



Trabajo de Fin de Grado

**ULL-AR Tecnología de realidad
aumentada en entornos
universitarios**

Alejandro Hernández Padrón

La Laguna, 5 de julio de 2019

D. **Francisco de Sande González**, con DNI nº 42067050G profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que el presente trabajo titulado:

“ULL-AR Tecnología de realidad aumentada en entornos universitarios”

ha sido realizado bajo su dirección por D. **Alejandro Hernández Padrón**, con DNI nº 42221533L

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente memoria del Trabajo Fin de Grado en La Laguna a 5 de julio de 2019

Agradecimientos

Mis agradecimientos al profesor Francisco de Sande González por su labor como tutor de este proyecto y orientando este trabajo, compartiendo su conocimiento y exigiendo siempre lo mejor. A mis padres, hermano y amigos por apoyarme en el transcurso de mis estudios universitarios.

Licencia



- © Esta obra está bajo una licencia de Creative Commons
Reconocimiento-NoComercial-CompartirIgual 4.0
Internacional.

Resumen

Este documento refleja el trabajo de investigación del alumno durante el proceso de desarrollo de una aplicación para dispositivos móviles Android mediante el uso de técnicas de realidad aumentada (RA) combinadas con técnicas de geolocalización.

Partimos de los conocimientos de programación en Java adquiridos en la asignatura: “Programación de Aplicaciones Interactivas” cursada en el itinerario de Ingeniería de Computación, también los conocimientos obtenidos en JavaScript y Node.js en la asignaturas: “Procesadores del Lenguaje” y “Prácticas Externas” realizada en la empresa “Itop Consulting”. Estas asignaturas, impartidas en el tercer y cuarto curso del Grado en Ingeniería Informática de la Universidad de La Laguna, han sido las que han sentado los fundamentos a partir de los cuales se ha desarrollado el trabajo.

Durante este proyecto, el alumno ha conseguido adquirir independencia en su trabajo, visión y planificación realizando tareas de investigación, desarrollo y documentación, que han dado como resultado la obtención de conocimientos durante el proceso de trabajo.

Palabras clave: Aplicaciones Android, Java, *Cloud Computing*, Dispositivos Móviles, Programación, Realidad Aumentada, Node.js, MongoDB, Google Maps.

Abstract

This document reflects the research work of the student during the development process of an application for Android mobile devices through the use of augmented reality (AR) techniques combined with geolocation techniques.

Based on the knowledge of the *Java* programming obtained in the subject: “*Programación de Aplicaciones Interactivas*” studied in the itinerary of Computer Engineering, also the knowledge acquired in *JavaScript* and *Node.js* in the subjects: “*Procesadores del Lenguaje*” and “*Prácticas Externas*” carried out in the company “*Itop Consulting*”. These subjects, carried out in the third and fourth year of the degree in Computer Engineering of *La Universidad de La Laguna*, have been the ones who have laid the foundations from which the work has been developed.

Moreover, the student has learned independence in his work and gained vision and scheduling aptitudes, developing different labors of research, development and documentation that have come to give her a wide knowledge during the development of this project.

Keywords: *Application for Android, Java, Cloud Computing, Mobile Devices, Programming, Augmented Reality, Node.js, MongoDB, Google Maps.*

Índice general

1. Objetivos	1
2. Herramientas y Tecnologías	2
2.1. Herramientas de Desarrollo	2
2.1.1. Android Studio	2
2.1.2. LaTex	3
2.1.3. Github	3
2.1.4. Guía de uso de estas tecnologías en relación con el proyecto ULL-AR	4
2.2. Tecnologías utilizadas	6
2.2.1. El Sistema Operativo Android	6
2.2.2. Realidad Aumentada	6
2.2.3. Node.js	14
2.2.4. MongoDB	15
2.2.5. Heroku	16
2.2.6. mLab	17
2.2.7. Google Maps	17
3. RA en entornos universitarios	19
3.1. Aplicaciones en entornos universitarios	20
3.1.1. Prácticas en laboratorios	20
3.1.2. Practicas de campo y visitas	21
3.1.3. Libros y documentos	21
3.1.4. Aprendizajes experimentales	21
3.1.5. Información sobre la universidad	21
3.2. Ejemplos de uso de la RA	22
3.2.1. Fabricación de un automóvil de carreras	22
3.2.2. AR Sandbox	23
4. La aplicación ULL-AR	24
4.1. Requisitos y ventanas de la aplicación	24

4.1.1.	Especificación detallada de los requisitos	25
4.1.2.	Ventanas de la aplicación	26
4.2.	Inicio de ULL-AR	30
4.2.1.	Ventana inicial	30
4.2.2.	Ventana de <i>Inicio de Sesión</i>	31
4.3.	Modo de Realidad Aumentada	33
4.3.1.	Acceso a los sensores	34
4.3.2.	Modelos encargados de la navegación	36
4.3.3.	Obtención de la información	42
4.3.4.	Visualización	43
4.4.	Fragments	45
4.4.1.	MapsFragment	45
4.4.2.	HomeFragment	47
4.4.3.	AboutFragment	49
4.5.	Menú	49
4.6.	Instalaciones de la ULL	50
4.7.	Preferencias del usuario	55
4.8.	Back-end de la aplicación	55
4.8.1.	Base de datos	56
4.8.2.	Configuración del servidor	56
4.8.3.	Despliegue en Heroku	60
5.	Conclusiones y futuras líneas de Trabajo	61
5.1.	Conclusiones	61
5.2.	Conclusions	62
6.	Presupuesto	63
	Bibliografía	65

Índice de figuras

2.1.	Android Studio, un IDE flexible e intuitivo.	3
2.2.	Google Glass. Desmostración del uso de RA en deportes.	7
2.3.	Marker-bases AR. Demostración de su funcionamiento.	8
2.4.	Espacio de la Realidad Mixta.	10
2.5.	RA Volcán. Ejemplo de uso de RA en Educación.	11
2.6.	Funcionamiento de Node.js.	14
2.7.	Ventajas de Node.js.	15
2.8.	Ventajas de MongoDB.	16
3.1.	Construcción de un automóvil de carreras gracias a la RA.	22
3.2.	Funcionamiento de AR Sandbox.	23
4.1.	Ventana <i>Inicio</i> y el Menú de <i>ULL-AR</i>	26
4.2.	Ventana de <i>Navegación en modo RA</i> y <i>Mapa ULL ULL-AR</i>	27
4.3.	Ventanas de <i>Todas las instalaciones ULL</i> e <i>Información de la instalación de ULL-AR</i>	28
4.4.	Ventanas de <i>Configuración</i> e <i>Información de ULL-AR</i>	29
4.5.	Disposición de los ejes de un dispositivo Android.	35
4.6.	Explicación de los atributos “coneValue”, “dirToSite” y “disToSite”.	38

Capítulo 1

Objetivos

Este documento resume el trabajo de investigación y desarrollo realizado por el alumno en la consecución de su Trabajo de Fin de Grado (TFG), con el que culminará sus estudios del Grado en Ingeniería Informática cursados en la Escuela Superior de Ingeniería y Tecnología de la Universidad de la Laguna (ULL).

Este TFG tiene los siguientes objetivos principales:

1. Por un lado se pretende ampliar los conocimientos en tecnologías móviles en el sistema operativo *Android* [1] y en el desarrollo de aplicaciones para este sistema operativo.
2. Otro objetivo presente en este TFG es que el alumno investigue y profundice en las técnicas y tecnologías de realidad aumentada presentes en la actualidad.
3. A su vez, se busca que el alumno adquiera conocimientos sobre las capacidades y ventajas del uso de servicios de computación en la nube.
4. También se pretende que el alumno se familiarice con el uso de herramientas de control de versiones utilizando *Github* [2] y de edición de textos técnicos utilizando *LaTeX* [3].
5. Por último, tras las labores de investigación y recopilación de información correspondientes, se espera que el alumno aplique los conocimientos adquiridos para desarrollar una aplicación funcional que cubra las necesidades propuestas.

Capítulo 2

Herramientas y Tecnologías

Este capítulo tiene como objetivo presentar las distintas herramientas software y tecnologías empleadas por el alumno en el desarrollo de la aplicación Android objeto de este TFG. En adelante usaremos el nombre **ULL-AR** para referirnos a la misma.

2.1. Herramientas de Desarrollo

2.1.1. Android Studio

Android Studio [4] es el IDE [5] (Entorno de Desarrollo Integrado) oficial para el desarrollo de aplicaciones en Android, basado en IntelliJ IDEA [6]. Android Studio ofrece una serie de funcionalidades que han facilitado al desarrollador numerosas tareas, entre las cuales se pueden destacar:

- Un sistema de compilación basado en Gradle [7] que ha simplificado tanto la inserción de dependencias de las distintas librerías que se han tenido que utilizar, como la compilación de la aplicación.
- Un emulador rápido y fácil de utilizar que ha ayudado a visualizar las distintas pantallas durante el desarrollo, el cual ha sido utilidad en las primeras ventanas de la aplicación.
- La facilidad y velocidad para aplicar cambios a aplicaciones ya funcionando.
- Un sistema de visualización de las diferentes pantallas muy completo, con soporte visual para añadir componentes y cambiar atributos fácilmente.
- Un sistema de depuración, con una interfaz sencilla e intuitiva.



Figura 2.1: Android Studio, un IDE flexible e intuitivo.

2.1.2. LaTex

LaTeX [3] es un sistema de composición de textos, orientado a la creación de documentos que presenten una alta calidad tipográfica. Por sus características y posibilidades, es usado especialmente en la generación de artículos y publicaciones científicas que incluyen, entre otros elementos, expresiones matemáticas, gráficos o figuras.

LaTeX está formado por un gran conjunto de macros de TeX, escrito por Leslie Lamport en 1984, con la intención de facilitar el uso del lenguaje de composición tipográfica, creado por Donald Knuth. LaTeX es software libre bajo licencia LPPL.

Se ha decidido usar esta herramienta debido a la calidad profesional de los documentos que es posible generar con ella. Otra de las ventajas de LaTeX es que permite separar el formato y el contenido del documento, lo cual permite concentrar el esfuerzo en la creación del contenido sin tener que ocuparse del diseño.

2.1.3. Github

GitHub [2] es una plataforma de desarrollo colaborativo para alojar proyectos que utiliza el sistema de control de versiones Git. GitHub fue escrito en Ruby on Rails. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.

Se ha decidido crear un repositorio [8] en esta plataforma para poder llevar un control y una trazabilidad del proyecto ULL-AR. El tutor y el alumno han trabajado en este repositorio de manera conjunta. En el caso del tutor, principalmente para revisar el seguimiento semanal y llevar un control de las tareas. En el caso del alumno, para tener un repositorio donde subir los distintos elementos que se han ido

generando a lo largo del trabajo. En el repositorio del proyecto [8] están disponibles públicamente todos los ficheros necesarios para la elaboración con LaTeX de esta memoria, los ficheros correspondientes a la presentación para la defensa de este TFG así como el proyecto de Android Studio que contiene el código fuente de ULL-AR.

Mediante el uso de este repositorio, el alumno ha conseguido ampliar sus conocimientos en Git y familiarizarse con la interfaz de GitHub.

Para instalar GitHub en Linux se utiliza el siguiente comando:

```
1 $ sudo apt install git-all
```

2.1.4. Guía de uso de estas tecnologías en relación con el proyecto ULL-AR

En este apartado se explica con cierto nivel de detalle los pasos a seguir para:

- Descargar desde su repositorio todo el código fuente, tanto de esta memoria como de la aplicación ULL-AR.
- Compilar en Android Studio el proyecto ULL-AR para generar una aplicación para dispositivos móviles.
- Compilar con Latex esta memoria para generar el correspondiente fichero en formato PDF.

Descargar repositorio de la aplicación

Para descargar el repositorio de ULL-AR se necesitará tener instalado GitHub. Con GitHub instalado, se ejecuta el siguiente comando en consola para descargar el repositorio:

```
1 $ git clone https://github.com/alehdezp/TFG-ULL-AR
```

En caso de que no se disponga de GitHub se puede descargar directamente desde el repositorio [8], seleccionando el botón “Clone or download” y luego se selecciona “Download as ZIP”. Tras esto se descargará un archivo comprimido con todo el repositorio de ULL-AR.

Compilar el proyecto en Android Studio

Para compilar el proyecto es necesario tener instalado el IDE de Android Studio en el ordenador que se desee compilar el proyecto.

Una vez iniciado Android Studio, se indica que se quiere abrir un proyecto y se busca en la carpeta en la que se encuentra el repositorio de ULL-AR y se abre el proyecto que se encuentra en la ruta “*app/ULL_Navigation*”.

La primera vez que se abre el proyecto, en el caso de que no se haya hecho automáticamente, se ha de seleccionar en la barra superior la opción de “Build” y luego “Make Project”. Esto creará y configurará el proyecto para que se pueda empezar a trabajar con él.

Para poder instalar la aplicación que genera el proyecto en un dispositivo Android hay dos opciones. La primera es generar un archivo “.apk”. Para generar este archivo se ha de seleccionar en la barra superior la opción “Build”, luego “Build Bundle(s) / APK(s)” y por último “Build APK”. Esto generará un fichero *app-debug.apk* en la ruta “*ULL_Navigation/app/build/outputs/apk/debug*”, este fichero hay que colocarlo dentro del dispositivo Android en el que se quiera instalar la aplicación y se ejecuta.

La segunda opción es ejecutar e instalar directamente la aplicación desde Android Studio. Para ello hay que tener activado en el dispositivo Android la opción de depuración por USB. Esta opción solo estará disponible cuando se tenga activado el modo desarrollador en el dispositivo Android. En función de la marca y modelo del dispositivo la forma en la que se activa este modo varía. Con la opción activada solo hay que conectar el dispositivo al ordenador y seleccionar en el móvil el modo de “Transeferencia de archivos”, a continuación en Android Studio se selecciona la opción “Run” y luego “Run ‘app’”. Con estos pasos ya instalará la aplicación en el dispositivo y se solicitarán permisos en el dispositivo móvil para confiar en el desarrollador e instalar la aplicación ULL-AR.

Compilar la memoria del proyecto

Para compilar el proyecto en LaTeX en un máquina con Linux es necesario instalar “TeX Live” y “make”. Con el siguiente comando se instalarán ambos:

```
1 $ sudo apt install texlive-full make
```

Dentro de la carpeta “*Memoria/*” del repositorio, se abrirá una consola y se ejecutará el siguiente comando el cual genera el archivo con la presente memoria *memoria-TFG-Alejandro.pdf*.

```
1 $ make
```

2.2. Tecnologías utilizadas

A continuación, se revisan las distintas tecnologías utilizadas en el desarrollo de la aplicación.

2.2.1. El Sistema Operativo Android

Android [1] es un sistema operativo (SO) que emplea Linux en la interfaz del hardware. Los componentes del SO subyacentes se codifican en C o C++, pero las aplicaciones se desarrollan en Java. De esta manera Android asegura una amplia operatividad en una gran variedad de dispositivos debido a dos hechos: la interfaz en Linux ofrece gran potencia y funcionalidad para aprovechar el hardware, mientras que el desarrollo de las aplicaciones en Java permite que Android sea accesible para un gran número de programadores conocedores del código.

Este SO fue diseñado principalmente para dispositivos móviles con pantalla táctil: dispositivos móviles, tablets y otros dispositivos como televisores o automóviles. Fue desarrollado inicialmente por Android Inc., empresa que fue respaldada económicamente por Google y más tarde adquirida por esta misma empresa.

2.2.2. Realidad Aumentada

La Realidad Aumentada (RA) [9] o Augmented Reality (AR) en inglés, es el término que se usa para definir la visión de un entorno físico del mundo real, a través de un dispositivo tecnológico. Este dispositivo, permite expandir el mundo físico añadiendo capas de información digital generadas por un computador, como pueden ser imágenes, sonidos y vídeos a la visión del entorno en tiempo real.

La RA está cambiando la manera en la que sus usuarios pueden ver el mundo. Actualmente es una tecnología que se encuentra en auge debido a su enorme potencial. Empresas de numerosos sectores ya han estado invirtiendo en su desarrollo debido a que los beneficios que traerá esta tecnología y las posibles aplicaciones por descubrir son prometedoras.

En un futuro la RA estará integrada en el día a día y formará parte de la vida cotidiana. Sus posibles aplicaciones no tienen límites, pueden llegar desde reconocer plantas e incluso monumentos y mostrar información sobre lo que se está viendo, hasta añadir información en tiempo real en una operación a un paciente, comprobar cómo queda un mueble en un salón o sus aplicaciones para realizar videojuegos, como se puede comprobar con el reciente éxito de Pokemón Go!.

A continuación, se explicará en detalle:

- ¿Cómo funciona esta tecnología?.
- Tipos de Realidad Aumentada.

- Diferencias entre Realidad Aumentada y Realidad Virtual.
- Realidad Mixta.
- Futuro y usos de la Realidad Aumentada.
- Integración de Realidad Aumentada en Android Studio.

¿Cómo funciona esta tecnología?

La RA necesita de un dispositivo de visualización en el que mostrar esta unión del entorno real junto con la información digital. Esta unión puede ser visualizado en múltiples dispositivos: pantallas, gafas, dispositivos portátiles, dispositivos móviles, etc.

Además de estos sistemas de visualización, se necesita de un sistema de computación que realice los cálculos y reciba los datos provenientes de múltiples sensores, es decir: una CPU, una GPU, RAM, GPS, WIFI, bluetooth, acelerómetro, giroscopio, cámara, etc. Gracias a estos elementos el sistema puede reconocer el entorno real.

Todo esto necesita una parte software. El software en una primera parte deberá de reconocer el terreno, ubicaciones, objetos e imágenes, mediante los datos de los sensores y la cámara. Este proceso de transformación de diferentes conjuntos de datos a un sistema de coordenadas se llama registro de la imagen [10].

Posteriormente el software deberá reestructurar el mundo real en función del registro de imágenes, añadiendo y combinando la información correspondiente al entorno para generar la imagen de RA. Existen múltiples maneras en las que se puede reestructurar este mundo.



Figura 2.2: Google Glass. Desmostración del uso de RA en deportes.

Tipos de Realidad Aumentada

Existen hoy en día cuatro tipos de RA:

- Marker-based AR. Se basa en el reconocimiento de imágenes conocidas como *marker* o marcadores. Los marcadores son imágenes distintivas que son reconocidas fácilmente por un dispositivo, debido a que contienen puntos visuales únicos. Un buen ejemplo de este tipo de imágenes son los conocidos códigos QR [11]. Una vez se ha reconocido un marcador, se puede agregar información virtual. Esta información puede ser la incorporación de animaciones o videos en una página de un libro de texto, simulaciones de objetos 3D o arquitecturas sin llegar a construirlas de forma física (véase Figura 2.3).



Figura 2.3: Marker-bases AR. Demostración de su funcionamiento.

- Markerless AR. Corresponde a la RA que recoge los datos de su posición y orientación para mostrar la información correspondiente a esa área. Estos sistemas utilizan los datos obtenidos de la cámara, GPS, brújula, giroscopio y acelerómetro para establecer la ubicación del usuario en el entorno. Utiliza técnicas para reconocer terreno o ambientes para calcular la posición y orientación de la cámara. Con este tipo de tecnología se puede probar un mueble en el salón antes de llegar a comprarlo.
- Projection-based AR. Este modo de RA consiste en la proyección de luz en superficies y objetos en el mundo real. Existen muchos usos interesantes de esta tecnología, como aplicaciones para el uso de teclados virtuales proyectados que reconozcan cuando una “tecla” es pulsada. Esta proyección también se puede hacer en medio del aire con ayuda de la tecnología de láseres de plasma.

- Superimposition-based AR. Esta tecnología reemplaza la imagen original por una de RA, de forma completa o parcial. El reconocimiento de objetos juega un papel fundamental en este tipo de tecnología. Tiene gran utilidad el campo de la medicina, por ejemplo, un doctor podría examinar a un paciente mientras ve la imagen de RA que se creado a partir de una visión de rayos X y la imagen real de paciente, así puede ver y entender mejor el daño en un hueso.

A parte de los cuatro tipos de RA, existen subtipos dentro de cada uno de ellos.

La RA basada en la geolocalización es un tipo de “Markerless AR” que se centra más en los cálculos de la orientación y geolocalización del dispositivo. Esta técnica RA puede proveer de ayuda viajeros que necesiten una mano para moverse por la ciudad, ya que mediante el reconocimiento de su ubicación y la orientación pueden mostrarles la ruta para llegar a su destino o mostrarle información de los puntos de interés que les rodean. Esta será la técnica de RA implementada en la aplicación a desarrollar en este TFG.

Diferencias entre Realidad Aumentada y Realidad Virtual

En los últimos años la realidad virtual (RV) [12] o virtual reality (VR) en inglés y la realidad aumentada han empezado a recibir mucha más atención. Llevando a desarrolladores realizar integraciones de ambas tecnologías en numerosas industrias.

De acuerdo con un análisis de expertos, la RV iba tener el liderazgo en 2018 como tecnología pionera, sin embargo, la RA va a tener mucha más importancia y a largo plazo, llegará a formar parte del día a día.

La realidad virtual es un entorno de escenas u objetos de apariencia real. Aleja al usuario del entorno real y le brinda la sensación de estar inmerso en él. Dicho entorno es contemplado por el usuario a través de un dispositivo conocido como gafas o casco de realidad virtual. Este puede ir acompañado de otros dispositivos, como guantes o trajes especiales, que permiten una mayor interacción con el entorno, así como la percepción de diferentes estímulos que intensifican la sensación de realidad.

La realidad aumentada no aísla al usuario de mundo exterior, sino que traslada al mundo real objetos virtuales mediante la superposición de imágenes en tiempo real.

Realidad Mixta

Existe otro tipo de tecnología que nace de la unión de realidad aumentada y la realidad virtual, la realidad mixta (RM) [13]. Es un tipo de realidad similar a la realidad aumentada, pero con una idea más ambiciosa de mezclar lo real con lo virtual.

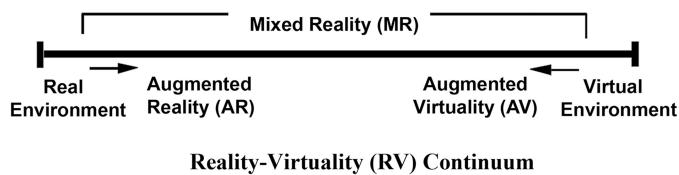


Figura 2.4: Espacio de la Realidad Mixta.

En la RM se trata de llevar el mundo real al mundo virtual (véase Figura 2.4). La idea es generar un modelo 3D de la realidad y sobre él superponer información virtual. De esta forma, se podrán combinar ambas realidades para agregar contenido adicional de valor para el usuario de RM. La RM es mucho más inmersiva que la realidad aumentada y requiere de mucha más capacidad de procesamiento.

Futuro y usos de la Realidad Aumentada

Actualmente la RA se encuentra más disponible a cualquier usuario, que en años anteriores. Dispositivos como los teléfonos móviles ya incorporan las primeras muestras de esta tecnología, la cual está aún en una fase inicial de desarrollo, pero ya se puede ver el potencial y la enorme importancia que va a cobrar en un futuro.

Los usos actuales de esta tecnología se acercan a todos los sectores conocidos:

- Realidad Aumentada en educación. La llegada de la RA afectará a los procesos convencionales de aprendizaje. La RA tiene la capacidad de cambiar el horario y el lugar en el que se estudia y la posibilidad de introducir nuevos métodos de enseñanza.

Actualmente gran parte de la población joven tiene un dispositivo móvil el cual es un medio idóneo para la RA. Por lo tanto, se dan unas condiciones adecuadas para que la RA profundice en el campo de la educación y se hagan más descubrimientos ya que cada estudiante va a tener un dispositivo a mano capaz de reproducir la RA, lo cual puede ayudar al alumnado a tener contenidos más accesibles sobre cualquier asignatura o conseguir que información compleja sea más fácil de entender. Un ejemplo claro sería la creación de libros interactivos que al ser apuntados con la cámara del móvil muestren el funcionamiento en 3D de un volcán (véase Figura 2.5) o del latido de un corazón.

- Realidad Aumentada en los videojuegos. Sin duda uno de los sectores en lo que más interés se tiene para desarrollar esta tecnología y quizás donde más se haya avanzado en realidad aumentada. Todas las grandes empresas de este sector tienen ya potentes desarrollos y lanzamientos de videojuegos

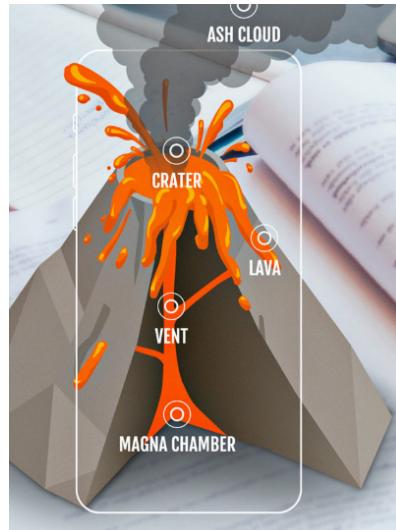


Figura 2.5: RA Volcán. Ejemplo de uso de RA en Educación.

que combinan la realidad física con la virtual, con múltiples posibilidades de personalización dentro de cada juego. En un futuro esta tecnología revolucionará todo el sector de los videojuegos y cambiará la manera en la que se interactúa con ellos.

Un claro ejemplo del potencial de la RA en este sector, está en el éxito de Pokemón Go! [14]. Una aplicación para dispositivos móviles que utiliza técnicas de RA basadas en la localización, la cual a través de la cámara del dispositivo y modelos 3D para representar a los personajes de la saga Pokemón [15], los cuales se encuentran escondidos en ubicaciones del mundo real.

- Realidad Aumentada en la industria. El desarrollo de aplicaciones de RA en el ámbito industrial también está creciendo, ya que ayuda a mejorar la productividad de los ciclos de trabajo. Por ejemplo, algunas empresas están desarrollando aplicaciones que ayuden a los trabajadores de una cadena de montaje. De este modo, los empleados pueden obtener información adicional sobre las acciones que llevan a cabo. Este mismo sistema también se puede implementar en las reparaciones de vehículos o maquinaria industrial, ya que la aplicación puede mostrar toda clase de avisos sobre las piezas deterioradas. Incluso puede llegar a mostrar contenido visual en 3D sobre cómo llevar a cabo la reparación o sustitución de esos elementos.
- Realidad Aumentada en turismo. El negocio turístico siempre ha intentado estar al día con la nueva tecnología para poder ofrecer nuevos servicios,

formas de publicitarse, transporte y actividades de ocio. Por eso no es raro que la RA se haya hecho un hueco en este sector debido al potencial que tiene para mejorar la experiencia de los turistas.

La RA tiene la capacidad de generar aplicaciones que permitan a los turistas realizar una visita guiada por las ciudades que visiten, permitiéndoles moverse sin problemas por la ciudad, señalando puntos de interés, datos históricos, restaurantes, hoteles, etc. El mismo concepto podría aplicarse en museos y zoos. Otro uso interesante sería el de romper la barrera del lenguaje gracias a la traducción inmediata de señales, textos y anuncios, al idioma del turista. Para los Juegos Olímpicos de Tokio 2020 se espera tener esta tecnología preparada para poder traducir a los visitantes en tiempo real todos los textos, señales y anuncios, a través de sus dispositivos móviles.

- Realidad Aumentada en medicina. En cuanto a la medicina, es interesante ver cómo avanza la tecnología en este campo que sin duda apunta prometedor para mejorar la calidad de vida y salud de la población.

Los principales usos que se plantean se encuentran enfocados principalmente en los quirófanos, en los que el especialista o cirujano monte una especie gafas-pantalla que le permitan realizar la operación sin la necesidad de apartar la vista del paciente para consultar información o ir monitorizando la operación. Esto se traduce en operaciones más rápidas y seguras sin que el cirujano se despiste. A su vez, esto podría mostrar información anatómica sobre el paciente en tiempo real, es decir, gracias a algoritmos de inteligencia artificial, permitir identificar nervios, venas mayores y huesos, y que estos sean marcados con un distinto color, facilitando mucho las labores de los médicos.

Android Studio y la RA

Para la integración de realidad aumentada en Android, se han probado y estudiado los kits de desarrollo de software (SDK [16]) de realidad aumentada disponibles para Android Studio. Los SDK que han sido evaluados son:

- **Vuforia** [17] es un SDK de realidad aumentada que permite a los desarrolladores de aplicaciones crear rápidamente experiencias de RA inmersivas, de alta fidelidad y centradas en el móvil. El SDK de Vuforia aprovecha la tecnología de visión por computadora para identificar y rastrear imágenes y objetos 3D en tiempo real. Esta funcionalidad permite orientar y colocar objetos virtuales, incluidos modelos 3D y otros contenidos, en relación con el entorno del mundo real. Los modelos 3D y la información digital se pueden

superponer sobre la escena del mundo real y verlos en relación con el entorno a través de un dispositivo móvil.

Este SDK es uno de los mejores del mercado con buen soporte para aplicaciones multiplataforma. Un inconveniente de Vuforia es que está diseñado para desarrollar aplicaciones y juegos principalmente en plataformas como Unity [18]. En cuanto al SDK para Android Studio, la falta de soporte, de documentación y los numerosos problemas hallados a la hora de instalarlo, lo convierten en un SDK con el que nos ha sido muy difícil trabajar. Es por ello que se ha decidido descartar Vuforia para su integración en ULL-AR.

- **Kudan AR SDK** [19] es una plataforma diseñada para desarrolladores de RA como una plataforma preparada para admitir tanto en el reconocimiento basado en marcadores como sin marcadores y el seguimiento de los mismos. El motor Kudan SDK principal, se desarrolla completamente en C++ y posee optimizaciones específicas de la arquitectura desarrolladas para proporcionar un rendimiento más rápido y sólido sin afectar negativamente el espacio de memoria.

La instalación de Kudan AR fue rápida y sencilla, debido a que Kudan dispone de la documentación necesaria para instalarlo en Android Studio. Además, dispone de una guía bien explicada para comenzar la implementación de sus funcionalidades. Se ha optado por la integración de este SDK debido a su sencillez y a que ofrece los requisitos mínimos para el objetivo de RA de ULL-AR.

- **MaxST SDK** [20] de realidad aumentada proporciona un motor RA integral multiplataforma equipado con todas las características requeridas por los desarrolladores para crear experiencias y aplicaciones de RA. MaxST AR proporciona las siguientes funcionalidades: seguimiento instantáneo, SLAM [21] (utiliza la cámara del teléfono inteligente para crear un “mapa virtual” del área circundante), rastreo de objetos y reconocimiento de imágenes y de marcadores.

MaxST AR ofrece unos de los mejores soportes para la integración de un SDK de RA en Android Studio. Dispone de tutoriales en su página web para la instalación del SDK, explicando el funcionamiento e implementación de cada una de las funcionalidades que ofrece. No se le ha encontrado ningún inconveniente para integrarlo en la aplicación, pero se ha preferido integrar Kudan AR SDK por su simplicidad.

2.2.3. Node.js

Node.js [22] es una librería y entorno de ejecución de E/S dirigida por eventos y por lo tanto asíncrona que se ejecuta sobre el intérprete de JavaScript creado por Google llamado Chrome V8 [23]. Node.js es un entorno de JavaScript del lado del servidor [24], basado en eventos. Utilizar el motor de Chrome V8 permite a Node.js un entorno de ejecución que compila y ejecuta JavaScript a velocidades increíbles.

Node.js fue desarrollado con el objetivo que fuera un sistema escalable y que tuviera la consistencia de generar un elevado número de conexiones de forma simultánea con el servidor. La mayoría de las tecnologías que de los servidores tradicionales tienden a accionar las peticiones de forma aislada y mediante hilos independientes. Esto se traduce en que a mayor número de solicitudes mayor es la cantidad de recursos necesarios para responderlas. Node.js se ha desarrollado para optimizar la gestión de estas solicitudes.

La solución que propone Node.js se basa en el tratamiento de las solicitudes de forma unificada en un único hilo complementado con un bucle de eventos de tipo asíncrono. De este modo cada petición que se recibe se trata como un evento y pertenecen a este único bucle (véase Figura 2.6). Este nuevo replanteamiento proporciona un lenguaje con una capacidad para gestionar una gran cantidad de solicitudes y conexiones con la máxima eficiencia.

Node.js utiliza un E/S de tipo asíncrono. En los modelos de tipo asíncronos, las tareas que efectúa el servidor se realizan de manera simultánea y repartidas entre los hilos del procesador. Este procedimiento evita que se produzcan bloqueos y proporciona una mayor potencia y velocidad de procesamiento.

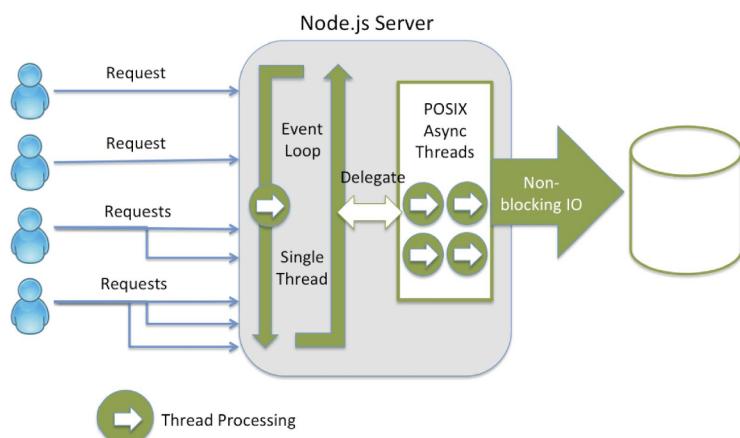


Figura 2.6: Funcionamiento de Node.js.

Otro de sus puntos fuertes es su gestor de paquetes Node Package Manager

(NPM). Gracias a este gestor se pueden instalar paquetes, módulos y agregar dependencias de manera simple.

Se ha optado por el uso de un servidor que implemente la tecnología Node.js como servidor de ULL-AR por la facilidad para ejecutar este servidor en cualquier tipo de sistema operativo y para su futura integración en la nube, gracias a la plataforma Heroku de la que se hablará más adelante. Otra de las razones por las que se ha elegido Node.js es porque el proceso de desarrollo y programación es rápido y sencillo y además se tenían conocimientos previos sobre esta tecnología, lo que ha facilitado su uso.

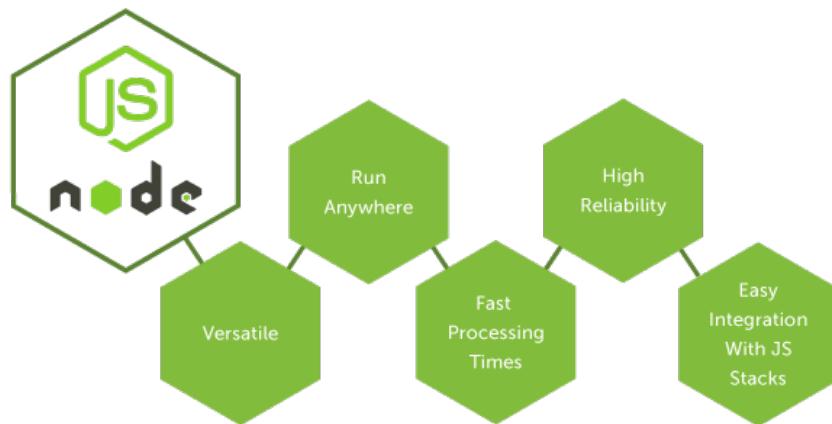


Figura 2.7: Ventajas de Node.js.

Para instalar Node.js en una máquina con Linux se deben ejecutar estos comandos:

```

1 $ curl -sL https://deb.nodesource.com/setup_11.x | sudo -E bash -
2 $ sudo apt install -y nodejs

```

2.2.4. MongoDB

Descripción

MongoDB [25] es un sistema de base de datos multiplataforma NoSQL [26] orientado a documentos. Esta base de datos es de esquema libre, es decir, al contrario que las bases de datos relacionales no utilizan una tabla fija para guardar la información.

Un registro en MongoDB es un documento, cuya estructura de datos se compone por el par campo y valor. Utilizan un formato para guardar estas estructuras que se llama BSON [27] también llamada JSON [?] Binario. Este formato tiene una ventaja pese a que puede ocupar más espacio de lo que lo haría el formato JSON. BSON guarda de forma explícita las longitudes de los campos, los índices de los arrays, y demás información útil para el escaneo de datos y así agilizar las búsquedas en estos documentos.

Estos documentos, que a su vez incorporan una clave primaria como identificador, son la unidad básica de datos en MongoDB. Las colecciones en MongoDB contienen un conjunto de documentos y funciones equivalentes a las de las bases de datos relacionales.

Debido a los requisitos de la aplicación, el uso de una base de datos de MongoDB permitía la creación de una base de datos de esquema libre, es decir, sin la preocupación de crear tablas de datos para almacenar la información, que permitía guardar de forma simple toda la información necesaria para el funcionamiento de la aplicación. Además, las consultas a esta base de datos se facilitan con el uso de un servidor Node.js, que ofrece un conjunto de librerías y paquetes preparados para realizar la comunicación con una base de datos de MongoDB.

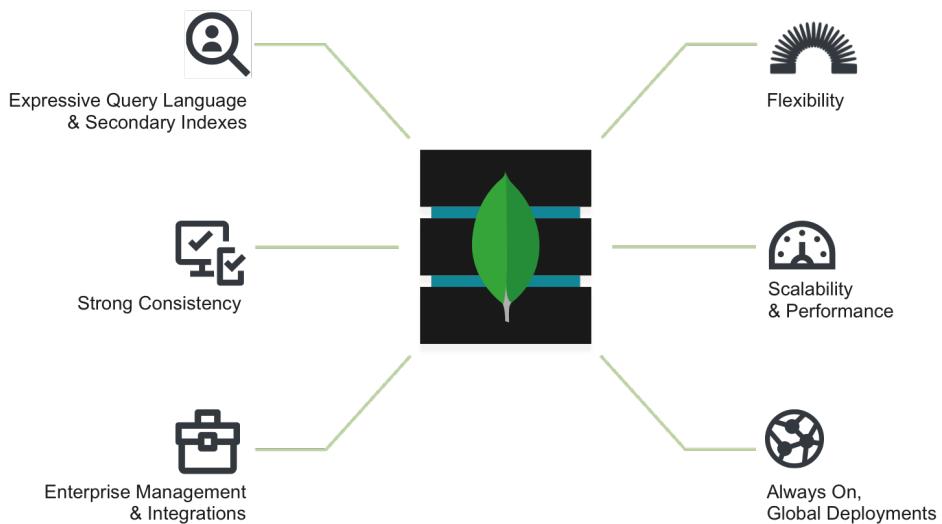


Figura 2.8: Ventajas de MongoDB.

2.2.5. Heroku

Heroku es una plataforma como servicio de computación (PaaS [28]) en la nube que soporta distintos lenguajes de programación. Heroku permite a desarrolladores

y compañías construir, proporcionar, monitorizar y escalar aplicaciones. Es una forma sencilla de montar una infraestructura en la nube.

Heroku es conocida por ejecutar las aplicaciones en “dynos”. Un dyno es simplemente un ordenador virtual que pueden ser encendido y/o apagado en función del tamaño y requisitos de la aplicación.

A cada uno de estos dynos se le puede configurar más espacio o capacidad de procesamiento, o se pueden añadir más dynos si la aplicación lo necesita. Heroku a cada mes realiza un cobro en función de los dynos que el usuario tenga contratados.

Se ha optado por utilizar el dyno gratis que ofrece Heroku a cada repositorio. El cual tienen un límite de 1000 horas activas al mes y se pone en suspensión a los 30 minutos sin recibir tráfico. Una vez que vuelve a recibir tráfico se activa de nuevo.

Se ha utilizado Heroku para albergar el servidor de ULL-AR, ya que ofrece un servicio gratuito idóneo para el diseño de pequeñas aplicaciones web en la nube, sin la complejidad de estar implementando un servidor físico.

Para instalar Heroku basta seguir las indicaciones que se hallan en [29].

2.2.6. mLab

mLab es un servicio de base de datos en la nube totalmente gestionado que ofrece aprovisionamiento y escalados automáticos de las bases de datos MongoDB, copia de seguridad y recuperación, monitoreo, herramientas de administración basadas en web y soporte experto. La plataforma de base de datos como servicio de mLab corre sus máquinas en proveedores de servicios en la nube como AWS [30], Azure [31] y Google Cloud [32].

En esta plataforma se ubicará la base de datos de ULL-AR. Dadas las facilidades de acceso a ella e integración con el servidor de la aplicación en Node.js, es una opción simple, fácil de manejar y con posibilidad de escalar.

2.2.7. Google Maps

La API de Google Maps [33] fue publicada para Android en 2008 y en 2012 para IOS. Esta API permite utilizar mapas basados en datos de Google Maps en una aplicación Android, y además ofrece métodos para personalizar el mapa:

- Creación de marcadores, polígonos y superposiciones sobre el mapa para resaltar puntos o zonas.
- Permite cambiar la vista del usuario de modo que se muestre un área del mapa en particular.

- Ofrece la posibilidad de elegir el tipo de mapa: de carreteras, satélite, híbrido (fusión de carretera y satélite) y de terreno.

Esta API fácil de integrar y con soporte de Google, es la mejor opción para tener un mapa funcional y que permita ubicar al usuario de la aplicación.

Capítulo 3

RA en entornos universitarios

En este capítulo se expondrán los usos y ventajas de la integración de la realidad aumentada en entornos universitarios.

La realidad aumentada se presenta en el ámbito educativo como una tecnología capaz de aportar transformaciones significativas a la forma en que el alumnado percibe y accede a la realidad física, proporcionando así, experiencias de aprendizaje más ricas e inmersivas.

En la actualidad en educación, la realidad aumentada rara vez se usa, pero cada vez más docentes, investigadores y desarrolladores están comenzando a moverse hacia nuevos métodos de enseñanza más interactivos. Por ello, la continua implantación de nuevas tecnologías en las aulas, junto al incremento de dispositivos móviles en la población, sitúa a la RA en una posición destacada para introducirse en las aulas.

Las aplicaciones que tiene la RA, en lo referente a la creación de materiales didácticos y actividades de aprendizaje son múltiples, directas y fáciles de imaginar en prácticamente todas las disciplinas, sobre todo, las relacionadas con las ciencias aplicadas (ingeniería, química y física, biología), pero también en el campo del diseño industrial, la cirugía, la arqueología, etc. La tecnología de la RA permite cambiar la forma de entender los contenidos de aprendizaje, puesto que aporta nuevas formas de interacción con el mundo real a través de capas digitales de información que amplían, completan y transforman en cierto modo la información inicial.

Los beneficios potenciales de la RA aplicados a la educación incluyen:

- Aumentar o enriquecer la información de la realidad para hacerla más comprensible al alumno.
- El uso de una interfaz tangible para la manipulación de objetos, que permite observar un objeto desde diferentes puntos de vista, seleccionando, el discente, el momento y posición de observación.

- Potenciar el aprendizaje ubicuo [34].
- Crear escenarios “artificiales” seguros para el alumnado como pueden ser laboratorios o simuladores.
- Enriquecer los materiales impresos para el alumnado con información adicional en diferentes soportes.
- Facilita la colaboración efectiva y discusión entre el alumnado.

Numerosos trabajos han demostrado la eficacia de que las aplicaciones de RA pueden mejorar el proceso de aprendizaje, aumentar la motivación y la efectividad [35] [36].

A la hora de diseñar un sistema de RA orientado a la enseñanza, existen una serie de requisitos para que el alumno no se distraiga con su uso y asegurar que el objetivo de mejorar el aprendizaje se cumpla. Los requisitos de estos sistemas de RA son:

- Ser sencillo y robusto.
- Permitir que el educador ingrese información de manera simple y efectiva.
- Proporcionar al alumno información clara y concisa.
- Permitir una fácil interacción entre estudiantes y educadores.
- Realizar procedimientos complejos transparentes para los alumnos.
- Ser rentable y fácilmente extensible.

3.1. Aplicaciones en entornos universitarios

Las aplicaciones de la RA en entornos universitarios son múltiples y diversas y no solo se encuentran enfocadas al ámbito de la enseñanza. A continuación, analizaremos las posibles aplicaciones que se encuentran de la tecnología de RA en la actualidad para el colectivo universitario.

3.1.1. Prácticas en laboratorios

En las realizaciones de las prácticas en un laboratorio, existe una gran gama de herramientas y máquinas que son desconocidas para el alumno. Para facilitar el uso de este instrumental, la RA permite asociar a cada elemento del laboratorio, información sobre el mismo, tutoriales de uso, indicaciones de los pasos a seguir

para la elaboración de la práctica. Esto permite agilizar el proceso de realización de las prácticas evitando que el alumnado requiera del profesorado para resolver parte de sus dudas y permitiéndole centrarse más en la realización de la práctica.

3.1.2. Prácticas de campo y visitas

La posibilidad de realizar una visita a un museo e identificar cada estatua o cuadro, y mostrar información adicional sobre su autor, datos históricos sobre las obras artísticas, reconocer estilos e influencias del autor para su realización, permite al usuario mejorar su adquisición de conocimiento al mezclarse el objeto de información y conocimiento acerca de aquél en el mismo lugar. Este mismo concepto se puede aplicar a prácticas de campo a medios rurales, bosques, montañas y lagos, permitiendo al usuario identificar la flora y la fauna del terreno y mostrar características únicas sobre ellos, que a simple vista son imperceptibles.

3.1.3. Libros y documentos

La RA permite el enriquecimiento de los textos de libros y documentos con información adicional referente al tema tratado en la lectura. Esta información puede ser un vídeo, texto adicional con información específica, un enlace web o incluso una representación en 3D de un objeto, con el que alumno pueda interactuar, con una explicación detallada del contenido teórico que se muestra en el libro. Esto promueve diferentes formas de asimilar el contenido de los libros con métodos más interactivos y participativos, lo cual mejora el proceso de aprendizaje.

3.1.4. Aprendizajes experimentales

Cada área o disciplina que se imparte en cualquier tipo de formación o titulación universitaria consta de parte experimental. A la realización de esta parte experimental se le pueden aplicar técnicas de RA que faciliten el aprendizaje y el desarrollo de competencias transversales. Un ejemplo del uso de la RA, se puede encontrar en medicina donde el uso de visores como las Google Glass pueden ayudar a la alumna a identificar anatómicamente las partes del cuerpo de un paciente al que se le está realizando una operación en el quirófano. En disciplinas de ingeniería o arquitectura la integración de técnicas de RA permite la visualización de modelos 3D de edificios y construcciones.

3.1.5. Información sobre la universidad

Dentro de las aplicaciones de la RA a los centros referidos al ámbito universitario, se encuentra el uso de marcadores y de técnicas de localización para dotar al

alumno de información sobre: eventos, seminarios, jornadas culturales y guiar al usuario hasta la ubicación, recinto, aula o sala en la que se desarrollan estas actividades. También permite a los usuarios inscribirse de manera simple y cómoda a estos eventos, con el simple escaneo de un código QR asociado.

Este ámbito de utilización de la RA en entornos universitarios es el que se ha elegido como caso de uso para el desarrollo de ULL-AR.

3.2. Ejemplos de uso de la RA

A continuación se comentarán algunas de las aplicaciones de técnicas de RA que se pueden encontrar en la actualidad.

3.2.1. Fabricación de un automóvil de carreras

El alumnado de la Universidad de Bath está utilizando una nueva herramienta RA desarrollada por la compañía de tecnología Rocketmakers. La herramienta RA ayudará en la construcción de la carcasa del automóvil, conocida como monocabo, específicamente con la aplicación de laminados de fibra de carbono. Su vehículo competirá en la competición de Fórmula Estudiantil 2019 organizada por la Institución de Ingenieros Mecánicos.



Figura 3.1: Construcción de un automóvil de carreras gracias a la RA.

Este proceso se llevará a cabo durante una semana, y los alumnos de Team Bath Racing trabajarán por turnos para aplicar cada laminado de fibra de carbono recortado en la ubicación correcta. La herramienta de Rocketmakers crea una versión RA del monocabo con la forma, ubicación y orientación correctas de cada segmento de laminado visible para el usuario durante el proceso de aplicación. Para ello utilizarán las Microsoft HoloLens [37], con archivos de diseño asistido por computadora (CAD) que los alumnos han desarrollado.

3.2.2. AR Sandbox

AR Sandbox [38] es el resultado de un proyecto desarrollado por el *UC Davis W.M. Keck Center for Active Visualization in the Earth Sciences* (KeckCAVES), conjuntamente con el *UC Davis Tahoe Environmental Research Center*, el Lawrence Hall of Science, y el *ECHO Lake Aquarium and Science Center*.

El proyecto combina aplicaciones de visualización 3D con una exhibición de AR Sandbox para enseñar conceptos de ciencias de la tierra. El entorno de RA permite a los usuarios crear modelos de topografía al dar forma a la arena real, que luego se aumenta en tiempo real mediante un mapa de color de elevación, líneas de contorno topográficas y agua simulada. El sistema enseña conceptos geográficos, geológicos e hidrológicos, como la forma de leer un mapa topográfico, el significado de las curvas de nivel, las cuencas hidrográficas, los diques, etc.

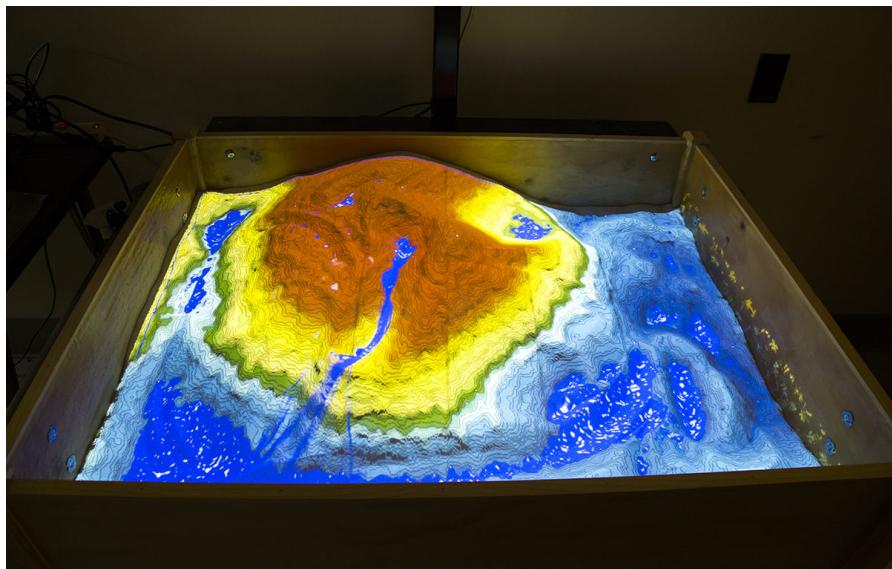


Figura 3.2: Funcionamiento de AR Sandbox.

Capítulo 4

La aplicación ULL-AR

En este capítulo se explicará en detalle la aplicación ULL-AR utilizando su especificación de requisitos y explicando su funcionamiento.

4.1. Requisitos y ventanas de la aplicación

ULL-AR es una aplicación para dispositivos móviles, más concretamente, para dispositivos que utilizan Android como sistema operativo. Se trata de una aplicación diseñada para la comunidad de la Universidad de La Laguna, la cual permita a sus usuarios ubicarse, detectar y reconocer las instalaciones y edificios pertenecientes a la universidad, mediante técnicas de realidad aumentada basadas en la geolocalización. La aplicación contempla diverso tipo de ubicaciones universitarias (facultades, bibliotecas, cafeterías, instalaciones deportivas, etc.). De forma genérica nos referiremos a las mismas como “instalaciones”.

Los requisitos principales de ULL-AR son:

- La aplicación se desarrollará para dispositivos con Android. Se utilizará Android Studio como IDE para su desarrollo.
- Se implementarán técnicas de realidad aumentada basadas en la geolocalización para mostrar al usuario la instalación de la ULL a la cual apunte con la cámara del dispositivo.
- Las instalaciones de la ULL, junto a su información correspondiente, estarán ubicadas en un base datos en la nube. El servidor que se conecte con esta base de datos también deberá estar en la nube.

4.1.1. Especificación detallada de los requisitos

La aplicación se iniciará una “Splash Screen” [39] o pantalla de inicio con el logo de la Universidad de La Laguna. Esta pantalla dará paso a una ventana de *Inicio de sesión*.

Para poder utilizar ULL-AR, el usuario ha de poseer una cuenta de correo institucional de la ULL. Este correo ha de tratarse de una cuenta de GSuite [40] de Google con el formato: “@ull.edu.es”. Sin ella no se podrá acceder a la aplicación.

Una vez autenticado con éxito, se accederá a una ventana de *Inicio* en la que aparecerá un acceso directo a las ventanas de *Mapa ULL* y *Navegación en modo RA* que se explicará más adelante. A su vez, dispondrá de accesos a enlaces web de interés de la ULL que se abrirán en un navegador externo.

ULL-AR contará con un menú para moverse por las diferentes ventanas de la aplicación. Se implementará un menú deslizante lateral o *Navigation Drawer* [41] ubicado en la parte superior izquierda de la aplicación. Este menú deberá ser simple e intuitivo.

Como accesos en este menú se dispone de las siguientes ventanas o funcionalidades:

- **Inicio:** Ventana principal de la aplicación.
- **Mapa ULL:** Esta ventana contendrá un mapa de la ULL con todas las instalaciones en la base de datos.
- **Navegación en modo RA:** En esta ventana, mediante el uso de la cámara, se identificarán las instalaciones de la ULL a las que el usuario apunte con el dispositivo y permitirá mostrar una ventana con información detallada de las mismas.
- **Todas las instalaciones ULL:** Contiene todas las ubicaciones e instalaciones de la ULL y permitirá al usuario realizar una búsqueda de estas.
- **Configuración:** Permitirá acceder a los ajustes de la aplicación.
- **Cerrar sesión:** Cerrará la sesión actual y devolverá al usuario a la ventana de *Inicio de sesión*.
- **Info:** Información de la aplicación y de su autor.

Cada instalación de la universidad tendrá una ficha de información que será accesible desde una ventana de la aplicación con la siguiente información:

- **Id:** Campo para identificar la instalación.
- **Nombre:** Nombre oficial de la instalación.

- **Ubicación:** La ubicación exacta en la que se encuentra la instalación.
- **Descripción:** Descripción de la instalación con el objetivo de esta y actividades que se desarrollan en ella.
- **Imagen:** Enlace URL de la imagen de la instalación.
- **Lista de enlaces de interés:** Una lista con los enlaces a las instituciones, servicios, departamentos y titulaciones asociadas con esta instalación.

Esta información estará guardada en una base de datos en la nube. Para acceder a esta, se dispondrá de un servidor en la nube que conecte con la base de datos y envíe la información a la aplicación.

4.1.2. Ventanas de la aplicación

Al iniciar ULL-AR, la primera ventana que aparece es la pantalla de inicio o Splash Screen. Tras unos segundos, se carga la ventana de *Inicio de sesión*. Para poder autenticarse el usuario debe poseer una cuenta de correo institucional de la ULL. Se dispondrá de un botón en el centro de la pantalla, que abrirá un cuadro de diálogo en el que poder introducir el correo y contraseña de esta cuenta.

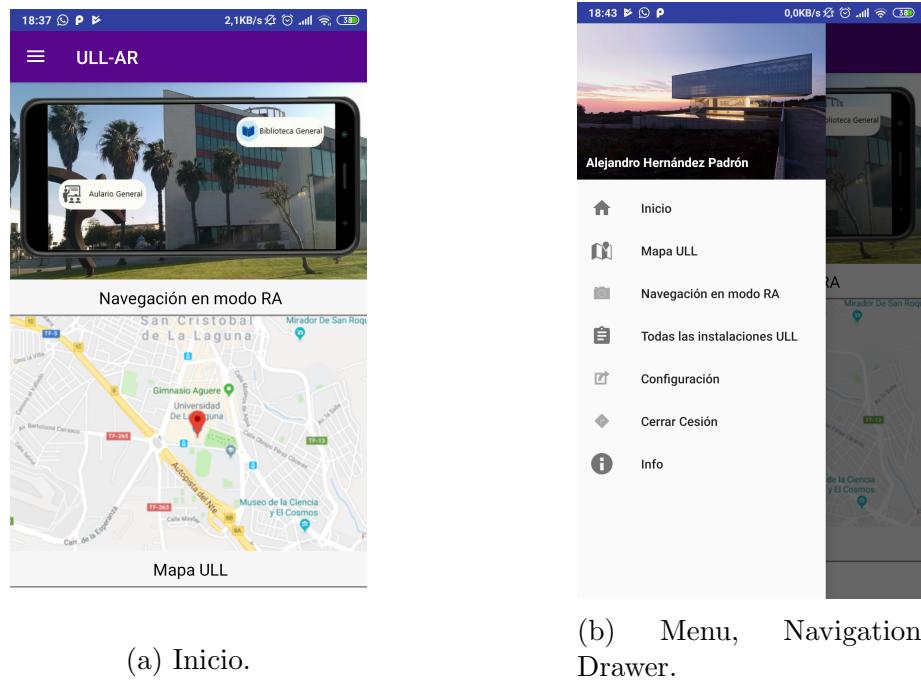


Figura 4.1: Ventana *Inicio* y el Menú de *ULL-AR*.

Cuando el usuario consiga autentificarse con éxito, se abrirá la ventana de *Inicio* (véase Figura 4.1a). En esta aparecerá una lista de accesos directos a las funcionalidades principales de ULL-AR, como son *Navegación en modo RA* y *Mapa ULL*, y una serie de enlaces a sitios web relacionados con la ULL.

En la esquina superior izquierda de ULL-AR se encuentra situado el botón de acceso al menú *Navigation Drawer*. Si se presiona se desplegará el menú que permite al usuario moverse por las distintas ventanas de la aplicación (véase Figura 4.1b).

Si el usuario se desplaza a la ventana de *Mapa ULL* (véase Figura 4.2a) encontrará el mapa generado por la API de Google Maps. En este mapa aparecerán con pines azules las instalaciones de la ULL que están guardadas en la base de datos. Cuando el GPS del dispositivo encuentre la ubicación actual, aparecerá un pin rojo que indicará su posición en el mapa. En la parte inferior y en el centro de la ventana, se dispone de un botón llamado “AR Mode” que permite acceder a la ventana de *Navegación en modo RA*.

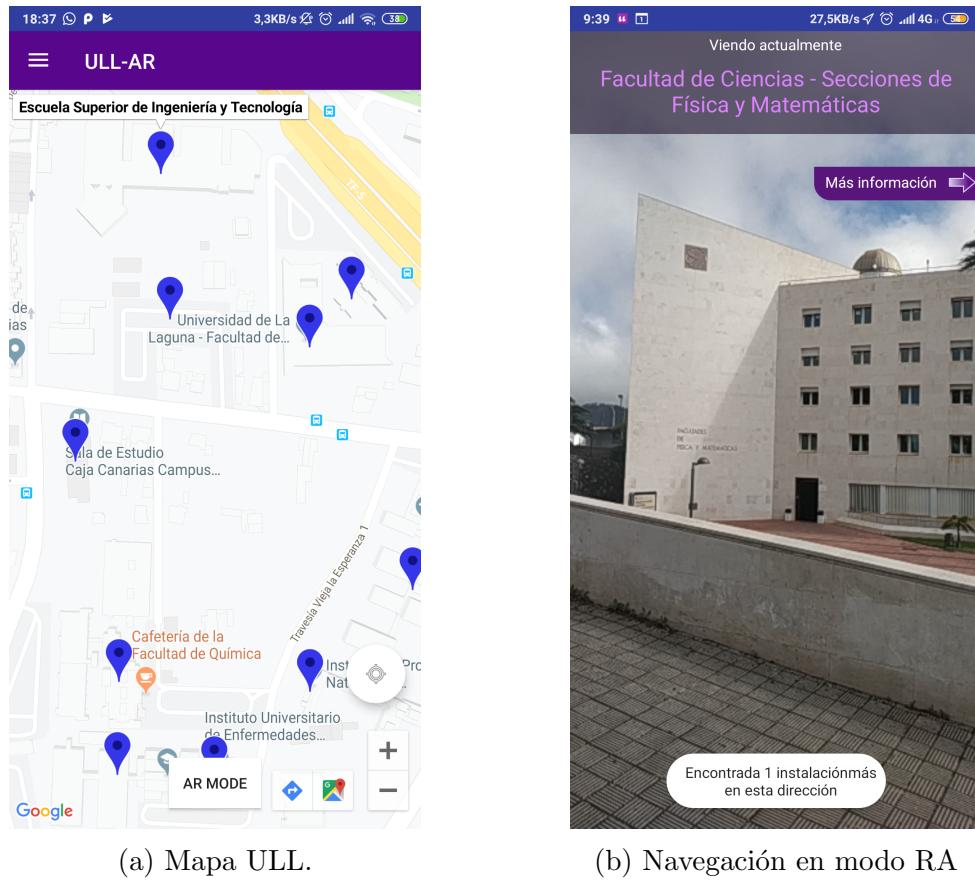


Figura 4.2: Ventana de *Navegación en modo RA* y *Mapa ULL ULL-AR*.

En la ventana de *Navegación en modo RA* se mostrará la imagen obtenida de la cámara del dispositivo. Con ella el usuario podrá ver en la pantalla la instalación a la que apunte con el dispositivo. Encima de esta imagen se mostrará un texto en el que aparecerán dos mensajes: uno informativo para indicar al usuario que apunte a alguna instalación de la ULL y una vez que el usuario se encuentre apuntando a una instalación, el mensaje anterior se cambiará por el nombre de la instalación y, al mismo tiempo, aparecerá un pequeño botón debajo del texto que llevará al usuario a la ventana de *Información de la instalación* (véase Figura 4.3b), con la información de dicha instalación. Por último, en la parte inferior se mostrará un botón que indicará si se han encontrado más instalaciones aparte de la que se muestra en la parte superior. Este botón permitirá acceder a una ventana con una lista de estas instalaciones.

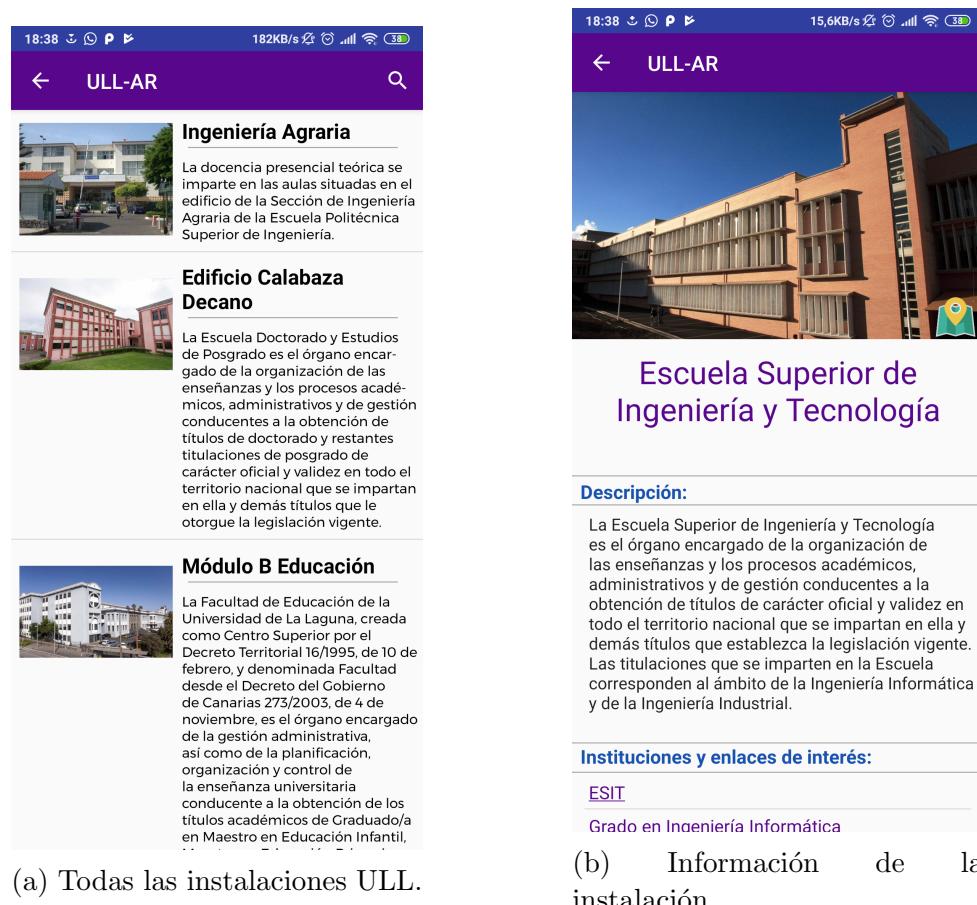


Figura 4.3: Ventanas de *Todas las instalaciones ULL* e *Información de la instalación de ULL-AR*.

A través del menú de la aplicación se puede acceder a la ventana de *Todas las instalaciones ULL* (véase Figura 4.3a). Aquí se mostrarán todas las instalaciones de la ULL que se encuentran en la base de datos. Además, se podrá hacer una búsqueda de cualquier instalación en la barra superior de la aplicación. Si se selecciona cualquiera de estas instalaciones se desplegará una ventana con la información detallada de la instalación.

En la ventana *Información de la instalación* (véase Figura 4.3b) se dispondrá la información perteneciente a cada instalación. Aquí se mostrará una imagen de ésta, nombre y descripción de la instalación y una lista de enlaces con los servicios, secretarías, titulaciones y departamentos asociadas con la misma. Se dispone de un botón en la parte inferior de la imagen de la instalación, que abrirá la ruta a su ubicación en la aplicación de Google Maps para permitir al usuario llegar a ella.

Por último, desde el menú se podrá acceder a las dos últimas ventanas de la aplicación. Estas son la ventana *Configuración* y la ventana *Información*.

En la ventana de *Configuración* (véase Figura 4.4a) se encuentran los ajustes de la aplicación. En ella se permite al usuario decidir si desea realizar una búsqueda, en la ventana de *Mapas* y de *Navegación en modo RA*, de las instalaciones que se encuentran en el área entre dos circunferencias. El centro de estas circunferencias será la ubicación actual del dispositivo.

La ventana *Información* (véase Figura 4.4b) muestra información básica de la aplicación como el nombre, versión, correo de contacto, autor y objetivo e información del desarrollo de la aplicación *ULL-AR*.

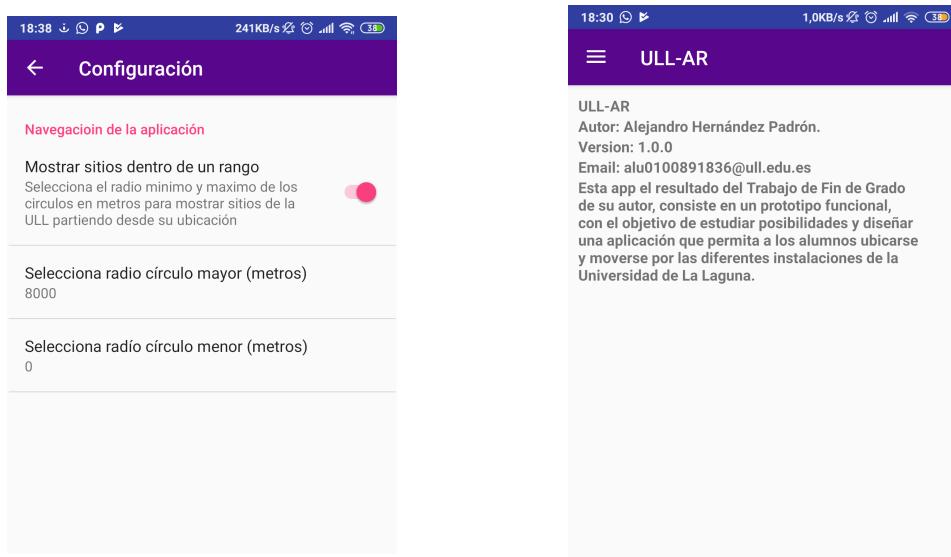


Figura 4.4: Ventanas de *Configuración* e *Información* de *ULL-AR*.

4.2. Inicio de ULL-AR

Para comenzar a desarrollar aplicación, primero, se necesita crear un proyecto nuevo en Android Studio. Este proyecto se nombra con el nombre de “ULL_Navigation” y se siguen los pasos del IDE para acabar de crear el proyecto. Este proyecto contendrá el código de la aplicación “ULL-AR”.

A continuación, se explicarán en detalle el funcionamiento e implementación de las primeras ventanas que aparecen cuando se inicia la aplicación.

4.2.1. Ventana inicial

La primera ventana que aparece en la aplicación es la *Splash Screen*. El objetivo de esta ventana es dar una mejor apariencia a la aplicación e informar al usuario que la aplicación se está iniciando y cargando.

En esta ventana se encuentra el logotipo de ULL-AR en el centro de la pantalla. Este logo se ha diseñado por medio del editor de imágenes online Pixlr [42]. Gracias a este editor se ha podido crear un logotipo simple combinando el icono de la marca de la ULL y el nombre de la aplicación.

En un principio la velocidad de carga y transición a la siguiente ventana de la aplicación se hacía de forma inmediata, debido a que los recursos necesarios para el inicio de la aplicación son escasos y no tardan en cargarse. Por lo tanto, para poder visualizarla correctamente se utilizó un temporizador de tres segundos, para que posteriormente, se lance la siguiente ventana, la cual corresponde a la ventana de *Inicio de sesión*.

Para la implementación de esta ventana *Splash Screen* se ha de configurar primero el fichero `AndroidManifest.xml`. Este fichero proporciona información esencial sobre la aplicación al sistema Android, información que el sistema debe tener para poder ejecutar el código de la aplicación. Aquí se le indica al sistema Android la ventana o “activity” [43] que inicia la aplicación (véase Listado 4.1). Además, a esta ventana se le indica el tema del activity que contendrá el logotipo de ULL-AR en el centro de la pantalla.

```

1 ...
2 <activity
3     android:name=".Activities.SplashActivity"
4     android:theme="@style/SplashScreen">
5     <intent-filter>
6         <action android:name="android.intent.action.MAIN" />
7         <category android:name="android.intent.category.LAUNCHER" />
8     </intent-filter>
9 </activity>
10 ...

```

Listado 4.1: Fichero `AndroidManifest.xml`, activity que inicia la aplicación.

Al mismo tiempo en el archivo `styles.xml` (véase Listado 4.2), se le indica que el archivo `splash_ull.xml` (véase Listado 4.3) se encargará de colocar el color del fondo y el logotipo de la aplicación en el fondo de la pantalla.

```

1 ...
2     <style name="SplashScreen" parent="Theme.AppCompat.NoActionBar">
3         <item name="android:windowBackground">@drawable/splash_ull</item>
4     </style>
5 ...

```

Listado 4.2: Fichero `styles.xml`, estilo de la *Splash Screen*.

```

1 <layer-list xmlns:android="http://schemas.android.com/apk/res/android">
2     <item android:drawable="@color/colorULL"/>
3     <item>
4         <bitmap
5             android:src="@drawable/splash_icon"
6             android:gravity="center"/>
7     </item>
8 </layer-list>

```

Listado 4.3: Fichero `splash_ull.xml`, configuración del color de fondo y el logotipo de la aplicación.

4.2.2. Ventana de *Inicio de Sesión*

Esta es la ventana que permitirá al usuario autenticarse con su correo institucional de la ULL. Para ello, dado que las cuentas de la ULL son cuentas de Google, se ha utilizado la API de Google para poder realizar la autenticación de forma sencilla y segura.

Requisitos

Para poder integrar la API de Google se necesita integrar los “Servicios de Google” en la aplicación. Para ello se ha de entrar en la consola de Firebase [44] y crear un proyecto con el nombre de **ULL-AR**. Una vez dentro del proyecto, se seleccionará que se quiere integrar Firebase a una aplicación Android. A continuación, se pedirá el nombre del paquete de la aplicación y la clave “SHA1”. Para obtener esta clave, se ha de ejecutar en la consola de Android Studio el siguiente comando:

```
1 $ keytool -list -v -alias androiddebugkey -keystore ~/.android/debug.keystore
```

Con el nombre del paquete y la clave SHA1, se descargará un fichero con la configuración de los Servicios de Google llamado `google-services.json`. Este fichero se ha de colocar en la carpeta `app/` del proyecto.

Por último para poder utilizar los Servicios de Google en **ULL-AR**, se tiene que indicar al fichero `build.gradle` del proyecto en las dependencias (véase Listado

4.4) y al fichero `build.gradle` de la aplicación (véase Listado 4.5) que se quiere utilizar los Servicios de Google.

```

1 ...
2 buildscript{
3     dependencies {
4         // Dependencias de los Servicios de Google
5         classpath 'com.google.gms:google-services:4.0.0'
6     }
7 }
8 ...

```

Listado 4.4: Fichero `build.gradle` del proyecto, dependencias para utilizar los Servicios de Google.

```

1 ...
2 dependencies {
3     // Dependencia necesaria para poder autenticarse con una cuenta de Google
4     implementation 'com.google.android.gms:play-services-auth:15.0.1'
5 }
6 // Plugin de los Servicios de Google
7 apply plugin: 'com.google.gms.google-services'

```

Listado 4.5: Fichero `build.gradle` de la aplicación, dependencias y plugin para utilizar los Servicios de Google.

Con estos requisitos ya se puede comenzar a implementar la ventana que realiza la autentificación con una cuenta de Google.

Implementación

Se empieza creando un nuevo activity y se nombra *LoginActivityULL*. Esto generará un fichero *LoginActivityULL.java* que tendrá asociado fichero *login_ull_activity.xml* con un layout [45]. Este layout será la vista del activity, con el logotipo de la aplicación y un botón para realizar el inicio de sesión con una cuenta de Google. La clase *LoginActivityULL* contiene los métodos y atributos que se encargarán de conectarse con los Servicios de Google para realizar una autentificación con una cuenta de correo de la ULL.

En el Listado 4.6 se muestran todos los métodos y conexiones de la API de Google para poder autenticarse. Cuando se presione el botón “Autentícate con una cuenta de la Universidad” se abrirá un cuadro de diálogo de la API de Google que permitirá al usuario seleccionar la cuenta con la que desee entrar en la aplicación.

En caso de que la cuenta no pertenezca a la ULL, es decir, una cuenta que no tenga el formato del dominio de las cuentas de correo de la ULL terminadas en “@ull.edu.es”, se cerrará la sesión de esta cuenta y se mostrará un mensaje con el tipo de cuenta necesaria para utilizar ULL-AR.

```

1  public class LoginActivityULL extends AppCompatActivity implements ... {
2      private GoogleApiClient googleApiClient;
3      // Metodo que se ejecuta cuando se lanza la ventana
4      protected void onCreate(Bundle savedInstanceState) {
5          ...
6          // Opciones de la autentificacion que se desea realizar
7          GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.
8              DEFAULT_SIGN_IN).requestEmail().build();
9          // Se crea una instancia de la API de google con las opciones
10         googleApiClient = new GoogleApiClient.Builder(this).enableAutoManage(this, this)
11             .addApi(Auth.GOOGLE_SIGN_IN_API, gso).build();
12     }
13     // Manejo de eventos
14     public void onClick(View v) {
15         if (v.getId() == loginButton.getId()) { // Si se presiona loginButton
16             // Se crea y lanza el cuadro de dialogo de Google que permite autenticarse
17             Intent intent = Auth.GoogleSignInApi.getSignInIntent(googleApiClient);
18             startActivityForResult(intent, 777);
19         }
20         // Metodo que obtiene el resultado de la autentificacion
21         protected void onActivityResult(int requestCode, int resultCode, Intent data) {
22             if(requestCode == 777) {
23                 GoogleSignInResult result = Auth.GoogleSignInApi.getSignInResultFromIntent(data);
24                 handleSingInResult(result); // Se maneja el resultado de la autentificacion
25             }
26         }
27         // Metodo que comprueba si ha sido correcta la autentificacion con Google
28         private void handleSingInResult(GoogleSignInResult result) {
29             if(result.isSuccess()== true) { // Exito en el inicio de sesion con Google
30                 String userEmail = result.getSignInAccount().getEmail(); // Correo de la cuenta
31                 if(userEmail.matches("(.*)@ull.edu.es")) { // Se comprueba si es un correo de la ULL
32                     Intent intent = new Intent(this, MainActivity.class); // Se ejecuta la ventana
33                     startActivity(intent); // principal de la aplicacion
34                 }else{ logoutNotULLAccont(); } // No es un correo universitario
35             }else{ ... } // Fallo al conectar con Google
36         }
37         // Metodo que realiza el logout de la cuenta cuando la cuenta no pertenece a la ULL
38         private void logoutNotULLAccont() {
39             Auth.GoogleSignInApi.signOut(googleApiClient).setResultCallback( ... )>() {
40                 ... // Mensaje que indica que el correo no es valido y el tipo de
41                 // correo necesario "aluxxxxxxxx@ull.edu.es"
42             };
43         }
}

```

Listado 4.6: Fichero `LoginActivityULL.java`, código que se encarga de realizar el inicio de sesión del usuario con su correo electrónico.

4.3. Modo de Realidad Aumentada

La técnica de realidad aumentada que se ha implementado en la aplicación es la de realidad aumentada basada en geolocalización. Es decir, a partir de la ubicación y la orientación de un dispositivo, se combina la información correspondiente a esos datos y la imagen que se obtiene de la cámara del dispositivo, para mostrar el resultado por la pantalla. Con estos datos de orientación y ubicación del

dispositivo, junto con las ubicaciones de las instalaciones de la ULL, se podrán hacer los cálculos para identificar ante qué instalación se encuentra el usuario apuntando con la cámara del dispositivo móvil.

El fichero `ARNavigation.java` contiene el activity encargado de: la recogida de los datos de los sensores, conexión con el servidor de la aplicación, manejo de los objetos requeridos para la identificación de las instalaciones y de la visualización de la técnica de realidad aumentada implementada.

4.3.1. Acceso a los sensores

Para ubicar y orientar el dispositivo móvil en el mundo físico, se necesita obtener acceso a diferentes sensores de este: GPS, acelerómetro y magnetómetro.

Para obtener acceso al GPS, se han de solicitar los permisos necesarios lo cual se refleja en el archivo `AndroidManifest.xml` (véase Listado 4.7) y a su vez requerir del usuario el permiso para acceder al GPS del dispositivo móvil con ULL-AR (véase Listado 4.8).

```

1 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
2 <uses-feature android:name="android.permission.LOCATION_HARDWARE" />
3 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Listado 4.7: Fichero `AndroidManifest.xml` del proyecto, permisos para acceder a la ubicación del dispositivo.

```

1 // Si el usuario no ha concedido los permisos para utilizar el GPS en la aplicacion
2 if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
3     != PackageManager.PERMISSION_GRANTED) {
4     // Se solicita el permiso para acceder a los datos de GPS de dispositivo
5     requestPermissions(new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
6                         MY_PERMISSIONS_REQUEST_LOCATION);
7 }
```

Listado 4.8: Código para que el usuario conceda permiso para acceder a la ubicación del dispositivo.

Con los permisos concedidos, bastará con el código mostrado en el Listado 4.9 para empezar a requerir los datos del GPS.

```

1 locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
2 locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 3000, 5, this);
3 locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 3000, 5, this);
```

Listado 4.9: Código para activar el GPS del dispositivo.

Para acceder a la última coordenada registrada por el GPS y así poder ubicar al usuario en el mundo físico, se utilizará la siguiente línea código:

```
1 currentLocation = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
```

Ahora que se tiene acceso al GPS del dispositivo, se necesita conocer su orientación. Para ello se hará uso de la matriz de rotación que Android calcula a partir del acelerómetro y el magnetómetro. Con esta matriz se obtendrá el valor de la brújula magnética del dispositivo, es decir, su orientación con respecto al norte magnético. Este valor será el que se utilizará para identificar hacia donde está orientado el dispositivo en el mundo.

```

1 // Se accede a los sensores del dispositivo
2 mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
3 // Se accede al calculo de la matriz de rotacion que proporciona el valor de la brujula
4   magnetica del dispositivo
5 Sensor compass = mSensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION);
6 // Se escuchan a los cambios del sensor
7 mSensorManager.registerListener(this, compass, SensorManager.SENSOR_DELAY_NORMAL);

```

Listado 4.10: Código para acceder a la última ubicación registrada del GPS.

Un dispositivo Android dispone de 3 ejes: x, y, z. Estos ejes se disponen como se muestra en la Figura 4.5. La variable “compass” está formada por una lista de tres valores. El primero es denominado acimut y se refiere al ángulo de la orientación sobre la superficie de una esfera. Este valor representa el ángulo entre el eje “y” del dispositivo y el norte magnético en grados. Cuando se apunta con el dispositivo hacia el norte el valor del ángulo es 0, al sur es 180°, al este es 90° y al oeste 270°. Para poder trabajar con estos valores se convertirán a radianes. El rango de valores es desde 0 a 2π .

Para reconocer las instalaciones que se encuentran en frente al dispositivo móvil, se escucharán los cambios de los valores brújula o “compass” del dispositivo para actualizar la nueva orientación y realizar los cálculos que permiten identificar las instalaciones (véase Listado 4.11). El objeto “navULL” de la clase “Navigation” será el encargado realizar estos cálculos, los cuales se explicarán a continuación.

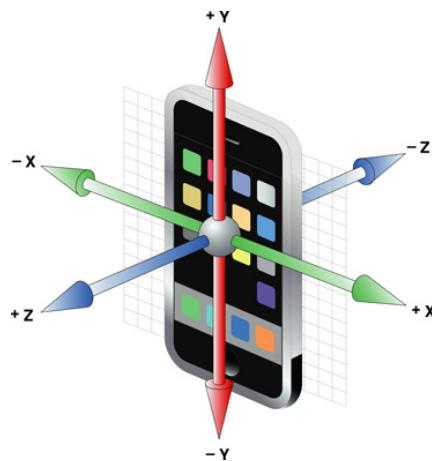


Figura 4.5: Disposición de los ejes de un dispositivo Android.

```

1 // Se escuchan los cambios en el sensor y se hacen los calculos
2 public void onSensorChanged(SensorEvent event) {
3     // Valor del sensor en grados
4     double radians = event.values[0];
5     // Se convierte a radianes
6     radians = Math.toRadians(radians);
7     // Se obtiene la ultima posicion registrada del GPS
8     LatLng lastPosition = getCurrentPos();
9     if (auxpos != null) { // Si la posicion no es nula
10         // Se le pregunta al objeto de la clase ‘‘Navigation’’ las instalaciones
11         // que se encuentran en esa direccion
12         allResultsSites = navULL.whatCanSee(lastPosition, radians);
13     }
14     // Si se obtiene al menos un resultado
15     if (allResultsSites != null) {
16         // Se obtiene la instalacion mas cercana, el indice 0 corresponde a la mas cercana
17         nearSiteResult = allResultsSites.get(0);
18         ... // Se muestra su informacion por pantalla para que usuario conozca la instalacion
19             // a la que se encuentra apuntando y un boton que lanza una con una ficha de
20             // informacion sobre esta
21         if(allResultsSites.size() > 2) {
22             ... // Si se obtiene mas de una instalacion se muestra al usuario el boton
23                 // que indica el numero de instalaciones que se encuentran en la misma
24                 // direccion y lanza una ventana con una lista de estas
25         }
26     } else { ... }
27 }
```

Listado 4.11: Código que se ejecuta cada vez que se registra un cambio en el sensor que calcula la orientación.

4.3.2. Modelos encargados de la navegación

A continuación, se explicarán las clases que intervienen en el proceso de reconocimiento de las instalaciones de la ULL que se encuentran frente al dispositivo móvil.

ULLSite.java

Cada instalación de la ULL se obtiene de una base de datos en formato JSON. Una instalación se representa con un objeto de la clase “ULLSite”. Esta contiene toda su información y los atributos y funciones necesarias para poder trabajar con ella (véase Listado 4.12). El objeto en formato JSON a partir de cual se creará el objeto de cada instalación se puede ver en el Listado 4.13.

En la clase “ULLSite” se encuentran los tres atributos principales para la identificación de las instalaciones: “disToSite”, “dirToSite” y “coneValue”. Estos atributos son calculados en tiempo de ejecución.

```

1 public class ULLSite {
2     private String id; // ID de la instalacion
3     private String name; // Nombre
4     private LatLng mapPoint; // Localizacion geografica
5     private Vector2D point; // Vector2D de la ubicacion
6     private String desc; // descripcion de la instalacion
7     private String imageLink; // imagen de la instalacion
8     // Enlaces de interes a las instituciones, grados, etc.
9     private ArrayList<String> interestPoints; // Nombres
10    private ArrayList<String> interestPointsLink; // Enlaces
11    // Variables necesarias para identificar la direccion de la instalacion
12    private double distToSite = -1; // Distancia a la instalacion
13    private double dirToSite = -1; // Direccion en la que se encuentra
14    private double coneValue = 0; // Valor del cono
15
16    public ULLSite(JSONObject object) {
17        ... // Se construye el objeto con los atributos que se encuentran en el objeto JSON de la
18        // instalacion.
19    }
20    ... // Metodos set() y get() de las variables
}

```

Listado 4.12: Fichero `ULLSite.java`, clase que contiene los atributos de una instalación de la ULL.

```

1 {
2     "id": "esit",
3     "name": "Escuela Superior de Ingenieria y Tecnologia",
4     "position": {
5         "lat": "28.482965",
6         "long": "-16.322003"
7     },
8     "desc": "La Escuela Superior de Ingenieria y Tecnologia es el organo ...",
9     "imageLink": "https://www.ull.es/donde/assets/img/facultades/esit.jpg",
10    "canFind": [
11        {
12            "id": "ESIT",
13            "link": "https://www.ull.es/centros/escuela-superior-de-ingeneria-y-tecnologia/"
14        },
15        ...
16    ]
17 }

```

Listado 4.13: Ejemplo de una instalación de la ULL en la base de datos.

El atributo “disToSite” representa la distancia, en metros, a la que se encuentra la instalación con respecto a la ubicación del dispositivo. El atributo “dirToSite” representa el valor de la brújula del dispositivo, en radianes, en la dirección en la que se encuentra la instalación, es decir, si la ubicación se encuentra hacia el norte con respecto al dispositivo el valor será 0 y si se encuentra hacia el este será $\pi/2$. El atributo “coneValue” representa un cono de visión imaginario cuyo centro es la dirección en la que se encuentra la instalación. Este cono forma un rango de visión alrededor de la dirección de la instalación a partir de la cual si el dispositivo se encuentra apuntando dentro de este rango se considerará que éste está “viendo” la instalación. El valor del atributo “coneValue” variará en función de la distancia

a la que se encuentre el dispositivo de la instalación. Estos atributos junto con el objeto “point” de la clase “Vector2D”, que contiene la ubicación en un plano de coordenadas cartesianas, permitirán realizar los cálculos para identificar si el dispositivo se encuentra apuntando a una instalación o no.

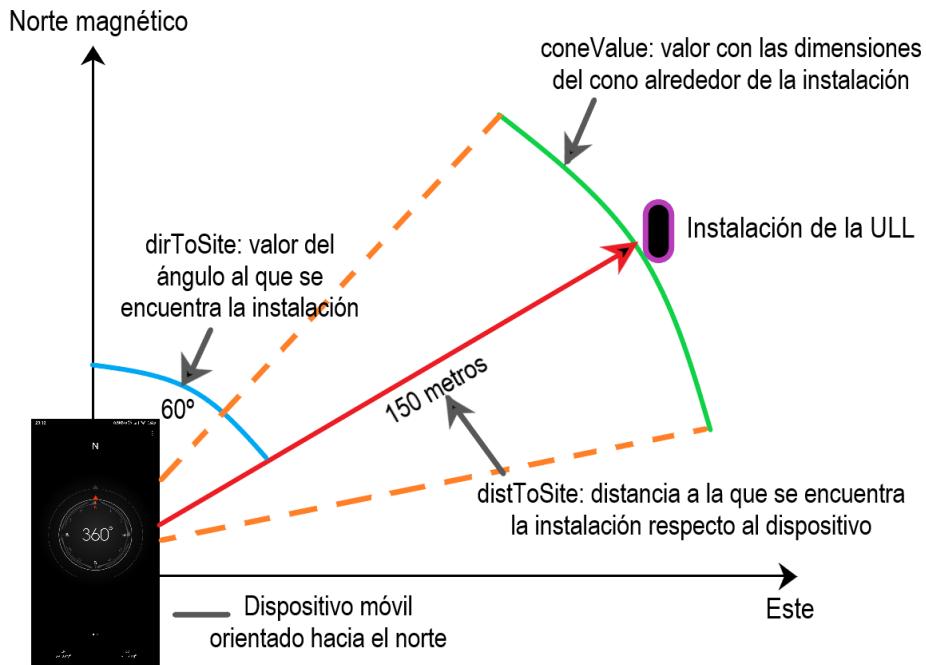


Figura 4.6: Explicación de los atributos “coneValue”, “dirToSite” y “disToSite”.

Navigation.java

La clase “Navigation” es la clase principal encargada de ejecutar todos métodos y realizar los cálculos que permiten identificar las instalaciones. A partir de los datos obtenidos de los sensores y de las ubicaciones de las instalaciones permite identificar la instalación más cercana que se encuentra en la dirección del dispositivo y también el resto de las instalaciones en esta misma dirección. En el Listado 4.14 se muestran los atributos y métodos de la clase.

A continuación se explicará de forma detallada la implementación de los métodos principales de esta clase.

En el Listado 4.15 se encuentra el método que calcula el valor del atributo “coneValue” para una instalación. Este valor se calcula de modo que cuanto más cerca se encuentre el dispositivo de una instalación, mayor sea el rango con el que poder decidir si se encuentra en frente de ella o no y, por el contrario, que este valor sea menor cuando se encuentre lejos de la instalación.

```

1 public class Navigation implements Serializable {
2     // Distancia en metros a partir en la que una instalacion se considera con "cercana"
3     private static final double NEAR_VALUE = 150;
4     // Cuando la distancia de una instalacion sea cercana se utilizaran los valores
5     // *_NEAR para los calculos en caso contrario se utilizaran *_FAR
6     private static final double MAX_CONE_GRADS_NEAR = Math.PI / 2; // Maximo valor del cono
7     private static final double MAX_CONE_GRADS_FAR = Math.PI / 8; // Maximo valor del cono
8     private static final double MIN_CONE_GRADS = Math.PI / 20; // Minimo valor del cono
9     // Las variable SCALE_CONE son valores con los que al escalar las dimensiones de los conos la
10    // aplicacion reconocia las instalaciones de forma precisa
11    private static final double SCALE_CONE_NEAR = 0.0078; // Variable que escala el cono
12    private static final double SCALE_CONE_FAR = 0.00078; // Variable que escala el cono
13    private Location location;
14    // Distancia maxima por defecto para considerar una instalacion en metros
15    private double maxDist = 200;
16    // Distancia minima por defecto para considerar una instalacion en metros
17    private double minDist = 0;
18    private Vector2D currentPos; // Posicion actual del dispositivo
19    private double currentDir; // Direccion actual del dispositivo
20    private ArrayList<ULLSite> allSites = new ArrayList<>(); // Todas las instalaciones
21    // Instalaciones que se encuentran dentro del rango de maxDist y minDist
22    private ArrayList<ULLSite> destSites = new ArrayList<>();
23    // Se construye el array de allSites con todas las instalaciones a partir de un JSON
24    public Navigation(JSONArray jsonULLSitesAux) { ... }
25    // Se realizan los calculos que permiten indentificar las instalaciones
26    public ArrayList<ULLSite> whatCanSee(LatLng currentPosAux, double actualDir) { ... }
27    // Calcula la distancia entre dos ubicaciones geograficas
28    public double getDistanceBetween(Vector2D v1, Vector2D v2) { ... }
29    // Se comprueba si la direccion del dispositivo se encuentra dentro de su cono de
30    // identificacion
31    private boolean isInCone(double directionToSite, double coneValue) { ... }
32    // Sirve para reorientar al norte magnetico, como inicio de rotacion, el angulo dado por
33    // el Vector2D.getAngleRad(Vector2D v)
34    private double recalculaAngVector2D(double angleRad) { ... }
35    // Se invierte el angulo
36    public double invertAng(double rad) { ... }
37    // Se rota -90 el angulo
38    public double rotateRad(double rad) { ... }
39    // Se calcula el valor del cono
40    public double calculateCone(double dist) { ... }
41    ... // Metodos Get() y Set() de los atributos
42 }

```

Listado 4.14: Fichero `Navigation.java`, clase “Navigation” que contiene los atributos y métodos que nos permiten identificar las instalaciones de ULL.

El cálculo del ángulo formado por dos puntos geográficos se realiza con el método “`Vector2D.getAngleRad(Vector2D v2)`”, como se puede ver en el Listado 4.16. Este método calcula el ángulo que se obtiene al transformar el plano de coordenadas rectangulares (distancia en eje “x” y distancia en el eje “y”) generado por la diferencia entre el punto con la ubicación geográfica del dispositivo “`v2`” y el punto con la ubicación geográfica de la instalación contenida en “`getX()`” y “`getY()`”, a coordenadas polares. En las coordenadas polares un punto en el plano se determina por una distancia y un ángulo. Este ángulo que se obtiene será la dirección en la que se encuentra la instalación.

```

1  public double calculateCone(double dist) {
2      // Si es una instalacion "cercana" del dispositivo
3      if (dist <= NEAR_VALUE) {
4          // Se calcula el valor del coneValue restandole al valor maximo las distancia a la
5          // instalacion por la constante SCALE_CONE_NEAR que permite que esta se escale
6          // gradualmente.
7          return MAX_CONE_GRADS_NEAR - dist * SCALE_CONE_NEAR;
8      } else { // Si es "lejana"
9          // Se calcula el valor del coneValue restandole al valor maximo las distancia a la
10         // instalacion por la constante SCALE_CONE_FAR que permite que esta se escale
11         // gradualmente en instalaciones lejanas.
12         double auxCone = MAX_CONE_GRADS_FAR - dist * SCALE_CONE_FAR;
13         if (auxCone < MIN_CONE_GRADS) {
14             return MIN_CONE_GRADS;
15         } else {
16             return auxCone;
17         }
18     }
19 }
```

Listado 4.15: Código para calcular el *coneValue* de identificación de cada instalación.

```

1  // Metodo que calcula el angulo que se obtiene al transformar el rectangulo formado por este
2  // punto y el punto v2 a coordenadas polares
3  public double getAngleRad(Vector2D v2) {
4      double dx = v2.getX() - getX(); // Se calculan las distancias en el eje x e y
5      double dy = v2.getY() - getY(); // se obtiene el rectangulo formado por estos dos puntos
6      double radian = Math.atan2(dy, dx); // Se realiza la arcotangente para calcular el angulo
7      // del rectangulo formado
8      return radian; // Se devuelve el resultado
9 }
```

Listado 4.16: Método que calcula el ángulo formado por dos puntos.

A este ángulo, hay que aplicarle unas transformaciones para que se ajuste a la orientación del norte magnético como el inicio de la rotación (norte magnético = 0°), para ello, el método “Navigation.recalculeAng(double angleRad)” se encarga de la correcta reorientación (véase Listado 4.17).

```

1  private double recalculeAng(double angleRad) {
2      double aux = rotateRad(angleRad); // Rota -pi/2
3      aux = invertAng(aux);           // Se invierte el angulo
4      return aux;                   // Se devuelve el resultado
5 }
```

Listado 4.17: Método que recalcula en ángulo para orientarlo en función del norte magnético.

El método “whatCanSee” (véase el Listado 4.18) es el método principal que se encargará, a partir de los datos de ubicación y dirección del dispositivo, de identificar qué instalaciones se encuentran en frente del dispositivo y cuál es la más cercana. Para ello, en un primer paso, se calculan las distancias de las instalaciones y las que se encuentran a una distancia mayor que “maxDist” y menor

que “minDist” serán descartadas, para poder reducir el número de instalaciones a identificar. A continuación, para cada instalación anterior se calcula su dirección, posición y el valor del cono, con respecto a la ubicación del dispositivo. Si la instalación se encuentra orientada dentro del cono que se forma en la dirección de la instalación, esta instalación se considerará como un posible resultado y se añadirá a la variable con una lista de estos resultados, “result”. De los posibles resultados, la instalación más cercana al dispositivo será la instalación ante la que teóricamente se encuentra el dispositivo y guardará ésta en el principio de esta lista.

```

1 // Metodo principal que se encarga identificar las instalaciones en frente del dispositivo
2 // Recibe la posicion y orientacion actual del dispositivo
3 // Devuelve la lista de instalaciones en esa direccion indicando cual es la mas cercana
4 public ArrayList<ULLSite> whatCanSee(LatLng currentPosAux, double actualDir) {
5     currentPos.set(actualPos.longitude, actualPos.latitude); // Posicion actual del dispositivo
6     currentDir = actualDir; // Orientacion del dispositivo
7     int id = -1; // Indice de la instalacion mas cercana en frente del dispositivo
8     double nearSiteDist = maxDist; // Distancia maxima valida para identificar un instalacion
9     // Array a devolver con las instalaciones encontradas
10    ArrayList<ULLSite> result = new ArrayList<>();
11    // Se calculan todas las instalaciones que se encuentran entre maxDist y minDist
12    for (int i = 0; i < allSites.size(); i++) {
13        double distToSite = getDistanceBetween(currentPos, allSites.get(i).getPoint());
14        if ((distToSite < maxDist) && (distToSite > minDist)) {
15            destSites.add(allSites.get(i));
16        }
17    }
18    // Para cada una las instalaciones dentro del rango anterior
19    for (int i = 0; i < destSites.size(); i++) {
20        // Se calcula la direccion, distancia y valor del cono de cada instalacion a partir
21        // de la actual ubicacion del dispositivo
22        double dirToSite = recalculeAng(currentPos.getAngleRad(destSites.get(i).getPoint()));
23        double distToSite = getDistanceBetween(currentPos, destSites.get(i).getPoint());
24        double coneValue = calculateCone(distToSite);
25        // Se comprueba si el dispositivo esta orientado hacia dentro del cono que se forma en
26        // la direccion de la instalacion
27        if (isInCone(dirToSite, coneValue)) {
28            // Se guarda los valores calculados anteriormente en el objeto ULLSite del array
29            destSites.get(i).setConeValue(coneValue);
30            destSites.get(i).setDirToSite(dirToSite);
31            destSites.get(i).setDistToSite(distToSite);
32            result.add(destSites.get(i)); // Se guarda esta instalacion como resultado
33            if (nearSiteDist > distToSite) { // Se comprueba si es la instalacion mas cercana
34                nearSiteDist = distToSite; // Si lo es, se actualiza la distancia mas cercana
35                id = i; // Se guarda el indice de la instalacion mas cercana
36            }
37        }
38    }
39    if (id != -1) { // Si se ha encontrado alguna instalacion
40        result.add(0, destSites.get(id)); // La instalacion mas cercana la se guarda la primera
41        return result; // Se devuelve las instalaciones
42    } else
43        return null; // Si no se encuentra ninguna instalacion
44}

```

Listado 4.18: Método principal que realiza el cálculo que permite reconocer las instalaciones que se hallan frente al dispositivo móvil.

4.3.3. Obtención de la información

Como se ha comentado anteriormente, toda la información perteneciente a las instalaciones de la ULL estará en una base de datos. Además, para comunicarse con ésta, se dispone de un servidor que atienda a las peticiones de la aplicación, se conecte con esta base de datos y envíe la información. Más adelante, en la Sección 4.8, se explicará detalladamente el funcionamiento de esta base de datos y servidor. Por ahora, se explicará cómo se realiza la conexión con el servidor y el tipo de respuesta que se obtiene.

Para poder realizar una petición al servidor se utilizará la clase “GetData”. Esta clase formaliza una petición con la URL que se le pase como parámetro al método “doInBackground” y devuelve una cadena de caracteres con la respuesta del servidor. En el Listado 4.19 se puede ver cómo funciona este método.

```

1 public class GetData extends AsyncTask<String, Void, String> {
2     protected String doInBackground(String... strings) {
3         StringBuilder result= new StringBuilder(); // Resultado a devolver de la peticion
4         try {
5             URL url = new URL(strings[0]); // url a la que se realiza la peticion
6             HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
7             urlConnection.setConnectTimeout(10000); // Tiempo de espera maximo del conexion
8             urlConnection.setRequestMethod("GET"); // Metodo de la conexion
9             urlConnection.setRequestProperty("Content-Type", "application/json"); // Contenido
10            urlConnection.connect(); // Se realiza la conexion
11            ... // Cuando se recibe la respuesta se lee, se crea y se devuelve un string
12            // con su contenido
13        }catch (IOException e) { ... }
14        return "Error"; // Si falla la conexion
15    }
16}

```

Listado 4.19: Fichero `GetData.java`, código encargado de la conexión con el servidor y manejo de la respuesta.

En el Listado 4.20 se encuentra el método ejecutado en la clase “ARNavigation” para realizar la conexión con el servidor para obtener la respuesta de la base de datos con todas las instalaciones de ULL. Posteriormente, se creará una instancia de la clase “Navigation” con todas las instalaciones de la base de datos en formato JSON. Este objeto se denominará con el nombre de “navULL” y será el objeto encargado de identificar las instalaciones y trabajar con ellas.

Los últimos datos que se necesitan para trabajar con el objeto “navULL”, son los valores de “maxRadius” y “minRadius”, los cuales el usuario puede editar en la ventana de *Configuración* y que se puede acceder desde cualquier activity gracias a las “Shared Preferences”. Las Shared Preferences son un conjunto de datos accesible desde cualquier activity de la aplicación y que se utiliza para guardar los ajustes del usuario. En el método “getRadius()” del Listado 4.21 se puede ver como se accede a estos datos.

```

1  private void getSitesFromDB() {
2      try{
3          GetData getSites = new GetData();
4          String sites = getSites.execute("https://server-ull-ar.herokuapp.com/api/ull-sites").
5              get();
6          JSONArray array = new JSONArray(sites);
7          // Se crea una instancia de la clase "Navigation" con todas las instalaciones
8          // Este es el objeto encargado de encontrar las instalaciones
9          navULL = new Navigation(array);
10     } catch (JSONException e) {...}
11 }
```

Listado 4.20: Método que conecta con el servidor y recibe la respuesta con todas las instalaciones de la base de datos.

```

1  private void getRadius() {
2      try {
3          // Se obtiene los valores configurables "maxRadius" y "minRadius" en la ventana de
4          // "Configuracion"
5          settingsPref = PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
6          String auxMaxRadius = settingsPref.getString("maxRadius", "null");
7          String auxMinRadius = settingsPref.getString("minRadius", "null");
8          navULL.setMaxDist(Integer.parseInt(auxMaxRadius)); // Se guarda el valor "maxRadius"
9          navULL.setMinDist(Integer.parseInt(auxMinRadius)); // Se guarda el valor "minRadius"
10     }catch (Exception e) { ... }
11 }
```

Listado 4.21: Fichero ARNavigation.java, código que se encarga de obtener los valores de “maxDist” y “minDist” del objeto “navULL”.

4.3.4. Visualización

El activity “ARNavigation” será el encargado de mostrar la técnica de realidad aumentada por geolocalización en la pantalla de dispositivo móvil. Este activity hereda de la clase “ARActivity” perteneciente al SDK de Kudan para Android Studio. Este SDK permite el reconocimiento de objetos e imágenes a través de la cámara del dispositivo y queda como recurso para una futura ampliación de la funcionalidad de la aplicación. El uso principal de este SDK en ULL-AR, es el acceso que otorga a cámara del dispositivo como vista principal del activity. Es decir, permite visualizar en la ventana lo que se está observando con la cámara.

Para poder hacer uso del SDK de Kudan se tiene que descargar el SDK y configurar la “ARAPIkey” específica para el proyecto. Ambos se pueden encontrar en la página oficial de Kudan [19].

El fichero aractivity.xml (véase Listado 4.22) contiene el layout con la información a mostrar al usuario cuando esté en frente de una instalación. Este layout muestra al usuario en la parte superior de la ventana, el nombre de la instalación ante la que se encuentra y un botón que le permite acceder a una ventana con la ficha de información de la instalación. Además incorpora un botón

en la parte inferior que le indica el número de instalaciones adicionales que se encuentran en la misma dirección. Al pulsar este botón se despliega una ventana con una lista de estas instalaciones.

```

1 <android.support.constraint.ConstraintLayout ...>
2     ... <!-- Imagenes de fondo de los TextView para facilitar la lectura de los mensajes-->
3     <!-- Texto informativo de que se ha encontrado una instalacion -->
4     <TextView android:id="@+id/seenText" android:text="Viendo actualmente" .../>
5     <!-- Nombre de la instalacion encontrada -->
6     <TextView android:id="@+id/ullSiteText" android:text="Facultad de Ciencias ..." .../>
7     <!-- Texto por defecto cuando no se apunta a ninguna instalacion -->
8     <TextView android:id="@+id/.." android:text="Apunte a una instalacion ..." .../>
9     <!-- Boton que accede a la ventana con la informacion de la instalacion -->
10    <Button android:id="@+id/moreInfoButton" .../>
11    <!-- Texto del boton "moreInfoButton" -->
12    <TextView android:id="@+id/moreInfoText" android:text="Mas informacion" .../>
13    <!-- Boton que despliega la lista con el resto de instalaciones encontradas -->
14    <Button android:id="@+id/..." android:text="Encontradas X instalaciones" .../>
15 </android.support.constraint.ConstraintLayout>
```

Listado 4.22: Fichero aractivity.xml, layout del activity “ARNavigation”.

Los elementos del layout se mostrarán u ocultarán en función de si el usuario se encuentra en frente de una instalación o no. Como se puede ver en el Listado 4.11, se muestra cómo y cuándo se alterna la información del layout.

Por último, se ha de configurar el funcionamiento de los botones “moreInfoButton” y “moreSitesButton” (véase Listado 4.23), que mostrarán, respectivamente, una ventana con la ficha de información de la instalación que se está viendo actualmente y una ventana con una lista que contiene el resto de las instalaciones que se encuentran en la misma dirección.

```

1 public void onClick(View v) {
2     if(v.getId() == moreSitesButton.getId()) { // Si coincide
3         ... // Se inicializa el activity que muestra una lista de las instalaciones adicionales
4         ArrayList aux = new ArrayList(moreResultsSites.subList(1, moreResultsSites.size()-1));
5         SitesArray sitesArray = new SitesArray(aux);
6         // Se le pasa como "extra" una la lista con las instalaciones a mostrar
7         intent.putExtra("sitesToShow", sitesArray);
8         startActivityForResult(intent); // Se inicia el activity
9     }
10    if(v.getId() == moreInfoButton.getId()) { // Si coincide
11        ... // Se inicializa el activity que muestra la descripcion de la instalacion
12        ULLSiteSerializable actualULLSite = new ULLSiteSerializable(nearSiteResult);
13        // Se le pasa como "extra" el objeto que contiene la informacion de la instalacion
14        intent.putExtra("actualULLSite", actualULLSite);
15        startActivityForResult(intent); // Se inicia el activity
16    }
17 }
```

Listado 4.23: Fichero ARNavigation.java, código para manejar los eventos de los botones.

4.4. Fragmentos

Los fragmentos [46] actúan como una sección modular de un activity que tiene su ciclo de vida propio, es decir, recibe sus propios eventos de entrada y que se pueden agregar o quitar mientras el activity se esté ejecutando. Se han utilizado estos fragmentos para mostrar el contenido de ciertas ventanas de la aplicación. La principal ventaja que aportan los fragmentos es la facilidad que otorgan para intercambiar estos fragmentos dentro de un mismo activity y la mejora de rendimiento que se obtiene con respecto a creación de un activity para cada ventana que se quiera en la aplicación.

4.4.1. MapsFragment

Este fragmento contiene el mapa generado por la API de Google Maps. Para poder utilizar esta API, se tiene que acceder a la consola de desarrolladores de Google [47]. En ella se tiene que crear un proyecto con el nombre de ULL-AR, para después habilitar la API de “Maps SDK for Android”. Una vez habilitada se otorgará una clave que se encuentra en el fichero `app/res/values/google_maps_api.xml` (véase Listado 4.24).

```

1 <resources>
2   <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">API_Maps<
3     /string>
</resources>
```

Listado 4.24: Fichero `google_maps_api.xml`.

A su vez, se debe añadir la siguiente línea a las dependencias de fichero `build.gradle` de la aplicación:

```
1 implementation 'com.google.android.gms:play-services-maps:15.0.1'
```

Con esto ya se puede utilizar la API de Google Maps en la aplicación. Ahora toca implementar el fragmento que contendrá la vista de Google Maps.

Se crea un fragmento en Android Studio con el nombre de `MapsFragment`. Se generará un fichero llamado `MapsFragment.java` y un layout asociado, `fragment_maps.xml`.

La clase “MapsFragment” (véase Listado 4.25) se encargará de: obtener la ubicación, dibujar en el mapa los marcadores de las instalaciones de ULL, obtener la ubicación del dispositivo y, si se activa en los ajustes, de obtener las dos circunferencias, cuyo centro será la ubicación del dispositivo y que representarán un rango de búsqueda de las instalaciones. Este rango funcionará a modo de que solo aparezcan las instalaciones que se encuentran en el espacio entre las dos circunferencias. En esta clase se harán uso de los métodos que ya se mostraron para obtener la ubicación GPS, las instalaciones de la base de datos y los datos necesarios guardados en las Shared Preferences.

```

1  public class MapsFragment extends Fragment implements OnMapReadyCallback ...{
2      private MapView mapView; // Vista que contiene el mapa de Google Maps
3      private GoogleMap goMap; // API de google maps para modificar el mapa
4      ... // Resto de variables
5      private Button buttonARStart; // Boton que lanza el ARNavigation.class
6      private ArrayList<ULLSite> allSites= new ArrayList<ULLSite>(); // Todos las instalaciones
7      private ArrayList<Marker> allMarkers = new ArrayList<Marker>(); // Sus marcadores
8      private Circle circleMax; // Circulo mayor dibujado en el mapa
9      private Circle circleMin; // Circulo menor
10     private int maxRadius; // Radio del circulo mayor
11     private int minRadius; // Radio del circulo menor
12     private boolean showRadius; // Se indica si se dibujan los circulos o no
13     // Metodo que se ejecuta cuando se lanza el fragment
14     public View onCreateView(LayoutInflater inflater, ViewGroup container ...) {
15         View rootView = inflater.inflate(R.layout.fragment_maps...); // Creamos la vista principal
16         getRadius(); // Se obtienen los radios de la settingsPref
17         getSitesFromDB(); // Se obtienen las instalaciones de la base de datos
18         return rootView; // Se devuelve la vista
19     }
20     // Metodo que se ejecuta cuando la vista ya esta creada
21     public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
22         mapView = (MapView) rootView.findViewById(R.id.mapView); // Se busca el MapView del layout
23         ... // Se configura el mapa y se pide el mapa de Google Maps de la zona actual
24     }
25     // Metodo que se ejecuta cuando el mapa de Google Maps ya este creado
26     public void onMapReady(GoogleMap googleMap) {
27         goMap = googleMap; // Se guarda el objeto que tiene el mapa
28         LatLng currentPos = getCurrentPos(); // Se obtiene la posicion GPS
29         actualPosMarker = goMap.addMarker(new MarkerOptions()); // Marcador en la ubicacion GPS
30         drawAllSites(); // Se dibujan todas las instalaciones de la base de datos
31         showSitesOnRadius(getCurrentPos()); // Se dibujan los circulos circleMax y circleMin
32         ...
33     }
34     // Metodo que actualiza los dibujos de los mapas a la ubicacion actual cuando cambia
35     public void onLocationChanged(Location location) {
36         LatLng position = getCurrentPos(); // Se obtiene la posicion del gps
37         drawActualPos(); // Se dibuja la posicion actual
38         redrawCircles(position); // Se actualizan los circulos
39         showSitesOnRadius(position); // Se muestran las instalaciones dentro del los circulos
40     }
41     // Metodo que muestra las ubicaciones entre "circleMax" y "circleMin"
42     private void showSitesOnRadius(LatLng position) {
43         for (int i = 0; i < allSites.size(); i++) { // Para todas las instalaciones
44             ... // Se calcula la distancia entre la instalacion y la ubicacion GPS
45             if(distance > minRadius && distance < maxRadius) { // Si la instalacion esta dentro del
46                 ...
47             } else { ... } // Si no, se oculta
48         }
49     }
50     private void getRadius() { ... } // Se obtienen maxRadius y minRadius de las SharedPrefences
51     private void getSitesFromDB() { ... } // Se obtienen las instalaciones de la BD
52     private LatLng getCurrentPos() { ... } // Posicion actual del GPS
53     private void drawAllSites() { ... } // Se crea un marcador en el mapa para cada instalacion
54     private void redrawCircles(LatLng position) {...} // Se actualizan los centros de los circulos
55     .... // Resto de metodos necesarios
56 }
```

Listado 4.25: Fichero MapsFragment.java, métodos principales.

El fichero fragment_maps.xml contiene la vista del mapa de Google Maps en el

cual se dibujarán los marcadores y los círculos que ya se han comentado. Además incorpora un botón con el nombre de “AR Mode” que ejecutará la ventana de *Navegación en modo AR*.

4.4.2. HomeFragment

El fragmento “HomeFragment” es la primera vista que se encuentra cuando se inicia sesión con éxito en la aplicación. Para el diseño de este fragmento se decidió por utilizar el modelo de “RecyclerView” [48]. Este modelo permite la visualización de listas de elementos o ítems de una forma más flexible, permitiendo configurar la vista de cada ítem mediante el uso de adaptadores. Un adaptador permite crear las vistas de cada ítem de una lista a partir del contenido de los atributos de cada uno de ellos.

El contenido de cada ítem vendrá dado por la clase “ItemHome”. Los atributos de esta clase serán: una imagen, un nombre, una variable que indica si es o no un enlace web del navegador y la URL del enlace.

```

1 public class ItemHomeAdapter extends RecyclerView.Adapter<ItemHomeAdapter.ViewHolder> {
2     private List<ItemHome> items;
3     private int layout;
4     private OnItemClickListener itemClickListener;
5     public ItemHomeAdapter(List<ItemHome> items, int layout, OnItemClickListener listener) {
6         ... // Se asignan a los atributos con sus respectivos valores
7     }
8     // El parametro @ViewGroup parent contendra la vista de todos los items
9     public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
10        View v = LayoutInflater.from(parent.getContext()).inflate(layout, parent, false);
11        ViewHolder vh = new ViewHolder(v); // Objeto ViewHolder
12        return vh; // Se devuelve la vista
13    }
14    // Se enlaza cada item con su objeto ViewHolder
15    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
16        holder.bind(items.get(position), itemClickListener);
17    }
18    // Clase que hereda de RecyclerView.ViewHolder que sera la vista de cada item
19    public static class ViewHolder extends RecyclerView.ViewHolder {
20        public TextView itemName;
21        public ImageView itemImage;
22        ...
23        public ViewHolder(View itemView) { // Constructor con el layout
24            ... // Se enlaza la informacion de los objetos con su layout
25            this.itemName = itemView.findViewById(R.id.textView_home_item);
26            this.itemImage = itemView.findViewById(R.id.imageView_home_item);
27        }
28        // Se asigna a cada vista el contenido de su item correspondiente
29        public void bind(final ItemHome itemHome, final OnItemClickListener listener) {
30            this.itemName.setText(itemHome.getName()); // Se asigna el texto
31            ... // Se asigna la ruta de la imagen
32            itemView.setOnClickListener(new View.OnClickListener() { ... }); // Listener del item
33        }
34    }
}

```

Listado 4.26: Fichero `ItemHomeAdapter.java`, clase que construye las vistas de los ítems.

Con una lista de estos ítems, el adaptador configurará la vista de cada ítem y luego se incorporarán a la vista de RecyclerView. La clase encargada de este procedimiento se llamará “ItemHomeAdapter” y hereda del adaptador “RecyclerView.Adapter” (véase Fichero 4.26). Los atributos de esta clase son: una lista de ítems, el layout con el diseño de cada ítem y un objeto “OnItemClickListener” que manejará los eventos de cada ítem. Dentro de esta clase, se tiene una clase “ViewHolder” que hereda de “RecyclerView.ViewHolder” y se encargará de enlazar el contenido de cada ítem con su layout. El layout con la vista de cada ítem se encuentra en el fichero `adapter_item_home.xml`. Este layout contendrá la imagen del ítem en la parte superior y el nombre inferior.

```

1 public class HomeFragment extends Fragment implements View.OnClickListener {
2     private List<ItemHome> itemsHome; // Lista de items a representar en la vista
3     private RecyclerView recyclerView; // Vista que contendrá todos los items
4     private RecyclerView.Adapter homeAdapter; // Contendrá el objeto de la clase ItemHomeAdapter
5     private RecyclerView.LayoutManager homeLayoutManager; // Layout del RecyclerView
6     public View onCreateView(LayoutInflater inflater... ) { // Primer metodo que se ejecuta
7         setAllItems(); // Se instancia los items
8         // Se infla la vista con el layout fragment_home.xml
9         return inflater.inflate(R.layout.fragment_home, container, false);
10    }
11    // Se instancia cada item con su nombre, imagen y si es un link externo y su url
12    private void setAllItems() {
13        ArrayList<ItemHome> auxItems = new ArrayList<ItemHome>();
14        auxItems.add(new ItemHome("Navegacion en modo RA", "home_ar_inicio", false, null));
15        auxItems.add(new ItemHome("Pagina de la ULL", "home_ull_site", true, "www.ull.es"));
16        ... // Resto de items
17        itemsHome = (List) auxItems; // Se guardan en atributo itemsHome
18    }
19    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
20        recyclerView = getActivity().findViewById(R.id.recyclerView_Home); // RecyclerView
21        homeLayoutManager = new LinearLayoutManager(getContext()); // Se asigna un LinearLayout
22        // Se instancia el objeto ItemHomeAdapter
23        homeAdapter = new ItemHomeAdapter(itemsHome, R.layout.adapter_home_item,
24            new ItemHomeAdapter.OnItemClickListener() {
25                // Se indica el comportamiento cuando se selecciona un item
26                public void onItemClick(ItemHome item, int position) {
27                    switch (position) {
28                        case 0: ... // Se lanzan los activities y fragments correspondientes
29                        default: // Por defecto se considerara un enlace web externo
30                            String url = item.getLink(); // Se obtiene el link
31                            Intent i = new Intent(Intent.ACTION_VIEW);
32                            // Se le dice a Android la acción y se le pasa la url
33                            i.setData(Uri.parse("http://" + url));
34                            startActivity(i); // Abre la url en el navegador externo
35                            break; // del dispositivo
36                    }
37                }
38            });
39        recyclerView.setLayoutManager(homeLayoutManager); // Se asigna el layout a recyclerView
40        recyclerView.setAdapter(homeAdapter); // y el adaptador
41    }
42}

```

Listado 4.27: Fichero `HomeFragment.java`, activity de la ventana de *Inicio*.

La clase “HomeFragment” (véase Listado 4.27) se encargará de crear una instancia de la clase “ItemHome” y el adaptador, “ItemHomeAdapter”, para que se sitúen correctamente en la vista que contiene el RecyclerView (véase Listado 4.28).

```
1 <LinearLayout ...>
2     <!-- RecyclerView, contenedor en el que irán las vistas de cada ítem de la clase ItemHome -->
3     <android.support.v7.widget.RecyclerView android:id="@+id/recyclerView_Home" ... />
4 </LinearLayout>
```

Listado 4.28: Fichero `fragment_home.xml`, vista principal de la ventana de *Inicio*.

4.4.3. AboutFragment

El fragmento “AboutFragment” se encarga de mostrar la ventana de *Información* la aplicación con la información general de la aplicación como su autor, versión, correo de contacto y una descripción de la misma. Este consta simplemente de un fichero `Fragment.java` que muestra el layout que contiene esta información en texto plano.

4.5. Menú

El menú de la aplicación constituye el elemento principal por el que el usuario navegará por la misma. Por ello se ha optado por incorporar un menú lateral deslizante llamado *Navigation Drawer*. Este es un menú que ha sido implementado por Google, se encuentra en sus principales aplicaciones como “Gmail” y “Google Play”. Un menú Navigation Drawer es un layout que incorpora dentro del mismo la vista de otras ventanas que se encuentren formato de fragmentos y facilita que el cambio entre los fragmentos sea fluido y rápido. Este menú (véase Figura 4.1b) se desplegará pulsando en la esquina superior izquierda de la aplicación.

El fichero `navigation_draw.xml` contiene la vista del Navigation Draw. Aquí se cuenta con un layout que muestra: la barra superior de la aplicación en la que irá el nombre de la aplicación “ULL-AR”, un “FrameLayout” en el que irá el fragmento actual a mostrar y un “NavigationView” que será la vista del menú Navigation Drawer y que al inicio de la aplicación estará sin desplegar.

En cuanto al contenido del menú del Navigation Drawer, se encuentra una cabecera en la parte superior del menú con una imagen de la ULL y el nombre del usuario. Después de la cabecera se encuentran las opciones del menú que vienen dadas por el fichero `nav_options.xml` (véase Listado 4.30). Cada uno de los ítems del menú de opciones tiene: un identificador, un ícono asociado y un nombre.

```

1 <android.support.v4.widget.DrawerLayout
2     android:id="@+id/drawer_layout" ... >
3     <!-- Contenido principal de la vista-->
4     <LinearLayout ... >
5         <include android:id="@+id/toolbar" ... />
6         <!-- Fragmento con el contenido principal de la vista -->
7         <FrameLayout android:id="@+id/content_frame" ... />
8     </LinearLayout>
9     <!-- Vista del menu lateral, con la cabecera y las opciones -->
10    <android.support.design.widget.NavigationView android:id="@+id/navigation_view"
11        app:headerLayout="@layout/header_navigation_drawer"
12        app:menu="@menu/nav_options" ... />
13 </android.support.v4.widget.DrawerLayout>

```

Listado 4.29: Fichero `navigation_drawer.xml`, layout del Navigation Drawer.

```

1 <menu xmlns:android="http://schemas.android.com/apk/res/android">
2     <!-- Opciones del menu del Navigation Draw -->
3     <group android:checkableBehavior="single">
4         <item android:id="@+id/menu_home"
5             android:icon="@drawable/ic_home"
6             android:title="Inicio" />
7         <item android:id="@+id/menu_maps" ... />
8         <item android:id="@+id/menu_ar" ... />
9         <item android:id="@+id/menu_sitios" .../>
10        <item android:id="@+id/menu_settings" ... />
11        <item android:id="@+id/menu_logout" ... />
12        <item android:id="@+id/menu_about" ... />
13     </group>
14 </menu>

```

Listado 4.30: Fichero `nav_options.xml`, opciones del menú Navigation Drawer.

Para la implementación del menú se ha utilizado una clase heredable llamada “ `BaseActivity`”, la cual incorpora la vista del Navigation Drawer, gestionará las opciones del menú, cambiará los fragmentos y permitirá moverse a los activities principales de ULL-AR. Con esta clase cualquier activity que quiera incorporar el menú Navigation Drawer solo tiene que heredar de ésta mediante la palabra reservada en Java “`extends`” y el nombre de la clase “ `BaseActivity`”.

4.6. Instalaciones de la ULL

En la base de datos se encuentran con la gran mayoría de las instalaciones de la ULL. Se recuerda que todas estas instalaciones constan de su nombre, ubicación, una breve descripción de esta y una lista de enlaces a los servicios, titulaciones, departamentos, etc. Toda esta información puede ser consultada por el usuario, mediante la ventana de *Todas las instalaciones* (véase Figura 4.3a). Esta ventana permite al usuario buscar la facultad, edificio o centro, que se encuentre en la base de datos, para posteriormente permitirle acceder a la ficha de información de la instalación (véase Figura 4.3b).

```

1   DrawerLayout drawerLayout; // Atributo del Navigation Drawer
2   ActionBarDrawerToggle actionBarDrawerToggle; // Boton que desplegará el menu
3
4   protected void onCreate(Bundle savedInstanceState) {
5       super.onCreate(savedInstanceState);
6       // Se le dice al activity la vista principal, este es el Navigation Drawer.
7       // Navigation Drawer muestra la vista de fragmentos y el menu desplegable Navigation View
8       setContentView(R.layout.navigation_draw);
9
10      // Se enlaza el Navigation Drawer de la vista
11      drawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
12      // Se incorpora la barra superior de la aplicación
13      toolbar = (Toolbar) findViewById(R.id.toolbar);
14      setSupportActionBar(toolbar);
15      // Se incorpora el botón de menú a la barra superior izquierda.
16      actionBarDrawerToggle = new ActionBarDrawerToggle(this, drawerLayout, toolbar, R.string.
17          app_name, R.string.app_name);
18      // Se indica al drawerLayout que escuche al botón para desplegar el menú
19      drawerLayout.setDrawerListener(actionBarDrawerToggle);
20
21      // Vista del menú del Navigation Drawer
22      NavigationView navigationView = (NavigationView) findViewById(R.id.navigation_view);
23      // Se enlazan los elementos del menú con su respuesta
24      navigationView.setNavigationItemSelectedListener(new NavigationView.
25          OnNavigationItemSelectedListener() {
26          // Cuando un elemento del menú sea seleccionado
27          public boolean onNavigationItemSelected(MenuItem item) {
28              FragmentTransaction tx;
29              // Para todas las opciones del menú
30              switch (item.getItemId()) {
31                  // Transición al fragmento HomeFragment
32                  case R.id.menu_home: // Opción "Inicio"
33                      tx = getSupportFragmentManager().beginTransaction();
34                      // Se cambia el fragmento
35                      tx.replace(R.id.content_frame, new HomeFragment());
36                      drawerLayout.closeDrawers(); // Se cierra el menú
37                      break;
38                  case R.id.menu_logout: // Opción "Cerrar sesión" del menú
39                      logout(); // Mediante la API de Google se sale de la cuenta
40                      goToLogin(); // Se regresa a la ventana de "Inicio de Sesión"
41                      break;
42                  ... // Resto de opciones del menú
43              }
44          ...
45      }

```

Listado 4.31: Fichero `BaseActivity.java`, código que se encarga de configurar la vista del Navigation Drawer.

En la ventana de *Todas las instalaciones* aparecerá una lista de ítems con todas las instalaciones de la ULL con: su imagen a la izquierda y, su nombre y descripción a la derecha. Para la creación de los ítems de la lista se optó por el uso de adaptadores al igual que en el fragmento “HomeFragment”, pero sin la necesidad de una vista RecyclerView. La clase encargada de representar las instalaciones es “SiteAdapter”, la cual hereda de la clase “BaseAdapter” de Android y, además,

implementa la interfaz “Filterable” que permite aplicar, de forma dinámica, un filtro de búsqueda sobre la lista de las instalaciones (véase Listado 4.32).

```

1  public class SiteAdapter extends BaseAdapter implements Filterable {
2      private ArrayList<ULLSiteSerializable> allSites; // Todas las instalaciones
3      // Instalaciones a mostrar con el filtro de búsqueda aplicado
4      private ArrayList<ULLSiteSerializable> filteredSites;
5      public SiteAdapter(Context context, int layout, SitesArray sitesULL) { // Constructor
6          ... // layout y contexto
7          allSites = sitesULL.getULLSiteSerializables(); // Se guarda el array con todas las
8              instalaciones
9          filteredSites = allSites; // Instalaciones a mostrar en un inicio
10     }
11     // Se crea la vista de cada ítem
12     public View getView(int position, View convertView, ViewGroup parent) {
13         View v = convertView; // vista
14         LayoutInflator layoutInflater = LayoutInflator.from(context);
15         // Se infla la vista con el layout de la instalación "site_item.xml"
16         v = layoutInflater.inflate(R.layout.site_item, null);
17         ... // Se enlazan en la vista el nombre, imagen y descripción de la instalación
18         // La librería Glide permite cargar imágenes de enlaces de la web
19         RequestManager requestManager = Glide.with(v.getContext());
20         RequestBuilder requestBuilder = requestManager.load(filteredSites.get(position).
21             getImageLink()); // Se obtiene la url de la imagen de la instalación
22         requestBuilder.into(imageSite); // Se carga la imagen en la vista
23         return v; // Se devuelve la vista
24     }
25     public Filter getFilter() {
26         return new Filter() { // Se instancia un objeto "Filter"
27             FilterResults performFiltering(CharSequence charSequence) { // Aplicamos el filtro
28                 String charString = charSequence.toString(); // String con el filtro
29                 if (charString.isEmpty()) { // Si está vacío
30                     filteredSites = allSites; // No hay filtro y se muestran todas las instalaciones
31                 } else { // Si no
32                     ArrayList<ULLSiteSerializable> auxFilteredList = new ArrayList<>();
33                     for (ULLSiteSerializable site : allSites) { // Para todas las instalaciones
34                         // Se comprueba si el nombre del filtro coincide con la instalación
35                         if (site.getName().toLowerCase().contains(charString.toLowerCase()))
36                             auxFilteredList.add(site); // Si coincide se agregan a la lista
37                             // auxiliar
38                     }
39                     filteredSites = auxFilteredList; // La lista auxiliar es igual a la de
40                         // las instalaciones filtradas a mostrar
41                     FilterResults filterResults = new FilterResults();
42                     filterResults.values = filteredSites;
43                     return filterResults; // Se devuelve el resultado del filtro
44                 }
45             @Override // Método que se ejecuta cuando se aplica el filtro
46             protected void publishResults(CharSequence charSequence, FilterResults filterResults) {
47                 filteredSites = (ArrayList<ULLSite...>) filterResults.values;
48                 notifyDataSetChanged(); // Se le dice al adaptador que el array con las instalaciones
49                     // ha sido modificado
50             };
51         };
52     }
53 }

```

Listado 4.32: Fichero `SiteAdapter.java`, adaptador que configura la lista de instalaciones con su imagen, nombre y descripción.

El contenido a representar en cada vista vendrá dado por una lista de objetos

de la clase “ULLSiteSerializable”. Esta clase es una copia de la clase “ULLSite” (véase Listado 4.12), pero que implementa la interfaz “Serializable” para permitir el paso de objetos de esta clase entre distintos activities.

El activity “SiteListActivity” (véase Listado 4.33) se encargará de crear una instancia del adaptador “SiteAdapter” con la lista de objetos de las instalaciones, las cuales recibe del activity anterior que lo ejecutó. Esté activity dispone de una barra de búsqueda la parte superior de la aplicación que permite al usuario buscar una instalación por su nombre. Esta indicará, al adaptador, el filtro de búsqueda que tiene aplicar a los nombres de las instalaciones cada vez que se escriba en ella o cuando el usuario le dé al botón de buscar, de esta manera solo aparecerán una lista de instalaciones que coinciden con los caracteres de la barra de búsqueda. Cuando una instalación de la lista sea seleccionada, se ejecutará el activity “SiteDescriptionActivity” y se le pasará el objeto que contiene la instalación seleccionada.

```

1 public class SiteListActivity extends AppCompatActivity { ...
2     private ListView listSites; // Vista que contendrá el adaptador con la lista de instalaciones
3     private SearchView searchView; // Barra superior de búsqueda
4     SiteAdapter siteAdapter; // Adaptador de las instalaciones
5     protected void onCreate(Bundle savedInstanceState) {
6         setContentView(R.layout.activity_sites_list); // Layout principal
7         // Se obtiene lista de las instalaciones enviadas por el activity anterior
8         sitesToShow = (SitesArray) getIntent().getSerializableExtra("sitesToShow");
9         showDinamicSites(); // Se muestran las instalaciones contenidas en la lista
10    }
11    private void showDinamicSites() { // Método que instancia la clase "SiteAdapter"
12        listSites = findViewById(R.id.listSites); // ListView que contendrá las instalaciones
13        siteAdapter = new SiteAdapter(this, R.layout.site_item, sitesToShow); // Adaptador
14        listSites.setAdapter(siteAdapter); // Se indica al ListView su adaptador
15        // Cuando se seleccione una instalación de la lista
16        listSites.setOnItemClickListener(new AdapterView.OnItemClickListener() {
17            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
18                Intent intent = new Intent(getApplicationContext(), SiteDescriptionActivity.class);
19                // Se instancia el activity "SiteDescriptionActivity" que muestra la información
20                // detallada de la instalación que se le pasa como extra
21                intent.putExtra("actualULLSite", siteAdapter.getFilteredSites().get(position));
22                startActivity(intent); } // Se lanza el activity
23            });
24        }
25        public boolean onCreateOptionsMenu(Menu menu) { // Barra de búsqueda
26            getMenuInflater().inflate(R.menu.search_bar,menu); // Layout con la barra de búsqueda
27            MenuItem searchItem = menu.findItem(R.id.app_bar_search); // Barra de búsqueda
28            searchView = (SearchView) MenuItemCompat.getActionView(searchItem); // Texto con el filtro
29            searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
30                @Override // Cuando se presione el botón de búsqueda
31                public boolean onQueryTextSubmit(String query) {
32                    siteAdapter.getFilter().filter(query); // Se indica al adaptador que aplique
33                    return false; } // el filtro
34                public boolean onQueryTextChange(String newText) {...} // Cuando el texto cambia
35            });
36            return super.onCreateOptionsMenu(menu); // Se devuelve la barra de búsqueda
37        }
    }
}

```

Listado 4.33: Fichero SiteListActivity.java.

El activity “SiteDescriptionActivity” muestra toda la información de la instalación contenida en el objeto “ULLSiteSerializable” proveniente del activity que ejecutó o lanzó a este. Este activity dispone de una ventana con: la imagen de la instalación junto con un botón en la parte inferior que abrirá la ruta de la instalación en la aplicación de Google Maps de dispositivo móvil, el nombre de la instalación, su descripción y una serie de enlaces URL relacionados con esta. Cuando uno de los enlaces es seleccionado, se abrirá el navegador externo del dispositivo móvil con la dirección web del enlace.

```

1 public class SiteDescriptionActivity extends ListActivity { ...
2     ULLSiteSerializable actualULLSite; // Instalacion a mostrar con su informacion
3     ArrayList<String> listItems = new ArrayList<String>(); // Lista de enlaces la instalacion
4     ArrayAdapter<String> adapter; // Adaptador con los enlaces
5     @Override // Metodo que inicia el activity
6     public void onCreate(Bundle savedInstanceState) { ...
7         setContentView(R.layout.activity_site_description); // Layout con la vista
8         // Se obtiene el objeto con la instalacion a mostrar
9         actualULLSite = (ULLSiteSerializable) getIntent().getSerializableExtra("actualULLSite");
10        listItems = actualULLSite.getInterestPoints(); // Se obtiene la lista de enlaces
11        setUI(); // Se muestra la informacion de las instalacion en la vista
12        setListSites(); // Se crea un adaptador con los enlaces las instalaciones
13    }
14    // Metodo que carga los textos, imagenes y enlaces de la instalacion a mostrar en la vista
15    public void setUI() {
16        ... // Se introduce la informacion del objeto de la instalacion en el layout
17        imageMaps.setOnClickListenner(new View.OnClickListener() {
18            @Override // Comportamiento de boton que abre la ubicacion de Google Maps
19            public void onClick(View v) { // Se le pasa a la url de maps + las coordenadas
20                Uri gmmIntentUri = Uri.parse("http://maps.google.com/maps?daddr="+ actualULLSite.
21                    getPoint().getY() + "," + actualULLSite.getPoint().getX());
22                Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);
23                mapIntent.setPackage("com.google.android.apps.maps"); // Paquete de Google Maps
24                startActivity(mapIntent); // Se lanza el intent que abre la ubicacion
25            }
26        });
27    }
28    // Este metodo crea una adaptador con los enlaces de la instalacion
29    public void setListSites() {
30        adapter = new ArrayAdapter<String>(this, R.layout.link_item, android.R.id.text, listItems)
31        {
32            @Override // Se configura la vista de cada enlace
33            public View getView(int position, View convertView, ViewGroup parent) {
34                ... // Se indica que la vista esta contenida en el fichero "link_item.xml"
35                } // Se introduce el nombre del enlace
36            };
37            setListAdapter(adapter); // Se indica al ListView por defecto de Android su adaptador
38            getListView().setOnItemClickListener(new AdapterView.OnItemClickListener() {
39                public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
40                    ... // Se lanza la url en el navegador externo
41                }
42            });
43            justifyListViewHeightBasedOnChildren(getListView()); // Se ajustan las dimensiones de
44                                            // la ventana
45        }
46        // Metodo que recalcula las dimensiones del layout para poder hacer scroll horizontal
47        public static void justifyListViewHeightBasedOnChildren (ListView listView) { ... }
48    }

```

Listado 4.34: Fichero SiteDescriptionActivity.java.

4.7. Preferencias del usuario

Se necesita un lugar donde el usuario pueda editar los ajustes de la aplicación. Para ellos se ha utilizado la clase heredable “PrefenceActivity” que ofrece Android para diseñar una ventana en la que poder editar de forma rápida e intuitiva los ajustes de una aplicación Android. Desde Android Studio se crea un “PrefenceActivity” con el nombre de “SettingsULLActivity”. A continuación, se generará un activity que hereda de “PrefenceActivity” en el cual insertar la vista que contiene los ajustes de la aplicación. El fichero `pref_nav_setting.xml` (véase Listado 4.35) contendrá la lista con los ajustes necesarios para la aplicación. Cada uno de estos ajustes pueden disponer, en función del tipo ajuste que se requiera, de un nombre, una descripción, una clave y un valor por defecto. Estos ajustes permitirán decidir al usuario si desea encontrar las instalaciones, en el mapa o en el modo realidad aumentada, dentro de dos circunferencias y la dimensiones en metros de ambos radios (véase Figura 4.4a).

```
1 <PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
2     <PreferenceCategory android:title="Preferencias de las instalaciones" ...>
3         <!-- Activar o no los radios de busqueda -->
4         <SwitchPreference android:key="showRadius"
5             android:title="Mostrar instalaciones dentro de un rango" .../>
6         <!-- Radio del circulo mayor -->
7         <EditTextPreference
8             android:defaultValue="max"
9             android:inputType="number"
10            android:key="maxRadius"
11            android:title="Selecciona radio circulo mayor (metros)" .../>
12         <!-- Radio del circulo menor -->
13         <EditTextPreference android:key="minRadius" ...>
14     </PreferenceScreen>
```

Listado 4.35: Fichero `pref_nav_setting.xml`.

Esta recurso de ventanas que ofrece Android para crear los ajustes de una aplicación, permite trabajar con las Shared Preferences para guardar los valores de los ajustes y poder acceder a ellos en otras ventanas de la aplicación. Cuando se edita un valor en la ventana de la aplicación, este valor se guardará automáticamente en las Shared Preferences con el valor de la clave “key”. Esta clave servirá para poder acceder al valor a través de las Shared Preferences en cualquier activity.

4.8. Back-end de la aplicación

Si se recuerda, en el capítulo 2 se comentó sobre el uso de las tecnologías de Node.js y MongoDB, como servidor y base de datos respectivamente. Para poder implementar estas tecnologías se decidió por utilizar las plataformas PaaS en la nube para gestionarlos, facilitar su despliegue y con motivos de profundizar y adquirir conocimiento sobre las capacidades y ventajas de este tipo de plataformas.

Estos plataformas son Heroku como servidor de la aplicación y mLab como base de datos. Ambos ofrecen una forma sencilla, accesible y escalable, para poder realizar las pruebas con la aplicación por las instalaciones de la ULL sin tener que montar un servidor físico.

4.8.1. Base de datos

Todos los datos necesarios para el funcionamiento de la aplicación estarán alojados en una base de datos en la nube. mLab será el proveedor de servicio escogido para alojar la base de datos. Este proveedor ofrece bases de datos NoSQL que utilizan la tecnología MongoDB.

Crear la base de datos es muy sencillo con mLab. Simplemente hay que registrarse y crear una base de datos. Lo primero que se necesita es elegir un plan de datos. Como en la base de datos que se necesita no es necesario mucho espacio de almacenamiento, se optó por el plan gratuito que dispone de 500 megas. Posteriormente se fija el nombre de la base de datos que será “bd-ull-AR”.

4.8.2. Configuración del servidor

Se necesita configurar el servidor de Node.js para poder desplegarlo en Heroku y que funcione correctamente. A continuación, se explicarán los pasos seguidos para la implementación del servidor explicando su funcionamiento y su conexión con la base de datos.

Requisitos previos

Para toda la instalación e implementación del servidor Node.js se ha utilizado Linux como sistema operativo.

Previo a la implementación se necesitará crear una cuenta en la plataforma de Heroku y en la de mLab. A su vez, se debe tener instalado GitHub, Node.js y Heroku. Para poder ejecutar los comandos que permitan creación y despliegue del servidor en la nube.

Dentro de la cuenta de Heroku, se crea un repositorio con el nombre de ”server-ull-AR” que será el servidor de la aplicación en la nube.

Instalación de Node.js en el repositorio

Terminados de instalar todos los requisitos necesarios, ya se puede empezar a implementar el servidor de Node.js.

Con el siguiente comando, ya se tiene preparado el repositorio para empezar a trabajar:

```
1 $ git clone https://github.com/heroku/server-ull-AR.git
```

Para empezar a trabajar con Node.js se necesita ejecutar el siguiente comando dentro del repositorio:

```
1 $ npm init
```

Este comando resulta en la creación de un archivo llamado `package.json`. Este fichero se utiliza para administrar los paquetes disponibles en el “Node Package Manager” que se instalan localmente en el repositorio.

A continuación, se instalarán los paquetes necesarios para funcionamiento del servidor Node.js. Estos son:

- **ExpressJS:** Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles.
- **mongoose:** Mongoose es una librería para trabajar MongoDB y Node.js.
- **bodyparser:** Se necesitará para manejar las peticiones de JSON.
- **node-restful:** Sirve para manejar las peticiones recibidas del servidor y conectarse con una base de datos de MongoDB.

Con un comando se instalarán todos los paquetes y se guardarán las dependencias utilizadas en el fichero `package.json`:

```
1 $ npm install --save express body-parser mongoose node-restful
```

El resultado del `package.json` sería el siguiente:

```
1 {
2   "private": true,
3   "name": "server-ull-AR",
4   "version": "0.0.1",
5   "dependencies": {
6     "body-parser": "^1.18.3",
7     "express": "^4.16.3",
8     "mongoose": "^5.4.16",
9     "node-restful": "^0.2.6"
10   },
11   "engines": {
12     "node": ">=0.10.25"
13   }
14 }
```

Listado 4.36: Contenido del fichero `package.json`.

Inicialización del servidor

Una vez ya instalados todos los paquetes se crea el fichero `server.js` (véase Listado 4.37)x . Este fichero será el que inicie el servidor y contenga las variables que configuran el mismo.

```

1 // Se declaran los paquetes que se necesitan
2 var express = require('express');
3 var app = express(); // Se inicia la app
4 // Se pone en modo escucha la app
5 app.listen(3000, function() {
6     console.log("Express server listening on port 3000"
7 });

```

Listado 4.37: Fichero `server.js`, configuración inicial del servidor.

Para comprobar que el servidor funciona correctamente, se utilizará el siguiente comando, que ejecutará el servidor en local en el puerto 3000.

```
1 $ node server.js
```

Más adelante se terminará de configurar este fichero. A continuación se explicará la estructura y organización del servidor.

Estructura de la aplicación

Dentro de la carpeta principal del repositorio se han de crear dos subdirectorios:

- `models/`
- `routes/`

```

1 // Se declara el variable restful para manejar las peticiones
2 var restful = require('node-restful');
3 // Se usa mongoose para conectarse a la BD
4 var mongoose = restful.mongoose;
5 // Estructura de las instalaciones de la ull contenidos en la base de datos
6 var ullSitesSchema = new mongoose.Schema({
7     id: String,
8     name: String,
9     position: {
10         lat: String,
11         long: String
12     },
13     desc: String,
14     imageLink: String,
15     canFind: [
16         {
17             id: String,
18             link: String
19         }
20     ]
21 });
22 // Se devuelve el modelo para poder utilizarlo en otros ficheros
23 // Este modelo se conectara con colección de la base de datos "ull_sites"
24 module.exports = restful.model('ull_sites', ullSitesSchema);

```

Listado 4.38: Fichero `ullSites.js`.

A continuación se creará el fichero `ullSites.js` (véase Listado 4.38) dentro de la carpeta de `models/`. Este fichero contiene el modelo que conecta con la colección de la base de datos y maneja las respuestas, como se explicará más adelante.

Se necesitará una ruta por la cual el servidor responderá con la información que se solicite de la base de datos. Para ello en la carpeta `routes/` se creará el fichero `api.js` (véase Listado 4.39). Este fichero manejará las peticiones que lleguen al servidor a través de la ruta “`https://server_url/api/`” y se encargará principalmente de conectarse con la base de datos para responder a estas peticiones.

En el caso de que se acceda a la ruta “`/api/ull-sites`” del servidor con una petición “HTTP” de tipo “GET”, se enviará una respuesta al cliente con toda la información de las instalaciones que se encuentran en la base de datos.

```

1 var express = require('express');
2 var router = new express.Router();
3 // Modelo que maneja la petición a la base de datos
4 var ullSites = require('../models/ullSites');
5 // Se selecciona los métodos que puede responder
6 // Al ser una aplicación sencilla solo se necesitan manejar peticiones get
7 ullSites.methods(['get']);
8 // Se indica al router la url que gestionara las peticiones
9 ullSites.register(router, '/ull-sites');
10 module.exports = router; //Exportamos este módulo

```

Listado 4.39: Fichero `api.js`.

Conexión con mLab

Con la cuenta de mLab y base datos ya creada en mLab, solo se va a necesitar una URL para poder acceder a ella. Esta URL está disponible en la página principal de la base de datos en mLab y tiene el siguiente formato:

```
1 mongodb://<dbuser>:<dbpassword>@ds235181.mlab.com:35181/ull-AR
```

Donde “dbuser” es el usuario que se usó para crear la base de datos y “dbpassword” la contraseña. Para evitar que el usuario y contraseña queden expuestos públicamente en el repositorio, se utiliza una variable de entorno. En este caso hay que configurar la variable de entorno para que funcione en Heroku. Para hacerlo se necesita utilizar este comando en la terminal:

```
1 $heroku config:set PROD_MONGODB=mongodb://username:password@ds235181.mlab.com:35181/ull-AR
```

Posteriormente la variable “PROD_MONGODB” se utilizará para conectarse al base de datos una vez este desplegado en Heroku.

Solo se va a necesitar crear una colección para el proyecto. Para ello se accede a la base de datos en el navegador y se pincha en el botón de “Add collection” y se nombrará “`ull_sites`”. Aquí se tiene la información de las instalaciones de la ULL, las cuales se añadirán manualmente desde la página de mLab.

Configuración final del servidor

Por último, se tiene que terminar de configurar el fichero `server.js` (véase Listado 4.40) que contiene el servidor. Se necesitará indicar al servidor la URL de la base de datos de mLab y de disponer que las rutas de la aplicación estén bien configuradas para responder a las solicitudes de ULL-AR.

```

1 var express = require('express'); // Servidor
2 var mongoose = require('mongoose');// Para conectar con BD
3 var bodyParser = require('body-parser');// Manejar peticiones JSON
4
5 var server = express(); // Se instancia el servidor
6 server.use(bodyParser.urlencoded({extended: false}));
7 server.use(bodyParser.json());
8 // Se usa el fichero que se encuentra en ./routes/api.js
9 api = require('./routes/api');
10 // Se realiza la conexión con la base de datos con la url correctamente
11 // guardada en PROD_MONGODB
12 mongoose.connect(process.env.PROD_MONGODB, function (error) {
13     if (error) console.error(error);
14     else console.log('mongo connected');
15 });
16
17 server.get('/', function (req, res) { // Ruta raíz del servidor
18     res.send('ULL-AR server');// Respuesta por defecto
19 })
20 // Se le dice al servidor que el archivo /routes/api.js se encargue de las
21 // solicitudes que se reciben de /api
22 server.use('/api', api);
23 // Se pone el servidor a escuchar en el puerto de producción o en el 3000
24 server.listen(process.env.PORT || 3000, function() {
25     console.log("Express server listening on port %d in %s mode", this.address().port, server.
26         settings.env);
27 });

```

Listado 4.40: Configuración final del fichero `server.js`.

4.8.3. Despliegue en Heroku

Terminada la implementación del servidor, se puede realizar el despliegue en Heroku. Para ello se necesitará crear un fichero `Procfile` (véase Listado 4.41) en la raíz del repositorio, que le dirá a Heroku cuál es el fichero que inicia el servidor.

```
1 web: node server.js
```

Listado 4.41: Fichero `Procfile`.

Para desplegarlo en Heroku se necesitarán estos tres comandos:

```

1 $ git add .
2 $ git commit -m "Despliegue del servidor"
3 $ git push heroku master

```

Con estos pasos completados ya se tiene el servidor desplegado y funcionando en la URL [49].

Capítulo 5

Conclusiones y futuras líneas de Trabajo

En este capítulo se presentarán las conclusiones a las que se ha llegado tras realizar este TFG y se discutirán posibles líneas de trabajo futuras.

5.1. Conclusiones

Actualmente la realidad aumentada está invadiendo nuestras vidas, cada día se descubren nuevas muestras de lo que esta tecnología es capaz. La gran mayoría de sus futuras aplicaciones se encuentran aún en desarrollo debido a lo reciente y compleja que es esta tecnología. La RA se encontrará integrada en múltiples ámbitos tanto como profesionales dados sus aplicaciones en medicina e ingeniería, como en el de la enseñanza permitiendo a los estudiantes mejorar e incentivar el proceso de aprendizaje.

En cuanto a la implementación y desarrollo de aplicaciones de RA, se encuentran limitadas al avance en el desarrollo de los SDK de RA que ofrecen empresas como Google, Apple o Vuforia, que cada día van ofreciendo más funcionalidades y mejor rendimiento pero que todavía tienen un amplio margen de mejora en función de lo que se espera de esta tecnología. Uno de los principales inconvenientes que se encuentra a la hora de desarrollar una aplicación de RA es elegir un SDK de RA que se adapte tanto a los requisitos como al presupuesto, ya que aunque la mayoría de los proveedores de estos SDK tienen versiones gratis para desarrolladores, a la hora de querer integrar esta tecnología en una aplicación, la mayor parte de sus funcionalidades más interesantes para el reconocimiento de objetos en la nube, se encuentran con unos precios mensuales fuera del alcance de muchas empresas y desarrolladores.

En cuanto de la integración de un SDK de RA en Android Studio, se han

encontrado con diversos problemas que dificultan trabajar con la RA en este IDE debido a que la mayoría de las empresas de RA que se encargan de desarrollar una versión de sus SDK para Android Studio, dejan de lado muchas funcionalidades y no ofrecen la documentación suficiente para instalar y empezar a trabajar con el SDK. En caso de querer realizar una aplicación de realidad aumentada de cierta calidad se recomienda el uso de plataformas de desarrollo 3D como Unity.

5.2. Conclusions

Nowadays, augmented reality is invading our lives, every day new samples of what this technology is capable of are being discovered. The great majority of its future applications are still under development due to the recent and complex nature of this technology. AR will be integrated in multiple fields, both as professionals given its applications in medicine and engineering, as well as in teaching, allowing students to improve and encourage the learning process.

As for the implementation and development of AR applications, they are limited to the advance in the development of AR SDKs offered by companies such as Google, Apple or Vuforia, which every day are offering more functionality and better performance but still have a wide margin for improvement depending on what is expected of this technology. One of the main drawbacks when developing an AR application is to choose an AR SDK that fits both requirements and budget, since although most of the providers of these SDK have free versions for developers, when trying to integrate this technology into an application, most of its most interesting features for the recognition of objects in the cloud, are with monthly prices beyond the reach of many companies and developers.

As for the integration of an AR SDK in Android Studio, they have encountered several problems that make it difficult to work with AR in this IDE because most of the AR companies that are responsible for developing a version of their SDK for Android Studio, leave aside many features and do not offer enough documentation to install and start working with the SDK. If you want to make an augmented reality application of a certain quality is recommended the use of 3D development platforms such as Unity.

Capítulo 6

Presupuesto

En este capítulo se expondrán las estimaciones de recursos necesarios para desarrollar y publicar el proyecto.

El desglose del presupuesto de la aplicación se puede contabilizar de la siguiente manera:

- En caso de que se desee publicar la aplicación en la Google Play Store, se necesita poseer de una cuenta de desarrollador de Google. Para ello se necesitará disponer de una cuenta de Google a una de desarrollador, se han de seguir los pasos en la web [50], en la que pedirán un abono único de 25 dólares.
- En cuanto a coste del servidor, Heroku ofrece distintos planes para la gestión de este. En Heroku los “dynos” son las unidades que proveen capacidad de cómputo. El pago mensual en Heroku dependerá de la cantidad y la potencia de los dynos que se tengan contratados. Las cuotas por dyno que ofrece Heroku son:
 - Free: 0 dólares al mes.
 - Hobby: 7 dólares por dyno al mes.
 - Standard 1x: 25 dólares por dyno al mes.
 - Standard 2x: 50 dólares por dyno al mes.
 - Perfomance M: 250 dólares por dyno al mes.
 - Perfomance L: 500 dólares por dyno al mes.

Los planes Free y Hobby no constan de mucha potencia y tienen limitados el número de procesos que pueden responder a las peticiones, son recomendadas para uso personal y de pequeñas aplicaciones. El resto de los planes ofrecen

número de procesos ilimitados y permite la escalabilidad horizontal en el caso de que la aplicación lo requiera.

Para los requisitos de la aplicación se necesitaría contratar un dyno del plan Standard 1x. El coste mensual de este plan sería de 300 dólares al año. En caso de que el dyno contratado no sea capaz de soportar el tráfico de la aplicación se podría cambiar de plan o agregar otro dyno Standard 1x.

- La base de datos se encuentra situada en la plataforma de mLab. Esta plataforma en función del número de GB que se necesiten para la base de datos ofrece los siguientes planes:
 - Sandbox: 0.5 GB gratis.
 - Shared Cluster: 15 dólares al mes por GB.
 - Dedicated Cluster: a partir de 180 dólares al mes por 40 GB.

El plan de Sandbox se ajusta a los requisitos de la base de datos de la aplicación, por lo que no tendría coste.

- La información referente a cada instalación sería recomendable obtenerla de primera mano del personal encargado de cada instalación para asegurar que la información esté completa y correcta. Esto supondría un costo de 300 o 400 euros para gastos de transporte e investigación.
- El salario promedio de los programadores en España es de unos 1600 euros al mes, de modo que el coste por hora se puede estimar en unos 10 euros por hora de trabajo (teniendo en cuenta 8 horas de trabajo diarias, 40 semanales y 160 mensuales). Para poder desarrollar la aplicación sería necesario contratar a un programador experto Android. Teniendo en cuenta que el desarrollo de ULL-AR se estima en unos dos meses de trabajo en estas condiciones, ello totalizaría unos 3200 euros por el desarrollo de la aplicación.
- Sería muy conveniente el mantenimiento de la aplicación. Se ofrece un programador en Android con conocimientos en Node.js que se haría cargo de las incidencias que pudieran surgir, posibles mejoras puntuales de la aplicación, del mantenimiento del servidor y de actualizar la información de la base de datos. Este programador trabajaría con un coste por hora de 20 euros/hora, de modo que el coste del mantenimiento se calcularía por parte de la organización interesada en ULL-AR mediante la contratación de una “bolsa de horas” de trabajo de este programador.

El cálculo del coste total de la aplicación se hará en función de los puntos comentados anteriormente.

Bibliografía

- [1] “Sistema operativo de Android.” [<https://www.android.com/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 1, 6
- [2] “Repositorio Github.” [<https://github.com/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 1, 3
- [3] “Latex.” [<https://www.latex-project.org/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 1, 3
- [4] “Android Studio.” [<https://developer.android.com/studio/index.html>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 2
- [5] “IDE.” [https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado]. [Disponible electrónicamente. Último acceso, julio de 2019]. 2
- [6] “IntelliJ IDEA.” [<https://www.jetbrains.com/idea/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 2
- [7] “Gradle.” [<http://gradle.org/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 2
- [8] “Repositorio Github del proyecto ULL-AR.” [<https://github.com/alehdezp/TFG-ULL-AR>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 3, 4
- [9] “Realidad Aumentada.” [https://es.wikipedia.org/wiki/Realidad_aumentada]. [Disponible electrónicamente. Último acceso, julio de 2019]. 6
- [10] “Registro de Imagenes.” [https://es.wikipedia.org/wiki/Registro_de_la_imagen]. [Disponible electrónicamente. Último acceso, julio de 2019]. 7
- [11] “CodigoQR.” [https://es.wikipedia.org/wiki/Codigo_QR]. [Disponible electrónicamente. Último acceso, julio de 2019]. 8

- [12] “Realidad Virtual.” [https://es.wikipedia.org/wiki/Realidad_virtual]. [Disponible electrónicamente. Último acceso, julio de 2019]. 9
- [13] “Relidad Mixta.” [https://es.wikipedia.org/wiki/Realidad_mixta]. [Disponible electrónicamente. Último acceso, julio de 2019]. 9
- [14] “Pokémon Go!” [<https://pokemongolive.com/es/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 11
- [15] “Pokémon.” [<https://es.wikipedia.org/wiki/Pokemon>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 11
- [16] “Kit de desarrollo de software.” [https://es.wikipedia.org/wiki/Kit_de_desarrollo_de_software]. [Disponible electrónicamente. Último acceso, julio de 2019]. 12
- [17] “Página web de Vuforia.” [<https://developer.vuforia.com/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 12
- [18] “Unity.” [<https://unity.com/es>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 13
- [19] “Kudan.” [<https://www.kudan.io/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 13, 43
- [20] “Página web de Maxst.” [<https://developer.maxst.com/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 13
- [21] “SLAM.” [https://es.wikipedia.org/wiki/Localizaci\penalty\@M\hskip\z@skip\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{o\global\mathchardef\accent@spacefactor\spacefactor}\accent19o\egroup\spacefactor\accent@spacefactor\penalty\@M\hskip\z@skip\spacefactor\sfcode'on_y_modelado_simult\penalty\@M\hskip\z@skip\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{a\global\mathchardef\accent@spacefactor\spacefactor}\accent19a\egroup\spacefactor\accent@spacefactor\penalty\@M\hskip\z@skip\spacefactor\sfcode'aneos]. [Disponible electrónicamente. Último acceso, julio de 2019]. 13
- [22] “Node.js.” [<https://nodejs.org/en/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 14

- [23] “Chrome V8.” [https://es.wikipedia.org/wiki/Chrome_V8]. [Disponible electrónicamente. Último acceso, julio de 2019]. 14
- [24] “Lenguajes de programación del lado del servidor.” [https://es.wikipedia.org/wiki/Script_del_lado_del_servidor]. [Disponible electrónicamente. Último acceso, julio de 2019]. 14
- [25] “MongoDB.” [<https://www.mongodb.com/what-is-mongodb?lang=es-es>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 15
- [26] “Tecnología NoSQL.” [<https://es.wikipedia.org/wiki/NoSQL>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 15
- [27] “Formato BSON.” [<https://es.wikipedia.org/wiki/BSON?lang=es-es>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 16
- [28] “PaaS.” [https://en.wikipedia.org/wiki/Platform_as_a_service]. [Disponible electrónicamente. Último acceso, julio de 2019]. 16
- [29] “Instalación Heroku.” [<https://devcenter.heroku.com/articles/heroku-cli>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 17
- [30] “Amazon Web Services.” [<https://aws.amazon.com/es/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 17
- [31] “Azure.” [<https://azure.microsoft.com/es-es/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 17
- [32] “Google Cloud.” [<https://cloud.google.com/?hl=es>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 17
- [33] “Google Maps API.” [<https://developers.google.com/maps/documentation/android-api>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 17
- [34] “Aprendizaje ubicuo.” [https://es.wikipedia.org/wiki/Aprendizaje_ubicuo]. [Disponible electrónicamente. Último acceso, julio de 2019]. 20
- [35] “Animating eco-education: To see, feel, and discover in an augmented reality-based experiential learning environment..” [<https://www.sciencedirect.com/science/article/pii/S0360131516300288>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 20

- [36] “Augmented Reality Trends in Education : A Systematic Review of Research and Applications.” [<http://disde.minedu.gob.pe/handle/123456789/5029>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 20
- [37] “Gafas de Microsoft Hololens.” [<https://www.microsoft.com/es-es/hololens>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 22
- [38] “AR Sandbox.” [<https://arsandbox.ucdavis.edu/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 23
- [39] “Splash Screen.” [https://en.wikipedia.org/wiki/Splash_screen]. [Disponible electrónicamente. Último acceso, julio de 2019]. 25
- [40] “GSuite de Google.” [<https://gsuite.google.com/intl/es/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 25
- [41] “Navigation Drawer.” [<https://material.io/design/components/navigation-drawer.html>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 25
- [42] “Pixlr.” [<https://pixlr.com/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 30
- [43] “Activity Android.” [<https://developer.android.com/guide/components/activities.html?hl=ES>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 30
- [44] “Consola de Firebase.” [<https://console.firebaseio.google.com/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 31
- [45] “Layout Android.” [<https://developer.android.com/guide/topics/ui/declaring-layout?hl=es-419>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 32
- [46] “Fragmentos Android.” [<https://developer.android.com/guide/components/fragments?hl=es-419>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 45
- [47] “Consola de Google.” [<https://console.developers.google.com/>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 45
- [48] “RecyclerView.” [<https://developer.android.com/guide/topics/ui/layout/recyclerview>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 47

- [49] “Servidor de ULL-AR.” [<https://server-ull-AR.herokuapp.com>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 60
- [50] “Google Play Console.” [<https://developer.android.com/distribute/console?hl=es>]. [Disponible electrónicamente. Último acceso, julio de 2019]. 63