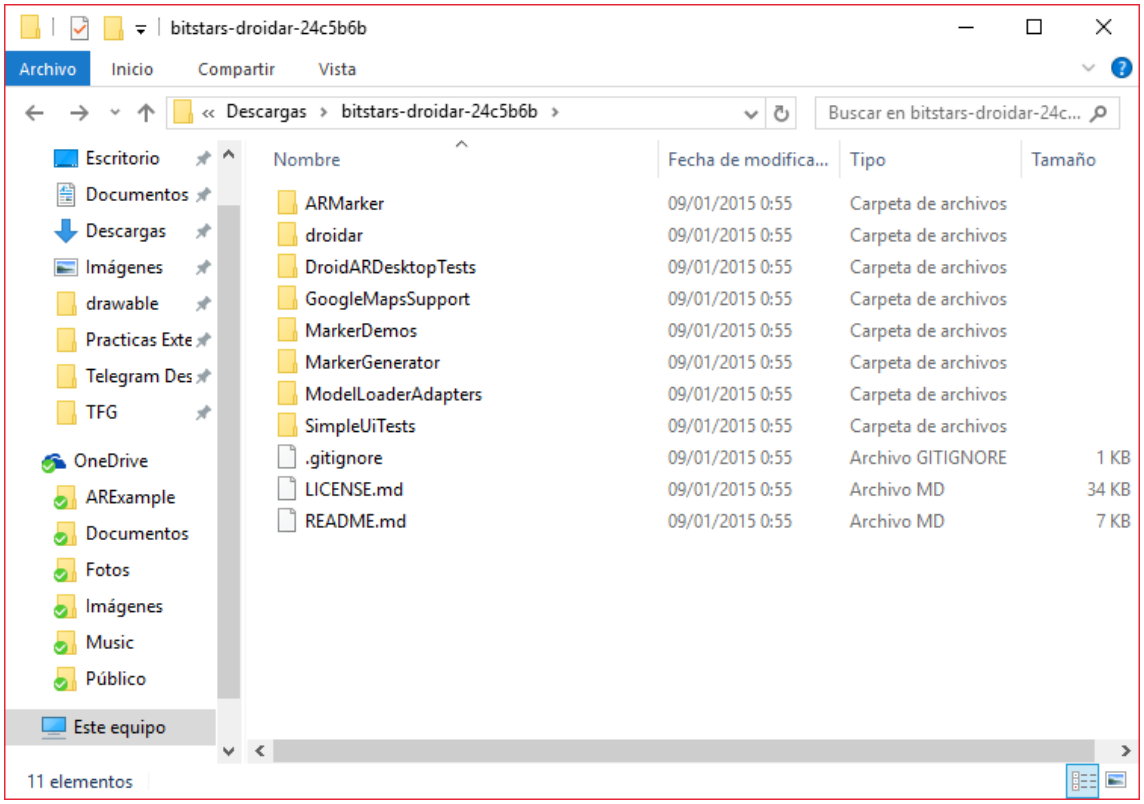
5. Sigue las indicaciones del asistente de configuración para instalar Android Studio y las herramientas de SDK necesarias, esto ocurre si no tienes una versión de Java instalada, si es así, por favor dirígete a la página de Java para descargar el JDK. Si esto ocurre por favor, comprueba que tienes descargado el SDK.

III. Anexo: Integración SDK DroidAR

A la hora de realizar el desarrollo de las aplicaciones se seleccionó un kit de desarrollo de realidad aumentada de acuerdo a los criterios expuestos en el **Estado del Arte**, este kit de desarrollo o SDK se encuentra expuesto en la página de github del proyecto, para ello deberemos descargarnos la carpeta completa desde la siguiente dirección: https://bitstars.github.io/droidar/

Como se comentaba con anterioridad este SDK de desarrollo esta íntegramente expuesto a Eclipse (entorno de desarrollo que no utilizamos en nuestro caso), por lo que es necesaria una integración con el IDE IntelliJ Idea (Android Studio). Teniendo en cuenta que Google dejo de dar soporte a Eclipse es lo mejor que podemos hacer.

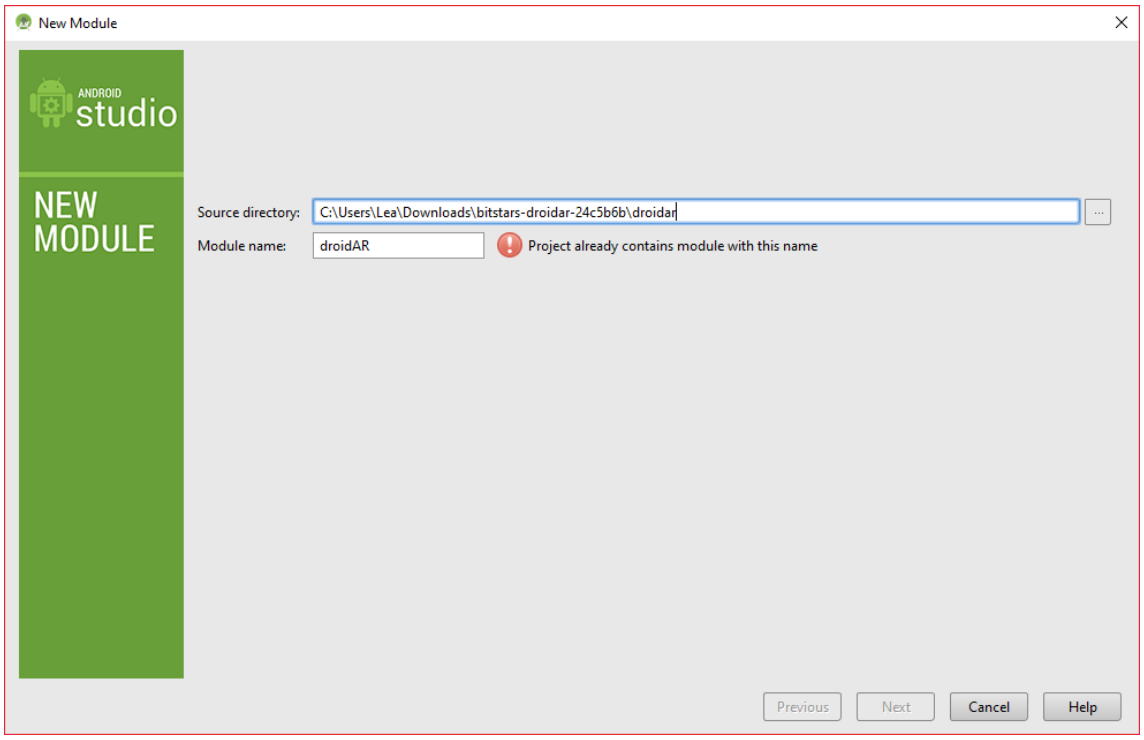
Una vez descargada la carpeta descomprimimos la misma en cualquier directorio, con lo que nos quedará una estructura como la siguiente:



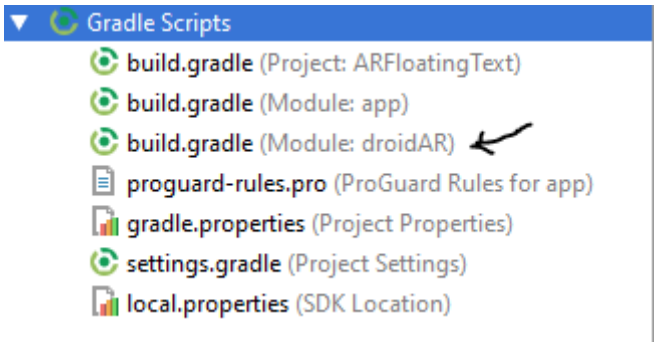
La carpeta de la que vamos a hacer uso es **DroidAR**, la cual contiene las necesidades básicas de este tipo de aplicaciones, si bien, también existen otras extensiones marcadas en las demás carpetas que están documentadas para su uso posterior y más avanzado.

Para iniciar la integración en cualquier proyecto Android Studio necesitamos haber creado uno previamente, para ello seguimos el asistente de entorno de desarrollo y creamos un proyecto vacío. File->New…->New Project->(nombre proyecto y paquete)->Empty project.

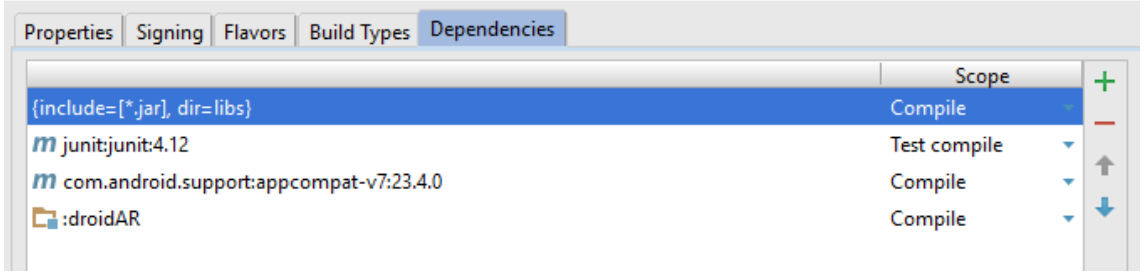
Sobre el proyecto vacío deberemos integrar el SDK DroidAR, esto se hace añadiendo el proyecto como un módulo, desde el asistente en: **File->New…->Import Module…** Donde deberemos indicarle la carpeta droidAr.



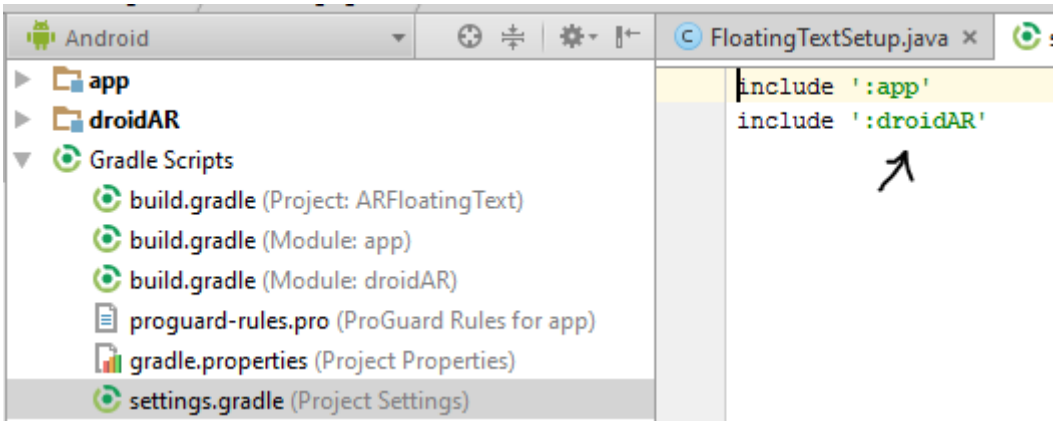
En mi caso como ya estaba importado el nombre del módulo ya existe, una vez que estemos listo pulsamos next y el modulo se añadirá a la vista del proyecto. Si todo ha ido correctamente, el modulo debería aparecer en nuestros Scripts de Gradle.



Si no es el caso, comprobamos que el modulo ha sido añadido o bien desde el asistente de modulo haciendo click sobre File->Project Structure… Donde deberá aparecer nuestro proyecto como una dependencia.



O bien en el gradle settings:



Para que todo funcione correctamente, deberemos coger el Manifest ubicado en el módulo droidAR y trasladar todas su actividades y configuraciones al Manifest del nuevo proyecto, teniendo en cuenta que la MainActivity debe cambiarse por la que nosotros queramos obviando la que viene por defecto en DroidAR.

```
<!--
     IMPORTANT: The following activities have to be added to each
project
     which imports the DroidAR Library
-->
      <activity
           android:name="system.ArActivity"
           android:configChanges="keyboardHidden|orientation"
           android:label="@string/app_name"
           android:screenOrientation="portrait">
      </activity>
```

Finalmente, también es necesario configurar el Gradle de droidAR para que este aplique el plugin de librería y no el de Application como viene por defecto.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá