Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN

Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

APLICACIÓN PARA SEGUIMIENTO DE RUTAS UTILIZANDO REALIDAD AUMENTADA

Alumno: Ángel Hurtado Garrido

Tutor: Prof. D. Ángel Luis García Fernández

Dpto: Informática

AGRADECIMIENTOS

En primer lugar a mis padres, por todo el apoyo que me han dado siempre y por estar ahí en todo momento, confiando en mí y ayudándome a continuar. Por su infinita paciencia, por nunca permitir que me ponga un techo. Sin ellos esto no hubiera sido posible.

A mis amig@s, compañer@s de pisos y sobre todo a mis compañer@s de carrera. Todos ellos forman parte de este trabajo.

A cada una de las personas que se han preocupado y me han apoyado durante todos estos años y más especialmente durante la realización del trabajo.

A todos los profesores que han contribuido en mi enseñanza, en especial a Ángel Luis por tutorizar este trabajo.

Índice

1. Presentación

La finalidad de este primer capítulo es proporcionar al lector una visión general de este proyecto, explicando su objetivo, contexto, motivación y la forma en que se estructura el resto del documento.

1.1. Objetivo

El principal objetivo de este trabajo es el desarrollo de una aplicación móvil que permita la obtención de una ruta entre dos puntos dentro del campus de Las Lagunillas de la Universidad de Jaén, utilizando códigos QR para el posicionamiento del usuario y generando realidad aumentada en su dispositivo móvil para guiarlo hasta su destino.

En el transcurso de los años de estudio, surge la necesidad de acudir a numerosos lugares dentro del campus de la universidad (despachos, laboratorios, aulas, biblioteca...), y aparece el problema; no conocemos exactamente su ubicación. Éste se acentúa más en el alumnado de primer año o en alumnos de otros países que acuden a las instalaciones por primera vez. Se dan vueltas enormes leyendo carteles, recorriendo pasillos interminables por los que, normalmente, no sería necesario pasar. Esto se traduce en un gasto de tiempo innecesario que puede ser solucionado fácilmente.

Con este trabajo se quiere proponer una solución a este problema. Se pretende realizar una aplicación que nos permita obtener la ruta más eficiente entre dos puntos y con ella, guiar a nuestro usuario con indicaciones visuales y sencillas hacia su destino.

1.2. Introducción

Los dispositivos móviles y el Internet de las cosas (IoT), dos tecnologías intrínsecamente conectadas entre sí, han dado lugar a un mercado que promete miles de millones de dispositivos. Cada día es más el número de personas que poseen un smartphone (en España ya los usa un 81% de la población) y el número de aplicaciones móviles de toda índole que encontramos para nuestros dispositivos crece, desde pequeñas utilidades como un calendario, a grandes aplicaciones que incluyen las últimas tecnologías del mercado, entre las que podemos citar a la última joya de Niantic, Inc., Pokemon GO.

El número de apps descargadas, sólo por los usuarios españoles, es de 3.8 millones al día y de media solo usamos activamente 14 aplicaciones de las instaladas en nuestros teléfonos, con las que pasamos el 89% del tiempo que dedicamos al uso de nuestros smartphones. [REF1.2.1]

Todos estos datos arrojan una señal inequívoca de que nuestro smartphone empieza a ser una herramienta imprescindible en nuestras vidas. Es por ello que empiezan a surgir nuevas aplicaciones, cada vez con tecnologías más sofisticadas, que pretenden facilitar el día a día a sus usuarios.

Una de estas tecnologías que se encuentran a la orden del día es la Realidad Aumentada. Tal y como define la wikipedia:

"La Realidad Aumentada es el término que se usa para definir una visión a través de un dispositivo tecnológico, directa o indirecta, de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta en tiempo real." [REF1.2.2]

Esta tecnología abre un mundo de posibilidades a nuevas aplicaciones móviles y, aunque no es una tecnología novedosa en la actualidad, en los últimos años son más las compañías que apuestan por ella para sus nuevos lanzamientos. Un ejemplo de las posibilidades de esta tecnología y del impacto tanto social como económico de ésta, es el mencionado anteriormente Pokemon GO (Ilustración 1.2.1).

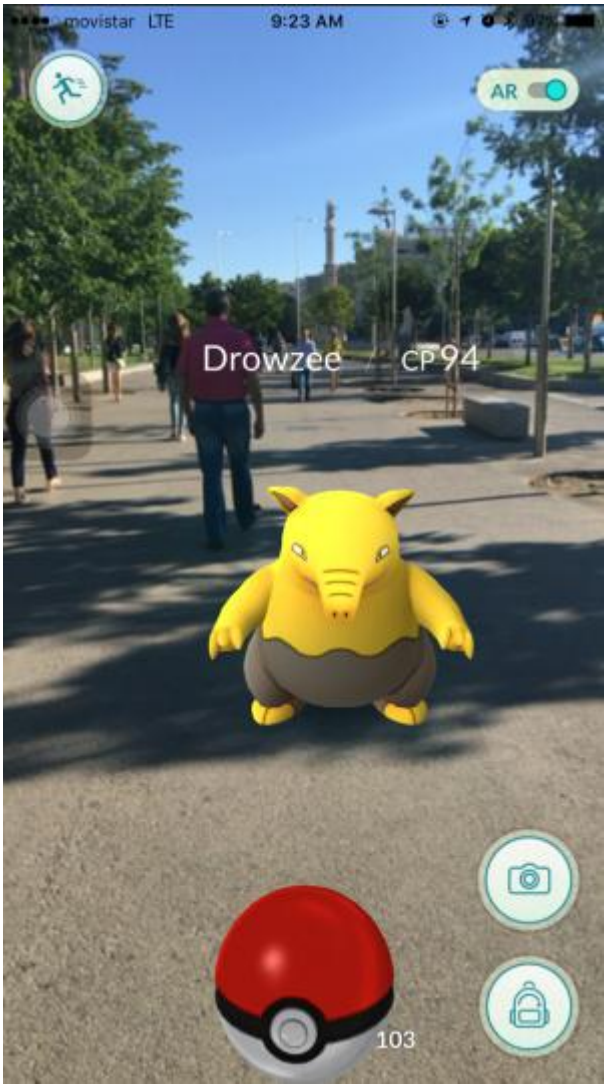


Ilustración 1.2.1. Captura de pantalla Pokemon GO.

[REF1.2.3]

Cuando hablamos de impacto social y económico no tenemos nada más que observar las gráficas Ilustración 1.2.2 e Ilustración 1.2.3 que permiten hacernos una idea de la dimensión de este lanzamiento.

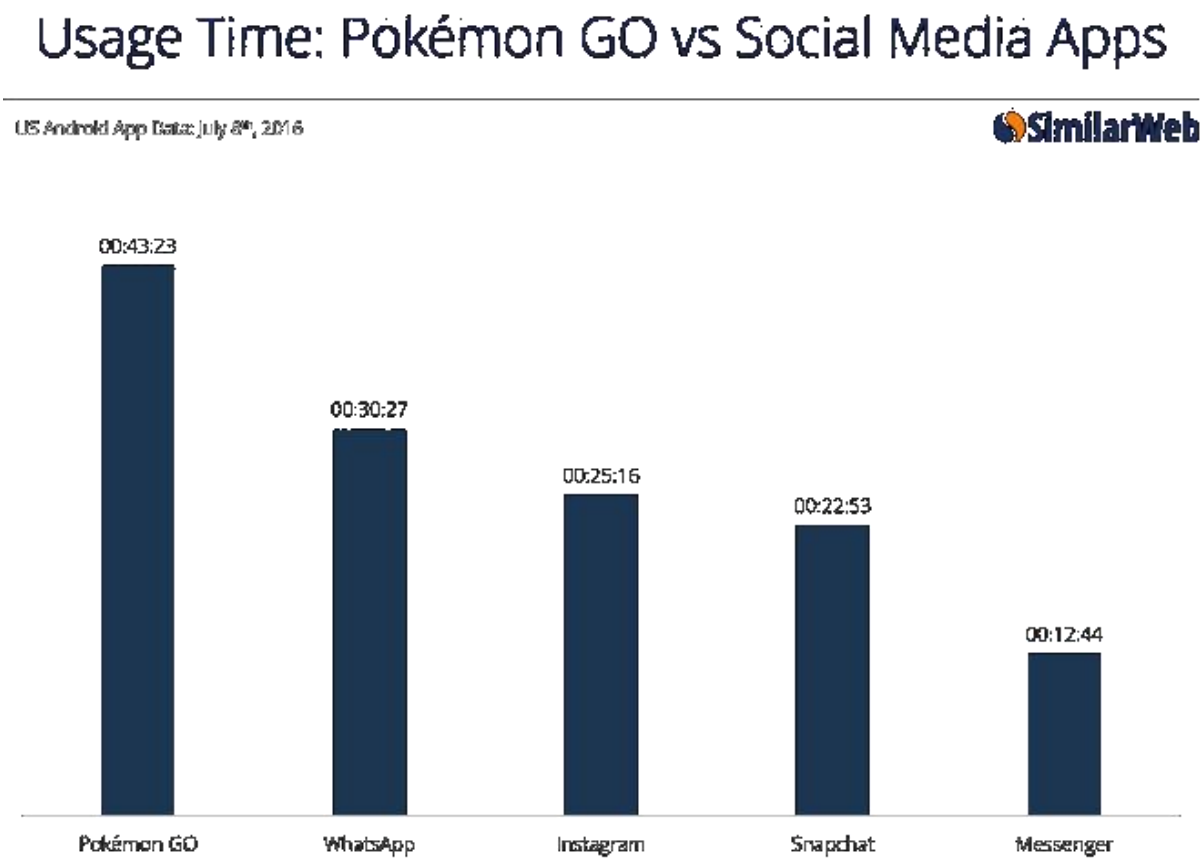


Ilustración 1.2.2. Tiempo de uso. Pokemon GO vs Social Media Apps.

[REF1.2.4]

Ilustración 1.2.3. Evolución empresa Nintendo en bolsa.

[REF1.2.5]

Estamos ante un universo de posibilidades que se encuentra ya disponible en numerosos ámbitos como en medicina, construcción o en viajes.

1.3. Estructura del contenido

Este trabajo se estructurará de acuerdo al ciclo a seguir durante el desarrollo de software.

En primer lugar se realizará un análisis de la aplicación a desarrollar, se enumerarán los requisitos de ésta, tanto funcionales como no funcionales. Se realizará un estudio de las distintas tecnologías que se verán involucradas durante el desarrollo del software y se elegirán las que más se adecuen al objetivo. Por último, en la fase de análisis, se presentará un presupuesto estimado del coste que supondrá el desarrollo de la aplicación.

En segundo lugar trataremos las distintas alternativas al diseño de la aplicación. Haremos un recorrido desde el modelado de datos, pasando por el diseño de la aplicación con sus respectivos diagramas, para finalizar con el diseño de la interfaz.

Posteriormente, se realizará un estudio de la implementación realizada. Se comentarán clases de interés, cambios en el diseño previsto, motivos y consecuencias de dichos cambios, así como integración entre módulos.

Por último tendremos la sección de pruebas, donde quedarán documentadas todas las pruebas realizadas al software durante y después de la implementación. También existirá un apartado donde quedarán reflejadas algunas métricas que han sido pasadas al código fuente al término de la implementación.

2. Análisis

En este apartado trataremos todos aquellos temas relacionados con el análisis previo que debemos realizar antes de empezar a trabajar en la aplicación.

2.1. Requisitos

Partiendo de la definición del objetivo del proyecto podemos obtener los requisitos básicos que debe cumplir la aplicación. A estos se añadirán algunos requisitos de rendimiento que permitan un uso óptimo de ésta.

2.1.1. Requisitos funcionales

Requisitos encargados de especificar todas las funcionalidades que el sistema debe satisfacer.

Nº	ATRIBUTO	CONTENIDO/EXPLICACIÓN
1	Identificador	RQ-F01
2	Nombre	Calcular ruta con menor distancia
3	Versión	1.0
4	Historia de cambios	
5	Prioridad	Alta
6	Descripción	La aplicación calculará la ruta más corta entre dos puntos: la ubicación del usuario y el destino seleccionado.

Tabla 2.1.1.1. Requisito funcional 1.

Nº	ATRIBUTO	CONTENIDO/EXPLICACIÓN
1	Identificador	RQ-F02
2	Nombre	Mostrar camino a seguir mediante realidad aumentada
3	Versión	1.0
4	Historia de	

	cambios	
5	Prioridad	Alta
6	Descripción	La aplicación reconocerá un patrón sobre el que se sobre-impresionará una flecha indicando la dirección a seguir.

Tabla 2.1.1.2. Requisito funcional 2.

Nº	ATRIBUTO	CONTENIDO/EXPLICACIÓN
1	Identificador	RQ-F03
2	Nombre	Escanear Código QR
3	Versión	1.0
4	Historia de cambios	
5	Prioridad	Alta
6	Descripción	La aplicación permitirá escanear un código QR que nos aportará la ubicación en la que se encuentra el usuario.

Tabla 2.1.1.3. Requisito funcional 3.

Nº	ATRIBUTO	CONTENIDO/EXPLICACIÓN
1	Identificador	RQ-F04
2	Nombre	Buscar profesor por criterio seleccionado
3	Versión	1.0
4	Historia de cambios	
5	Prioridad	Alta
6	Descripción	La aplicación permitirá buscar un profesor o profesores del campus usando un determinado criterio de búsqueda (correo, alias, nombre o apellido)

Tabla 2.1.1.4. Requisito funcional 4.

Nº	ATRIBUTO	CONTENIDO/EXPLICACIÓN
1	Identificador	RQ-F05
2	Nombre	Mostrar profesorado
3	Versión	1.0
4	Historia de cambios	
5	Prioridad	Alta
6	Descripción	La aplicación mostrará el listado del profesorado existente en el edificio. Por cada profesor del listado aparecerá su nombre y apellidos y su despacho.

Tabla 2.1.1.5. Requisito funcional 5.

Nº	ATRIBUTO	CONTENIDO/EXPLICACIÓN
1	Identificador	RQ-F06
2	Nombre	Seleccionar tipo destino
3	Versión	1.0
4	Historia de cambios	
5	Prioridad	Media
6	Descripción	El usuario podrá seleccionar, a través de un menú, el destino al que desea llegar. Este menú estará clasificado en Edificios, Aulas, Profesorado y Servicios.

Tabla 2.1.1.6. Requisito funcional 6.

Nº	ATRIBUTO	CONTENIDO/EXPLICACIÓN
1	Identificador	RQ-F07
2	Nombre	Mostrar ficha profesorado
3	Versión	1.0
4	Historia de cambios	
5	Prioridad	Media
6	Descripción	La aplicación mostrará una ficha del profesor elegido mostrando su nombre y apellidos, despacho, así como una fotografía.

Tabla 2.1.1.7. Requisito funcional 7.

Nº	ATRIBUTO	CONTENIDO/EXPLICACIÓN
1	Identificador	RQ-F08
2	Nombre	Calcular ruta de fácil acceso
3	Versión	1.0
4	Historia de cambios	
5	Prioridad	Opcional
6	Descripción	La aplicación permitirá calcular aquella ruta libre de barreras arquitectónicas existente entre la ubicación del usuario y el destino seleccionado.

Tabla 2.1.1.8. Requisito funcional 8.

2.1.2. Requisitos no funcionales

Se han añadido algunos requisitos no funcionales para añadirle valor y calidad a la aplicación.

La interfaz debe ser clara, intuitiva y debe evitar el exceso de opciones que provoque dudas al usuario. No debería haber más de dos opciones de configuración por pantalla (Elección de tipo de ruta, elección de tipo de destino, búsquedas...)

La aplicación debe asegurar el acceso íntegro a los datos almacenados en la base de datos. Debe evitar ataques de inyección SQL y controlar los datos introducidos por el usuario. Este requisito está englobado en el apartado de seguridad.

La aplicación debe desarrollarse realizando un diseño que pueda ser ampliado en un futuro (plantas, edificios, servicios...) sentando las bases de una aplicación de mayor envergadura. Este requisito está englobado en el apartado de escalabilidad

La aplicación debe evitar almacenar datos innecesarios o de gran tamaño en el terminal que provoquen una mala experiencia al usuario. Evitará el exceso de imágenes a la hora de mostrar la realidad aumentada. Este requisito está englobado en el apartado de rendimiento.

2.2. Tecnologías a utilizar

En este apartado se introducirán las herramientas, bibliotecas y tecnologías de las que se van a hacer uso en el desarrollo de la aplicación.

En la actualidad, es difícil encontrar un problema con una única solución. Lo mismo ocurre en programación, se dispone de una gran cantidad de opciones entre las que decantarse a la hora de resolver un problema. La elección de ésta puede basarse en temas de rendimiento, cantidad, coste o incluso de gusto, por lo que en este apartado veremos algunas de las posibilidades disponibles y la elección definitiva.

2.2.1. Plataformas de desarrollo

En cuestión a las plataformas de desarrollo existen varias alternativas en el mercado. Tres son los mayores competidores en el mercado actual de tecnología móvil (Android, iOS y Windows) y cada uno de ellos utiliza lenguajes de programación distintos para sus desarrollos.

En la figura 2.2.1 podemos observar la cuota de mercado en España de cada una de las alternativas.

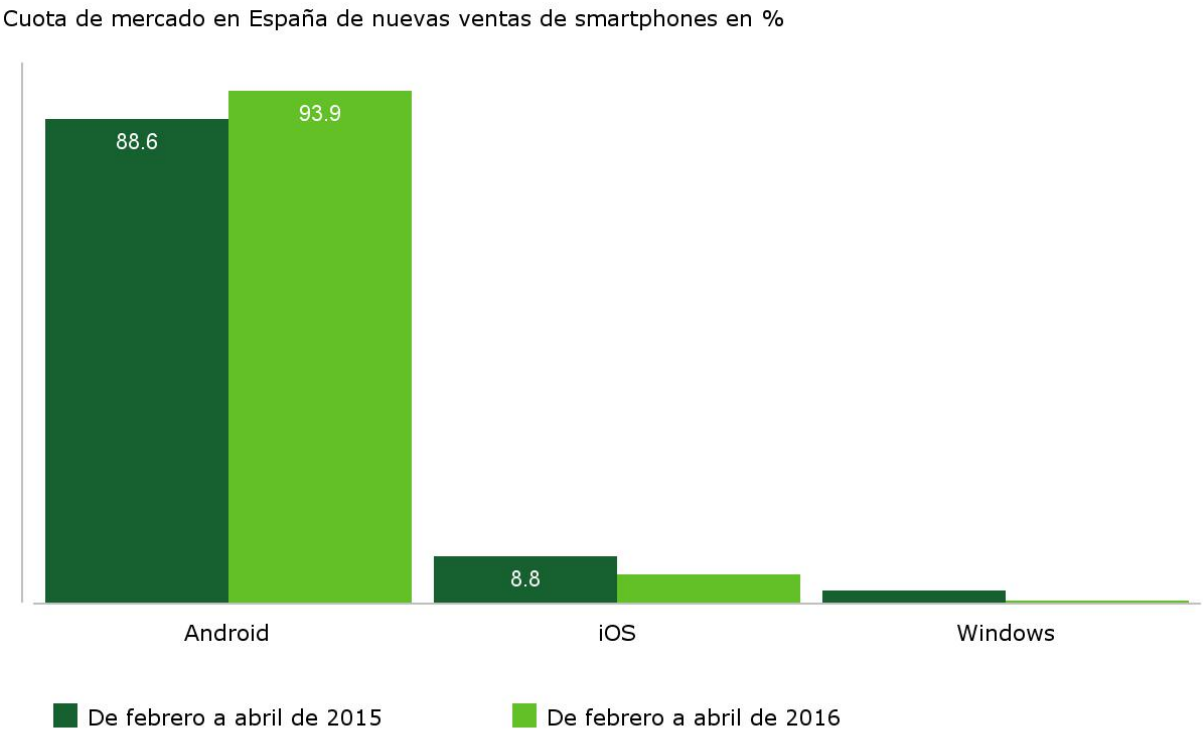


Ilustración 1.1 Cuota de mercado smartphones.

[REF2.2.1.1]

Como se puede observar en la figura 2.2.1, Android abarca más de un 90% de ventas en el mercado español, aumentando en un 5% las ventas en el mismo periodo del año anterior. Muy por debajo quedan los terminales iOS con menos de un 7% y casi sin ventas los terminales con sistema operativo Windows.

Con estos datos se decide utilizar Android como plataforma de destino de la aplicación. Uno de los motivos más importantes de dicha elección es el nicho de mercado del que dispone Android y, pese a no ser una aplicación de uso comercial, si deseamos que sea accesible para el mayor número de personas posible.

Android es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil (teléfonos inteligentes, tabletas, relojes...). Inicialmente fue desarrollado por Android Inc., pero más tarde esta empresa fue comprada por Google. El lenguaje utilizado para programar de forma nativa es JAVA, aunque también es posible utilizar otro tipo de lenguajes como C#. [REF2.2.1.2]

Una vez elegido el sistema operativo sobre el que desarrollaremos la aplicación, pasamos a elegir la versión del mismo. Para ello, pasamos a ver el porcentaje de distribución de cada una de las versiones.

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.0%
4.1.x	Jelly Bean	16	7.2%
4.2.x		17	10.0%
4.3		18	2.9%
4.4	KitKat	19	32.5%
5.0	Lollipop	21	16.2%
5.1		22	19.4%
6.0	Marshmallow	23	7.5%

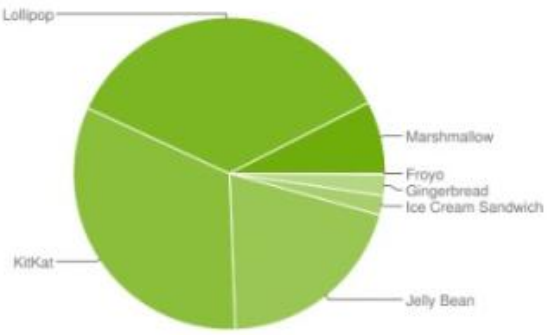


Ilustración 1.1 Porcentaje de distribución versiones Android.

[REF2.2.1.3]

Podemos observar que en el momento de realizar este trabajo, la versión Lollipop es la más extendida por todo el mundo (dispone de un 35.6% de cuota), mientras que las primeras versiones de Android tienen en la actualidad muy poco porcentaje de uso.

Se decide utilizar la versión Jelly Bean debido a que no existen grandes diferencias con la versión Lollipop y además aún sigue instalado en un 20% de terminales móviles, lo que permitirá que nuestra aplicación sea soportada por más de un 90% de terminales Android.

2.2.2. Entorno de desarrollo

Para programar en Android disponemos de Eclipse y Android Studio. Pese a que personalmente estoy más familiarizado con Eclipse, Android Studio no presenta una imagen muy distinta a Eclipse, y desde 2014 (fecha en la que se hizo su lanzamiento oficial) reemplazo a éste como entorno de desarrollo oficial. Es por esto que se ha utilizado para la realización de este trabajo.

Ilustración 2.2.2. Logotipo Android Studio

2.2.3. Localización

La aplicación deberá conocer la posición del usuario para realizar la búsqueda de rutas. Para realizar esta labor existen varias alternativas distintas que pueden ser utilizadas.

2.2.3.1. Localización por GPS.

"El sistema de posicionamiento global (GPS) es un sistema que permite determinar en toda la Tierra la posición de un objeto (una persona, un vehículo) con una precisión de hasta centímetros (si se utiliza GPS diferencial), aunque lo habitual son unos pocos metros de precisión. Para determinar las posiciones en el globo, el sistema GPS se sirve de 24 satélites y utiliza la trilateración." [REF2.2.3.1]

Esta opción queda descartada debido a que el GPS no funciona en interiores ya que la señal de los satélites no es capaz de atravesar la estructura de los edificios. En el caso de la aplicación, la localización dentro del edificio sería imposible realizarla con esta tecnología; sin embargo podría ser utilizada en el caso de ampliarse la aplicación a los exteriores del campus.

2.2.3.2. Localización por tecnología Situm.

En segundo lugar se encuentra una alternativa novedosa, creada por un grupo de investigadores de robótica móvil de la Universidad de Santiago conocida como Situm. Ellos mismos la definen como el "GPS" para interiores. [REF2.2.3.2.1]

En la referencia al enlace podemos encontrar información para desarrollar nuestras propias ideas de aplicación. [REF2.2.3.2.2]

De acuerdo con su web,

"Situm se basa en las redes WiFi del edificio. A eso le añade las señales de los beacons, que son unos novedosos dispositivos diseñados para emitir continuamente una señal WiFi o bluetooth de baja energía (BLE) y un identificador único para poder trabajar como baliza. Y tercero, Situm utiliza el propio campo magnético de cada edificación.

Estas tres señales se combinan con el movimiento del teléfono del usuario a través de los sensores inerciales del 'smartphone' como son el acelerómetro, la brújula o el giroscopio." [REF2.2.3.2.3]

Esta segunda opción también queda descartada debido a la cantidad de componentes que debe de tener activados nuestro terminal para realizar la localización. El uso del WiFi, bluetooth y el necesario estudio del campo magnético de la edificación provoca la inviabilidad de esta tecnología.

2.2.3.3. Localización por Códigos QR

Un código QR (del inglés Quick Response code) es un módulo para almacenar información en una matriz de puntos o en un código de barras bidimensional. Presenta tres cuadrados en las esquinas que permiten detectar la posición del código al lector. Fue creado en 1994 por la compañía japonesa Denso Wave, subsidiaria de Toyota, con el objetivo de que el código permitiera que su contenido se leyera a alta velocidad.

En la actualidad los códigos QR se encuentran en innumerables sitios, desde los que podemos encontrar en los supermercados en cada uno de sus artículos, hasta en revistas, tarjetas de presentación, anuncios publicitarios...

Un detalle importante sobre esta tecnología es que su código es abierto y sus derechos de patente (propiedad de Denso Wave) no se ejercen, por lo que gracias a nuestros smartphones podemos recuperar la información cifrada de estos códigos. Para ello, solo es necesario una de las muchas aplicaciones que permiten a través del dispositivo descifrar el código. [REF2.2.3.3.1]

En la Ilustración 2.2.3.3.1 podemos observar las distintas partes en las que un código QR almacena su información:

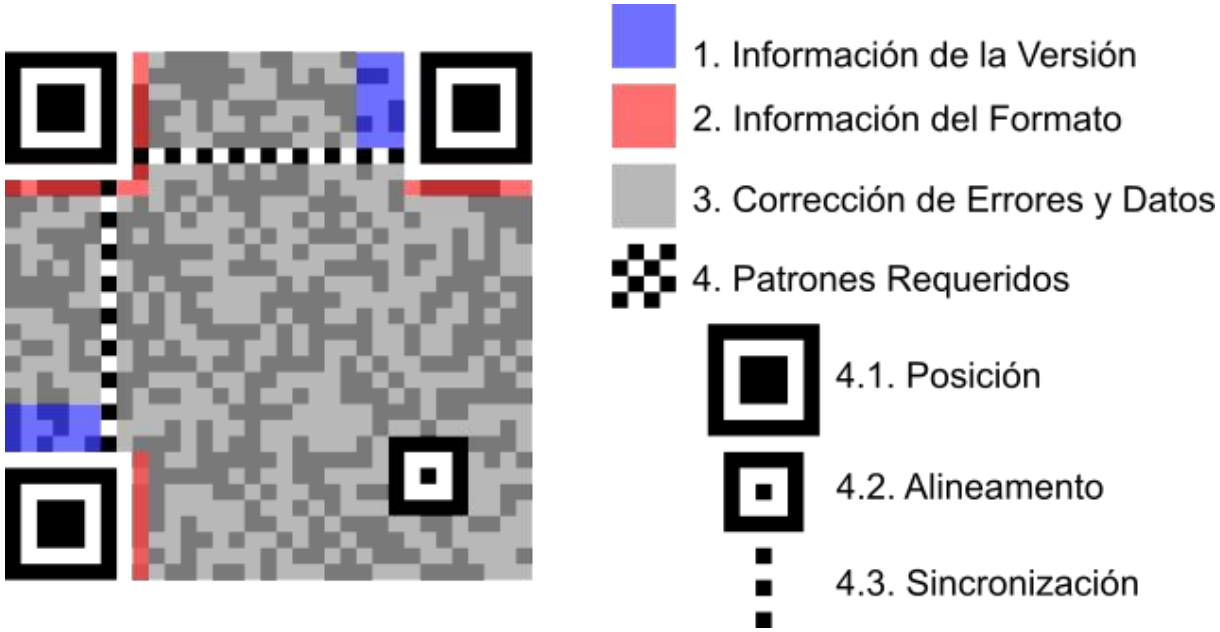


Ilustración 2.2.3.3.1. Partes código QR.

Esta alternativa es la más sencilla y la que menos coste supone a la batería del smartphone debido a que únicamente sería necesario el escaneo del código QR a través de la cámara del terminal para obtener la información almacenada en éste.

Para crear los códigos QR que utilizará la aplicación existen multitud de páginas web que ofrecen este servicio, pudiendo almacenar datos de todo tipo en los códigos. Para los utilizados en esta aplicación se ha utilizado la aplicación “generador QR code”, disponible en:

http://www.qrcode.es/es/generador-qr-code/

Para la lectura de los códigos QR se valoran dos alternativas.

Existe la posibilidad de integrar junto con la aplicación desarrollada, otra llamada Barcode Scanner. Esta opción requiere el uso de esta aplicación mientras se trabajaba con la desarrollada, lo que crea un cambio de interfaz para el usuario, así como una dependencia de otra aplicación que podía provocar problemas en un futuro si se quiere seguir desarrollando la idea.

Por otro lado, se dispone de una biblioteca de código abierto (ZXing), que puede integrarse en la propia aplicación, solucionando los problemas que causa la otra opción.[REF2.2.3.3.2]

ZXing es una biblioteca procesador de imágenes multi-formato en 1D/2D de código abierto. Permite reconocer múltiples formatos. En nuestro caso, se ha utilizado para el escaneo de los códigos QR que indicarán la posición del usuario dentro de los edificios.[REF2.2.3.3.3]

2.2.4. Realidad Aumentada

Realidad Aumentada es la incorporación de datos e información digital en un entorno real, por medio del reconocimiento de patrones que se realiza mediante un software.

La primera constancia de Realidad Aumentada que se tiene es en 1962, cuando Morton Heilig creó un simulador de moto llamado “Sensorama” con imágenes, sonido, vibración y olfato. Posteriormente, Ivan Sutherland inventa el head-mounted display (HMD) lo que abre una ventana al mundo virtual. De ahí partimos hasta llegar a los últimos avances de la industria, como las gafas de realidad aumentada de Google o el primer juego con esta tecnología para móviles, Ingress, desarrollado por Niantic y Google, que fue el antecesor de Pokemon GO. [REF2.2.4]

2.2.4.1. Tecnología

Los dispositivos de Realidad Aumentada normalmente constan de un sistema de visualización (gafa o similar) para mostrar al usuario la información virtual que se añade a la real. El sistema de visualización lleva incorporado sistemas de GPS, necesarios para poder localizar con precisión la situación del usuario.

Los sistemas de Realidad Aumentada modernos utilizan una o más de las siguientes tecnologías: cámaras digitales, sensores ópticos, acelerómetros, GPS, giroscopios...

[REF2.2.4.1]

2.2.4.2. Niveles

Pueden definirse distintos grados de complejidad que presentan las aplicaciones basadas en Realidad Aumentada.

Nivel 0: Las aplicaciones hiperenlazan el mundo físico mediante el uso de códigos de barras y 2D (por ejemplo, los códigos QR).

Nivel 1: Las aplicaciones utilizan marcadores para el reconocimiento de patrones 2D. La forma más avanzada de este nivel también permite el reconocimiento de objetos 3D.

Nivel 2: Las aplicaciones sustituyen el uso de los marcadores por el GPS y la brújula de los dispositivos móviles para determinar la localización y orientación del usuario y superponer “puntos de interés” sobre las imágenes del mundo real.

Nivel 3: Estaría representado por dispositivos como Google Glass, capaces de ofrecer una experiencia completamente contextualizada, inmersiva y personal.

[REF2.2.4.2]

2.2.4.3. Bibliotecas

Existen en el mercado numerosas bibliotecas que permiten incorporar Realidad Aumentada en nuestros dispositivos. Cada una ofrece unas características o requiere de alguna otra herramienta para realizar los diseños que aparecerán en los terminales. En el siguiente enlace se muestra una tabla comparativa de las más conocidas:

http://socialcompare.com/es/comparison/augmented-reality-sdks

Durante el desarrollo de esta aplicación se han estudiado algunas de estas alternativas. En primer lugar, se estudió la biblioteca Vuforia que se encuentra disponible para Android Studio. Tras estudiar un poco las características, así como la documentación disponible en la web (oficial y no oficial) se observa el potencial de la herramienta. Tiene un fantástico reconocimiento de target y permite mostrar elementos en 3D con bastante facilidad con el uso de Unity. Esta herramienta está más enfocada al desarrollo de videojuegos, por lo que resulta más complicada integrarla con nuestra aplicación.[REF2.2.4.3.1] [REF2.2.4.3.2]

Por otra parte, se encuentra Wikitude y se observa, en algunas aplicaciones que dispone de ejemplo, que realiza de una manera muy sencilla las tareas que debe realizar la aplicación (reconocimiento de patrones, muestra de objetos 2D en los target...). Por todo esto, tras visualizar la documentación oficial, así como algunos tutoriales, se elige Wikitude como la biblioteca a utilizar. [REF2.2.4.3.3] [REF2.2.4.3.4]

Wikitude es una SDK gratuita para proyectos no comerciales (debemos adquirir una licencia anual en caso contrario) que permite generar Realidad Aumentada en nuestra aplicación. Wikitude no es un SDK nativa para Android, por lo que se basa en el concepto de ARchitect worlds para crear la Realidad Aumentada. Este concepto consiste en la creación de páginas HTML que utilizan un architectView que actúa como una superficie en la cámara donde se manejan los eventos. Con JavaScript podremos interactuar con los elementos creados y con CSS podemos personalizar nuestro entorno.

Ilustración 2.2.4.3 Logotipo Wikitude

2.3. Presupuesto

En este apartado se hará una estimación de los costes que supondrá el desarrollo de la aplicación. Se van a detallar los costes tanto de personal como aquellos relacionados con el software y el hardware implicados en el desarrollo. Además se añadirán los costes de recursos utilizados durante el proceso de desarrollo.

2.3.1. Planificación inicial

Se describe la planificación inicial durante la fase de estudio del problema. La finalidad de este planificación es poder analizar las horas que dedicarán los miembros del equipo a cada una de las tareas necesarias y así poder realizar una estimación del coste que supondrá su trabajo.

Definición y análisis del proyecto:

-Estudio del proyecto: Reunión con el tutor para describir el problema, aclarar objetivos y dar pautas sobre tecnologías interesantes para la resolución del problema.

Tiempo estimado : 4 horas.

-Estudio de tecnologías: Esta tarea permitirá adquirir conocimientos y material necesario para la realización del trabajo. Incluye el tiempo de prueba con dichas

tecnologías. Al ser campos de estudio nuevos para el desarrollador tendrán una duración mayor.

Tiempo estimado: 48 horas.

Análisis del sistema:

-Requisitos del sistema: Tarea de recopilación de requisitos necesarios para la solución al problema. Descripción de éstos y análisis de posibles dependencias entre ellos.

Tiempo estimado: 8 horas.

-Diagramas de caso de uso: Realización de los diagramas de casos de uso correspondientes a los requisitos obtenidos.

Tiempo estimado: 4 horas.

-Diagramas de secuencia: Realización de los diagramas de secuencia de cada uno de los requisitos obtenidos.

Tiempo estimado: 12 horas.

-Presupuesto: Elaboración de una estimación de presupuesto para la aplicación.

Tiempo estimado: 12 horas.

Diseño del sistema:

-Diseño de la interfaz: Tarea que se encarga de las interacciones entre pantallas y diseño de éstas.

Tiempo estimado: 24 horas.

-Diseño de datos: Tarea encargada de elegir la mejor alternativa al diseño de datos, la creación de base de datos o alojamiento en servidores.

Tiempo estimado: 20 horas.

-Diseño de aplicación: Tarea encargada de la realización de los diseños en UML (diagramas de paquetes, diagrama de clases).

Tiempo estimado: 24 horas.

Implementación:

-Interfaz: Realización de las distintas interfaces diseñadas en tareas previas.

Tiempo estimado: 40 horas.

-Capa de datos: Realización de las distintas clases y métodos que comunican el modelo con la base de datos.

Tiempo estimado: 40 horas.

-Lógica de negocio: Realización de las distintas clases y métodos que otorgan funcionalidad a la aplicación.

Tiempo estimado: 60 horas.

Pruebas y métricas. Tiempo estimado: 24 horas

Documentación. Tiempo estimado: 60 horas.

Presentación. Tiempo estimado: 12 horas.

El tiempo estimado necesario para el desarrollo del proyecto asciende a 392 horas.

2.3.2. Diagrama de Gantt

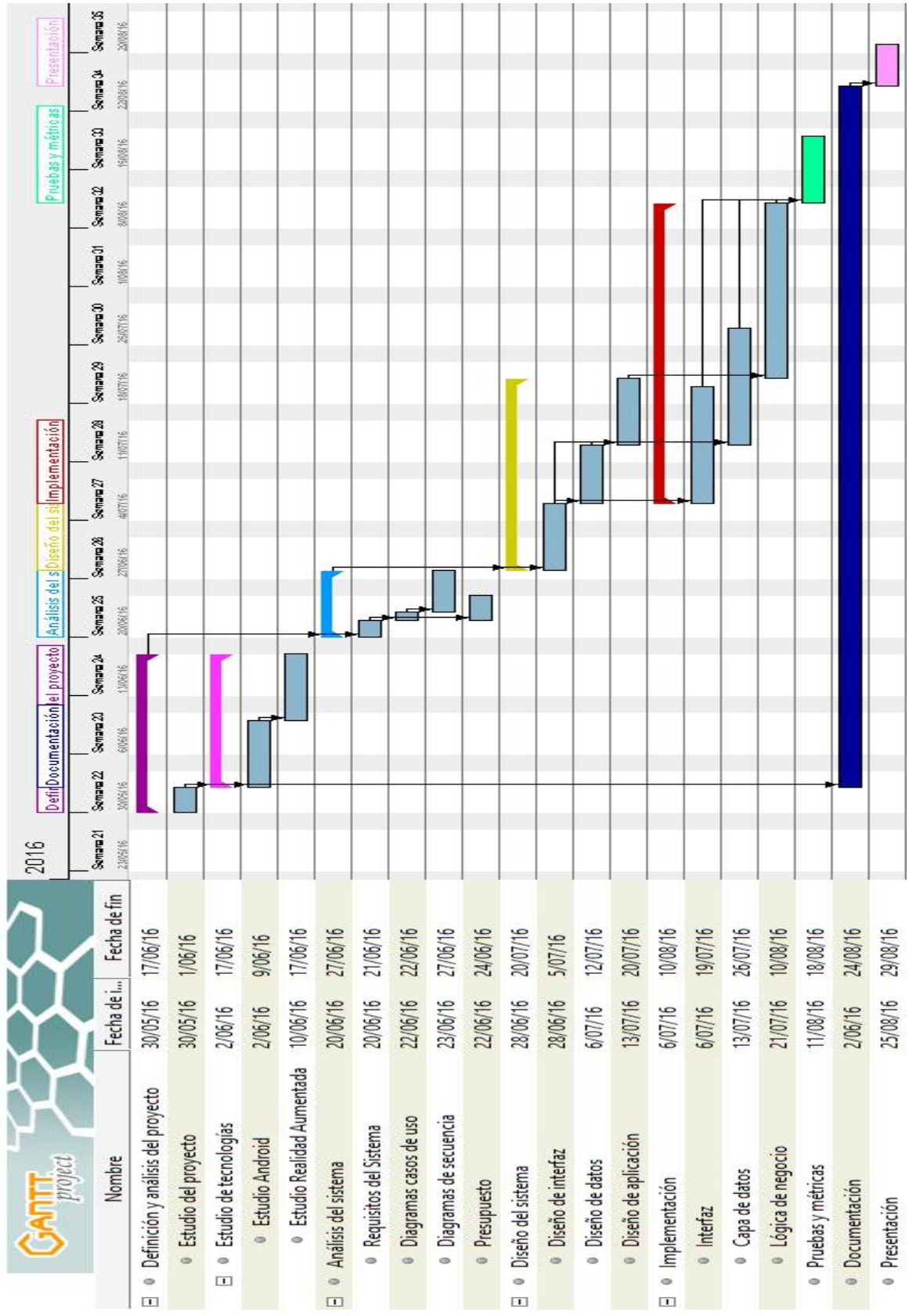


Ilustración 2.3.2 Diagrama de Gantt.

En la ilustración 2.3.2 se muestra el diagrama de Gantt correspondiente a las tareas a realizar para el desarrollo de la aplicación.

2.3.3. Costes de personal

Para el cálculo de costes de personal se tomarán en cuenta los convenios colectivos correspondientes a la rama de Informática. Los datos obtenidos de estos convenios son sueldos mensuales, por lo que se ha aplicado un incremento al realizarse un trabajo por horas. [REF2.3.3.1] [REF2.3.3.2]

Categoría	Tareas	Horas	Horas totales	Euros / hora	Coste total
Jefe de proyecto	-Estudio del proyecto -Requisitos del sistema	4 4	8	35	280
Analista Software	-Análisis del sistema -Diseño de datos -Diseño de aplicación	32 20 24	76	25	1900
Ingeniero Senior	-Capa de datos -Lógica de negocio	20 30	50	20	1000
Ingeniero Junior	-Estudio de tecnologías -Diseño de interfaz -Interfaz -Capa de datos -Lógica de negocio -Documentación -Presentación	48 24 40 20 30 60 12	234	15	3510
Testeador	-Pruebas y métricas	24	24	12	288
Total					6978 €

Tabla 2.3.3. Costes del personal por categoría.

2.3.4. Costes elementos software y hardware

El coste de los elementos software y hardware se refiere a los diversos equipos informáticos que se estiman necesarios para el desarrollo de la aplicación. En la siguiente tabla se muestra el producto, su precio, el periodo de amortización en meses y el tiempo que se ha utilizado.

Producto	Precio	Periodo de amortización	Periodo de uso del producto	Coste para el proyecto
Portátil ASUS F556UA-XO063T / Windows 10 Home / i7-6500U / HD Graphics 520 / 4GB RAM / 500GB HDD / 15.6" [REF2.3.4.1]	549,00€	48	3	34,3125
Logitech M510 - Ratón (USB, inalámbrico), Negro [REF2.3.4.2]	35,13€	12	3	8,7825
TOTAL				43,10€

Tabla 2.3.4. Costes de elementos software y hardware.

2.3.5. Otros

En este apartado se detallarán los gastos correspondientes a recursos utilizados, como luz, conexión a Internet...

Producto	Precio	Periodo de amortización	Periodo de uso del producto	Coste para el proyecto
Luz Endesa	60€ / mes	3 mes	8 horas / día	60 €
Fibra óptica 30 MB [REF2.3.5]	14,90€ / mes	3 mes	8 horas / día	14,90 €
TOTAL				74,90 €

Tabla 2.3.5. Costes varios.

2.3.6. Presupuesto final

Concepto	Importe
Costes de personal	6.978 €
Costes de elementos software y hardware	43,10 €
Otros gastos	74,90 €
Suma sin I.V.A.	7.096 €
Beneficio 25%	1.774 €
Total sin I.V.A.	8.880 €
I.V.A.	1.864,80 €
Total a facturar	10.744,80 €

Tabla 2.3.6. Presupuesto final.

3. Diseño

En este apartado se mostrará todo lo relacionado con los diseños de las distintas partes de la aplicación. En primer lugar, se mostrará el diseño de datos con sus respectivos diagramas.

3.1. Diseño de datos

En cuanto a la elección del mejor diseño de datos para aplicar al proyecto se partía de tres posibilidades distintas:

-Guardar datos, imágenes e infraestructuras en la memoria del teléfono.

-Guardar los datos en una pequeña base de datos interna que permitiera descargar de la memoria de la aplicación los datos más comprometidos y tenerlos estructurados de una manera más eficiente y a la vez segura.

-Guardar los datos en un servidor y acceder a ellos conforme fueran necesarios.

Con la primera opción, la aplicación sería muy sencilla en cuestión de acceso a los datos e imágenes, pero la aplicación podría volverse lenta en situaciones en los que el terminal no tuviera gran capacidad, por ejemplo en el listado de todo el profesorado con sus respectivas imágenes, por lo que provocaría una peor experiencia para el usuario. Se opta por un enfoque híbrido entre las tres opciones. Se almacenaran los datos en una pequeña base de datos, proporcionando un acceso seguro a ellos. Las imágenes, dependiendo del uso de éstas, serán almacenadas en la memoria del teléfono o por el contrario serán descargadas en tiempo de ejecución por la aplicación desde las urls almacenadas en la base de datos. De esta manera, por ejemplo, evitamos tener que guardar en la memoria del teléfono todas las imágenes de los profesores y únicamente se descargará la imagen del profesor elegido. La desventaja de esta opción sería el uso de Internet, pero teniendo en cuenta que el campus de la Universidad de Jaén cuenta con una red inalámbrica de acceso para sus usuario, éste no sería un gran problema.

3.1.1. Diseño de la base de datos

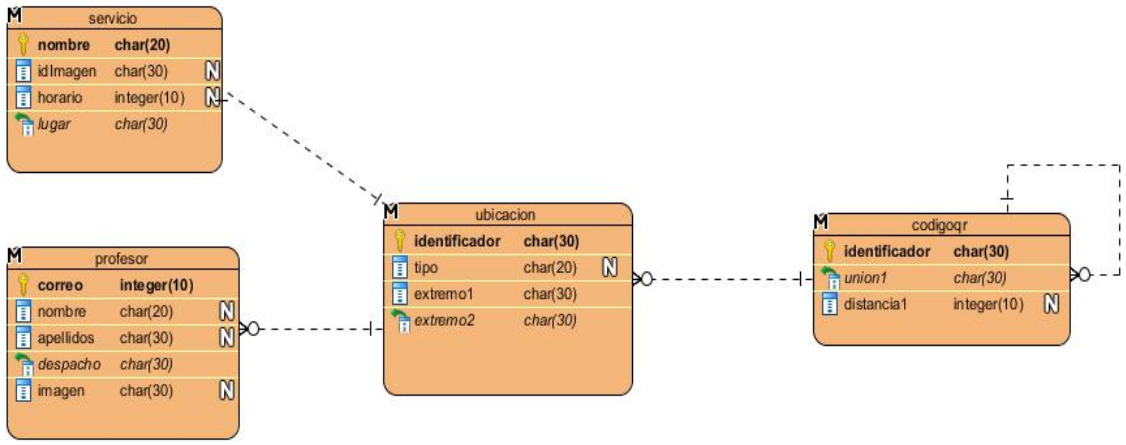


Ilustración 3.1.1. Diagrama entidad- relación base de datos.

La base de datos cuenta con cuatro tablas donde se almacenará la información de la aplicación.

La tabla codigoqr almacenará la información relativa a los códigos QR que se situarán en las distintas intersecciones del edificio, almacenando para cada uno de ellos su identificador y los códigos QR colindantes a él con sus respectivas distancia.

La tabla ubicación representa la ubicación de un posible destino de las rutas de la aplicación. Almacenará el tipo de objetivo (Despacho, almacén, aula, laboratorio, entrada...) y las intersecciones que lo rodean, representadas por extremo1 y extremo2 que son las claves foráneas hacia codigoqr.

Por último, las tablas servicio y profesor. Profesor almacena los datos de un profesor particular, mientras que servicio hace referencia a alguno de los servicios prestados por la Universidad de Jaén (cafeterías, biblioteca, bancos...).

Esta base de datos podría ser ampliada para añadir un mayor número de ubicaciones, como aulas, laboratorios, edificios...

3.2. Diseño de aplicación

3.2.1. Diagrama de paquetes

Para el diseño de la aplicación se seguirá el patrón MVC (Modelo-Vista-Controlador). Cada una de las vistas que estarán representadas por las pantallas de la aplicación dispondrán de un controlador que recoja las interacciones que realice el usuario y muestre las distintas opciones disponibles. Los controladores serán los encargados de la comunicación con las clases del modelo.

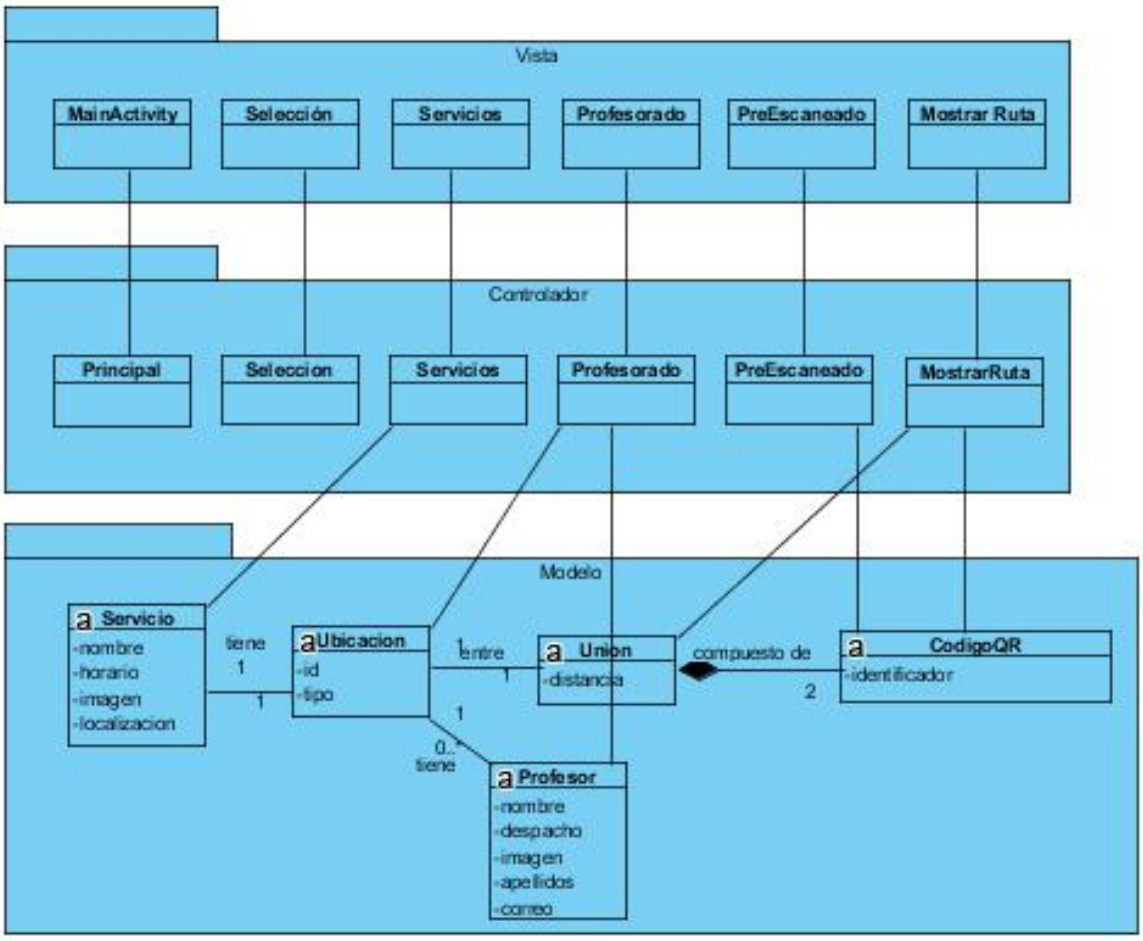


Ilustración 3.2.1. Diagrama de paquetes. Patrón MVC.

3.2.2. Diagrama de clases

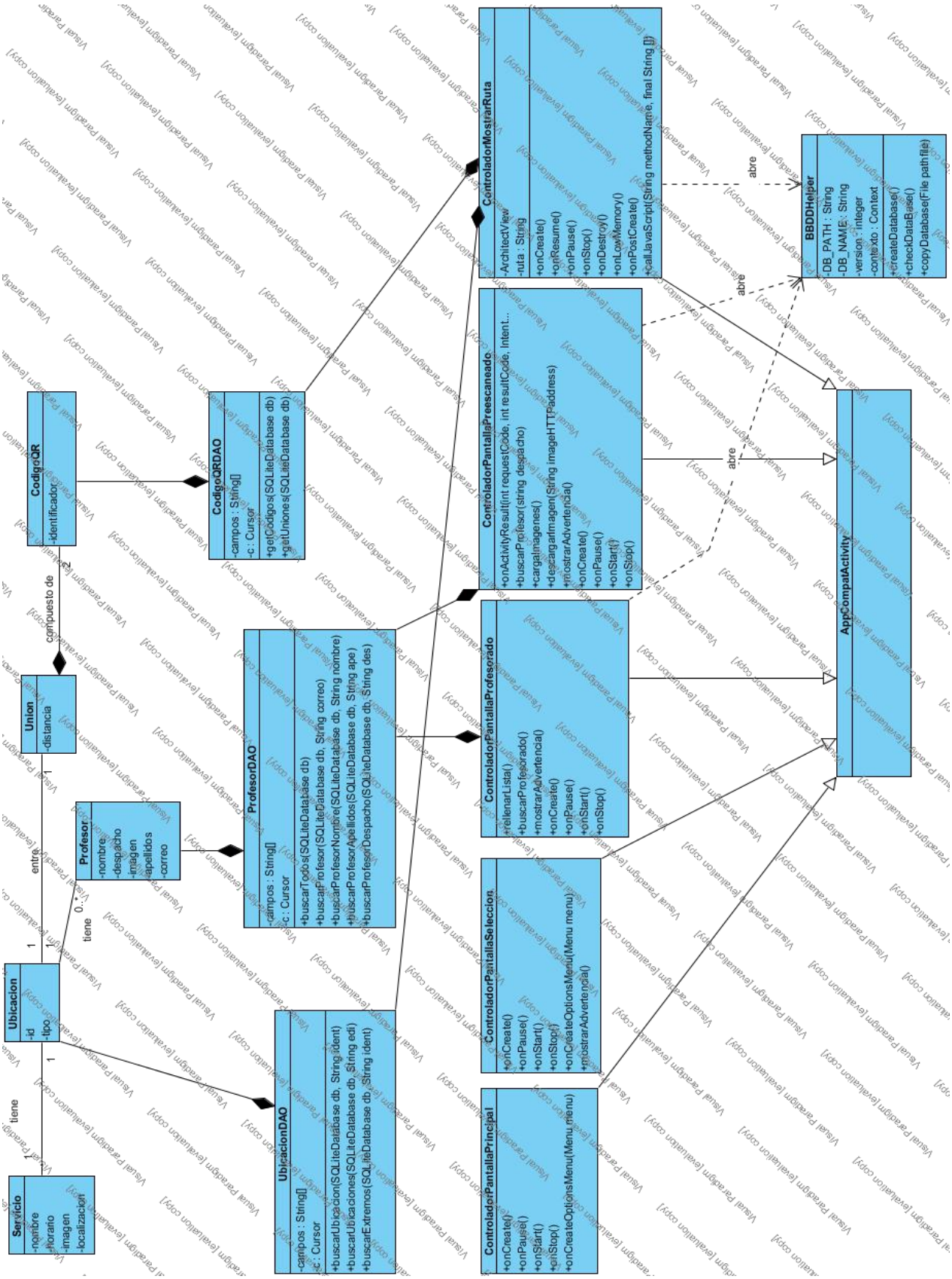


Ilustración 3.2.2. Diagrama de clases.

En la ilustración 3.2.2. se encuentran los cinco controladores de las pantallas que compondrán la aplicación, todos ellos heredan de la clase AppCompatActivity.

La conexión entre la base de datos y la aplicación se realizará a través de cuatro clases. BBDDHelper será la encargada de la creación de la base de datos y de proporcionar una instancia a las tres clases restantes para realizar búsquedas sobre ella. Tanto ProfesorDAO, UbicacionDAO y CodigoQRDAO tendrán los métodos implementados para realizar un acceso seguro a la base de datos, incluyendo consultas parametrizadas que impidan ataques de inyección SQL.

El resto de clases representan los elementos comunes con los que se va a trabajar en la aplicación, ubicaciones, profesores, servicios, códigos QR y unión. Esta última representa la arista formada entre dos códigos QR que permitirá formar el grafo para poder ejecutar el algoritmo de Dijkstra.

3.2.3. Diagrama de casos de uso

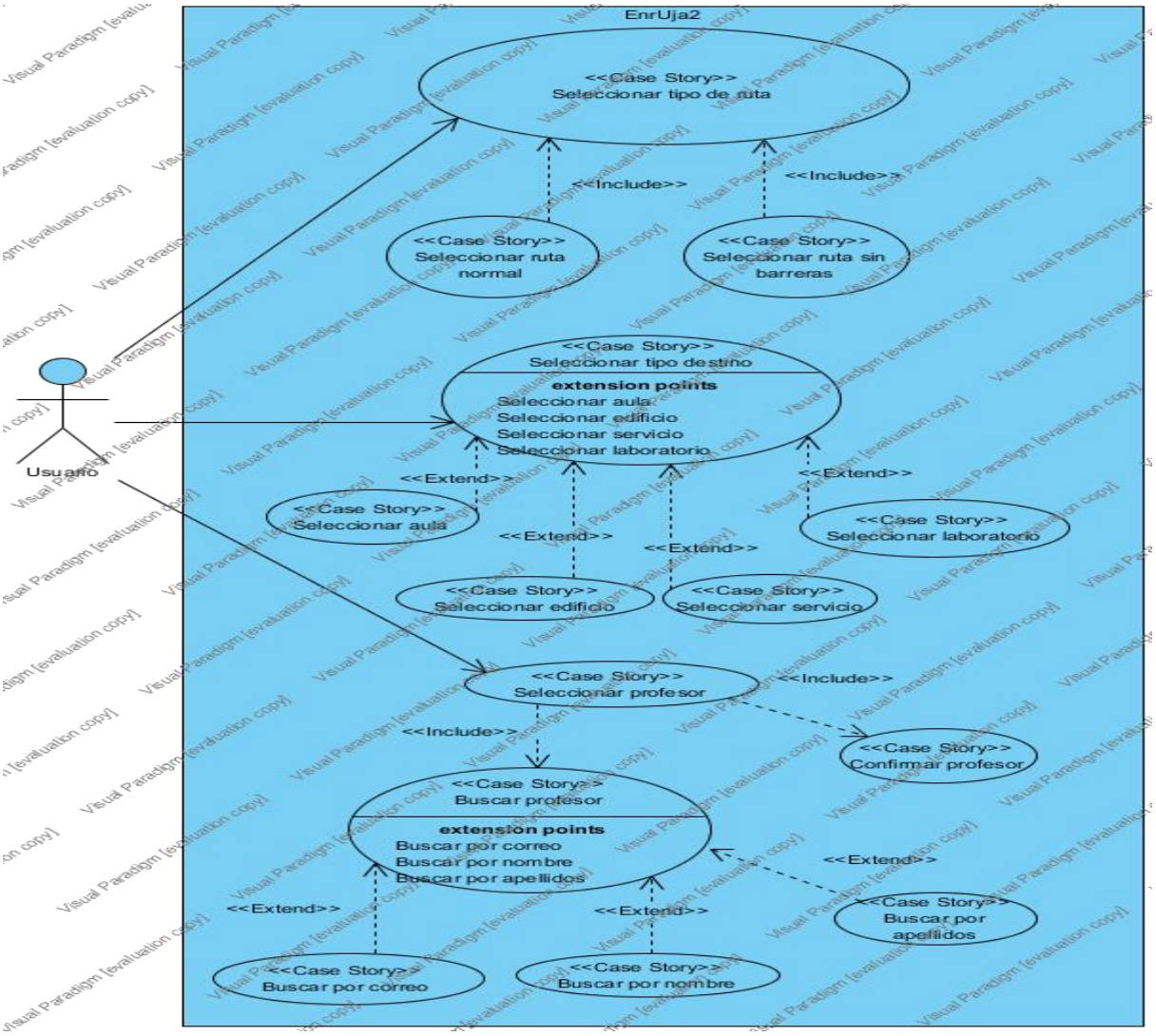


Ilustración 3.2.3. Diagrama de casos de uso.

En la Ilustración 3.2.3. se muestra el diagrama de casos de uso. En él se encuentra los objetivos Seleccionar tipo de ruta, Seleccionar tipo de destino y Buscar profesor.

-En el objetivo Seleccionar tipo de ruta se consideran parte fundamental los objetivos Seleccionar ruta normal y Seleccionar ruta sin barreras para la consecución del objetivo principal, de ahí su relación de inclusión.

-En el objetivo Seleccionar tipo de destino se considera que el resto de objetivos extiende el funcionamiento del objetivo principal, de ahí dicha relación.

-Por último, el objetivo Seleccionar profesor se consideran parte fundamental los objetivos Confirmar profesor y Buscar profesor. Los sub-objetivos Buscar por correo, Buscar por nombre y Buscar por apellidos complementan la funcionalidad del objetivo Buscar profesor.

IDENTIFICADOR:	CU 01	NOMBRE:	SELECCIONAR TIPO DESTINO
Objetivo	Seleccionar el tipo de destino que quiere tomar como punto de llegada.		
Actor	Usuario de la aplicación.		
Precondiciones			
Postcondiciones			
Escenario básico	1. Se accede a la aplicación. 2. Se elige tipo de destino.		

Tabla 3.2.3.1 Descripción textual del caso de uso CU01.

IDENTIFICADOR:	CU 02	NOMBRE:	SELECCIONAR PROFESOR
Objetivo	Seleccionar el profesor con su despacho correspondiente.		
Actor	Usuario de la aplicación.		
Precondiciones	Debe haberse seleccionado el tipo de destino Profesorado.		
Postcondiciones			

Escenario básico	1. Se accede a la aplicación. 2. Se elige el tipo de destino. 3. Se introduce en un campo de entrada de texto el correo, nombre o apellido del profesor. 4a. Se selecciona profesor. 5.Se confirma elección.
Escenario alternativo	4b. Mostrar mensaje no existen profesores que cumplan los criterios de búsqueda

Tabla 3.2.3.2 Descripción textual del caso de uso CU02.

IDENTIFICADOR:	CU 01	NOMBRE:	SELECCIONAR TIPO RUTA
Objetivo	Seleccionar el tipo de ruta que quiere seguir entre la ubicación y el destino.		
Actor	Usuario de la aplicación.		
Precondiciones	Debe haberse seleccionado destino.		
Postcondiciones			
Escenario básico	1. Se accede a la aplicación. 2. Se elige tipo de destino. 3. Se elige destino. 4. Se selecciona tipo de ruta.		

Tabla 3.2.3.3 Descripción textual del caso de uso CU03.

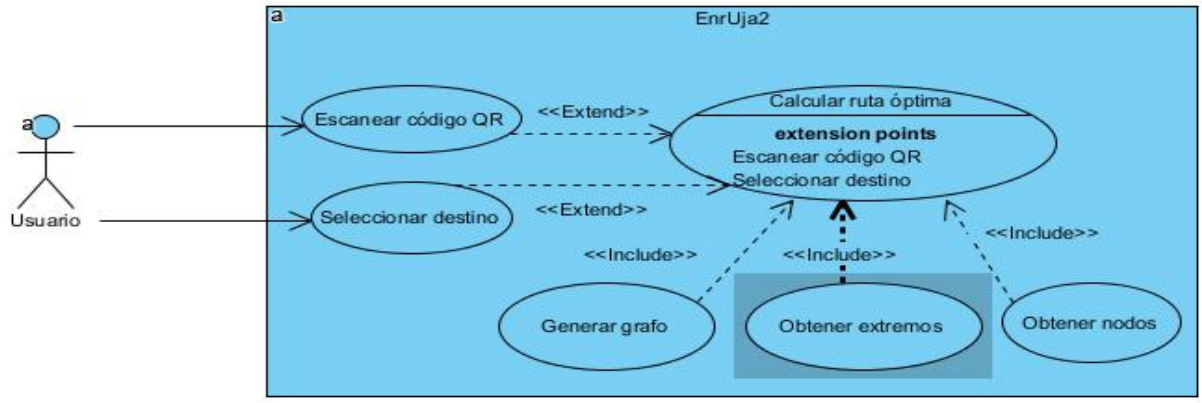


Ilustración 3.2.4. Diagrama de casos de uso. Calcular ruta óptima.

IDENTIFICADOR:	CU 04	NOMBRE:	CALCULAR RUTA ÓPTIMA
Objetivo	Cálculo de la ruta óptima existente entre dos puntos.		
Actor	Usuario de la aplicación.		
Precondiciones	Debe haberse seleccionado destino. Debe haberse escaneado QR. Debe haberse seleccionado tipo de ruta.		
Postcondiciones			
Escenario básico	1. Se accede a la aplicación. 2. Se selecciona tipo de destino. 3. Se selecciona destino. 4. Escanear código QR. 5. Se selecciona tipo de ruta. 6. Generar grafo. 7. Se obtienen los nodos. 8. Se obtienen los extremos del destino. 9. Se calcula ruta óptima. 10. Mostrar ruta óptima		
Escenario alternativo	10b. Destino no alcanzable		

Tabla 3.2.3.4 Descripción textual del caso de uso CU04.

En la Ilustración 3.2.4 os objetivos Generar grafo, Obtener extremos y Obtener nodos se consideran parte esencial del cálculo de la ruta óptima ya que sin éstos no se podría calcular la ruta. Sin embargo Escanear código QR y Seleccionar destino nos sirven de base para realizar los cálculos, tienen un significado fuera de este caso de uso.

3.2.4. Diagramas de secuencia

A continuación se muestran los diagramas de secuencia correspondientes a los distintos requisitos enunciados en capítulos anteriores.

Seleccionar tipo destino

En la Ilustración 3.2.4.1. se puede ver la secuencia de pasos que sigue la aplicación para mostrar la pantalla de destinos elegida por el usuario.

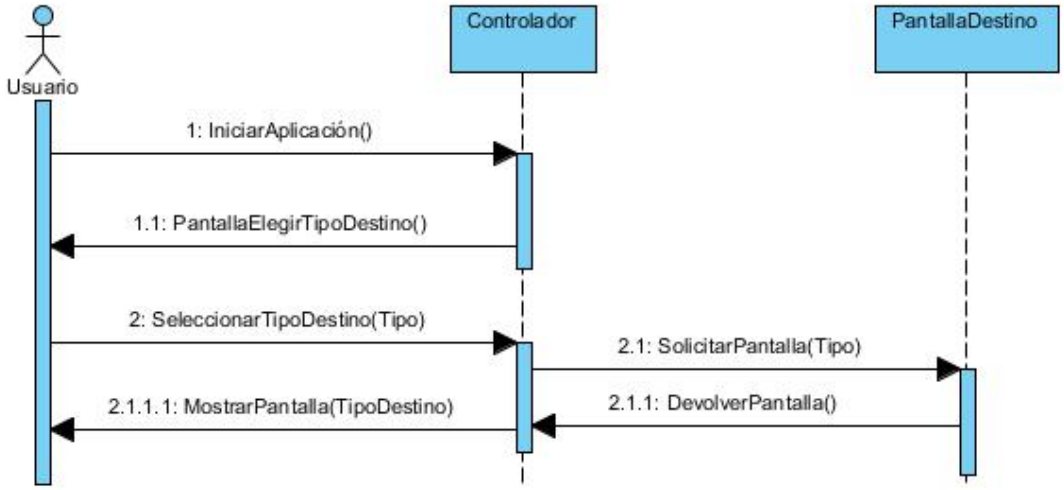


Ilustración 3.2.4.1. Diagrama de secuencia. Seleccionar tipo destino.

Seleccionar tipo ruta (normal o sin barreras arquitectónicas)

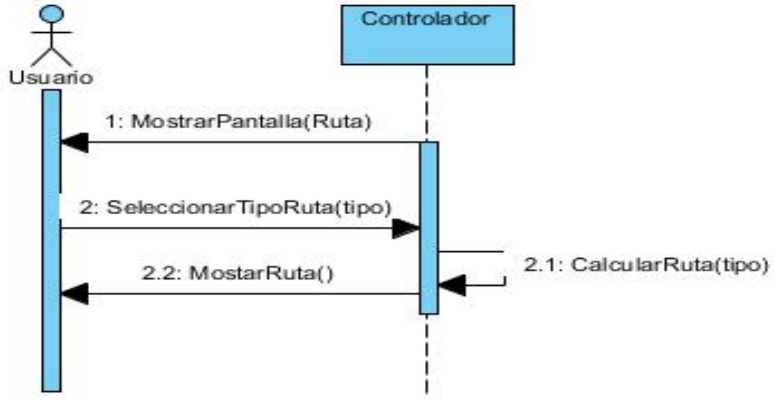


Ilustración 3.2.4.2. Diagrama de secuencia. Seleccionar tipo de ruta

Escanear código QR

En la Ilustración 3.2.4.3 el Usuario una vez elegido el destino deberá escanear el código QR más cercano para obtener su ubicación, para ello, el controlador mostrará la cámara. Una vez escaneado el código QR, se procesará la imagen y se mostrará al usuario la pantalla “seleccionar tipo ruta” para que elija el tipo de ruta.

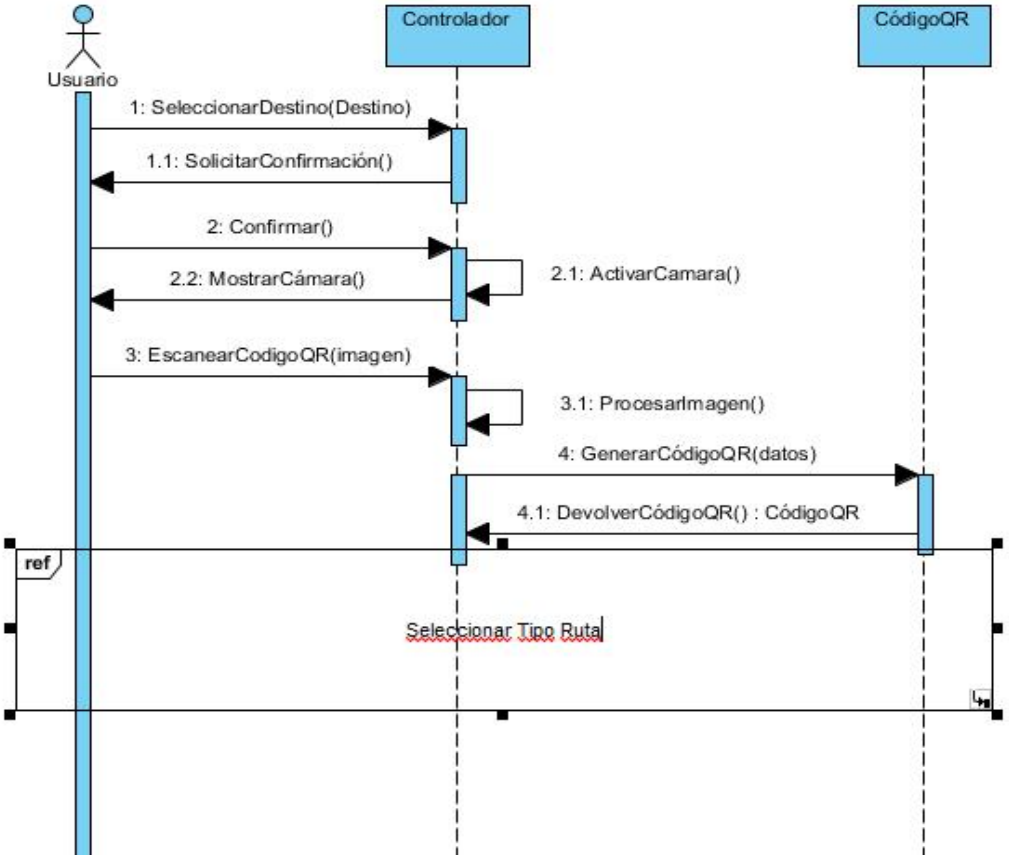


Ilustración 3.2.4.3. Diagrama de secuencia. Escanear código QR

Calcular ruta más corta entre dos puntos

Como se puede ver en la Ilustración 3.2.4.4. una vez el usuario ha introducido su ubicación, el destino y el tipo de ruta, el controlador se encargará de calcular la ruta más corta entre ambos puntos. Para ello generará el grafo correspondiente a la planta del edificio donde se encuentre el usuario. Una vez obtenido, solicitará los extremos del destino para conocer las intersecciones que lo rodean. Por último calculará la ruta más corta.

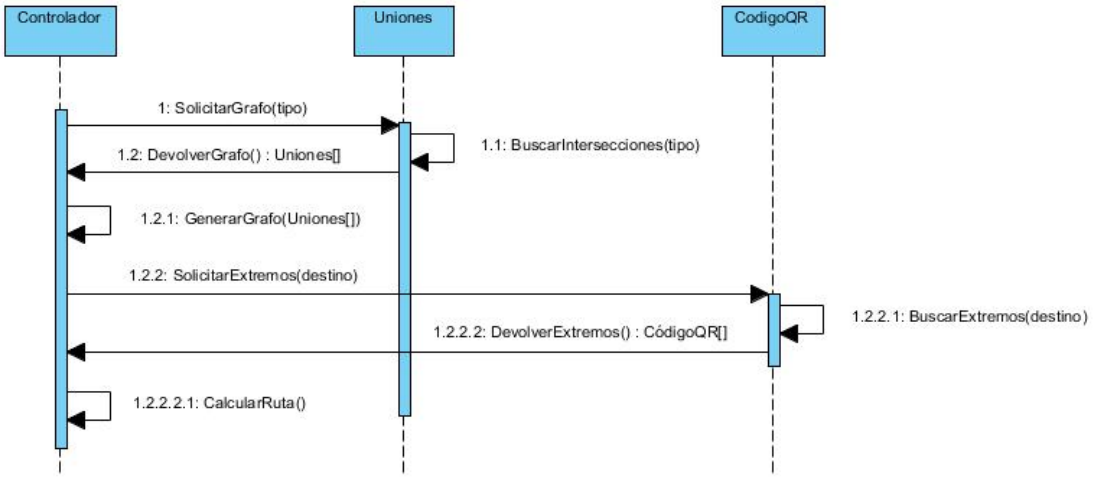


Ilustración 3.2.3.4. Diagrama de secuencia. Calcular ruta más corta.

Mostrar profesorado

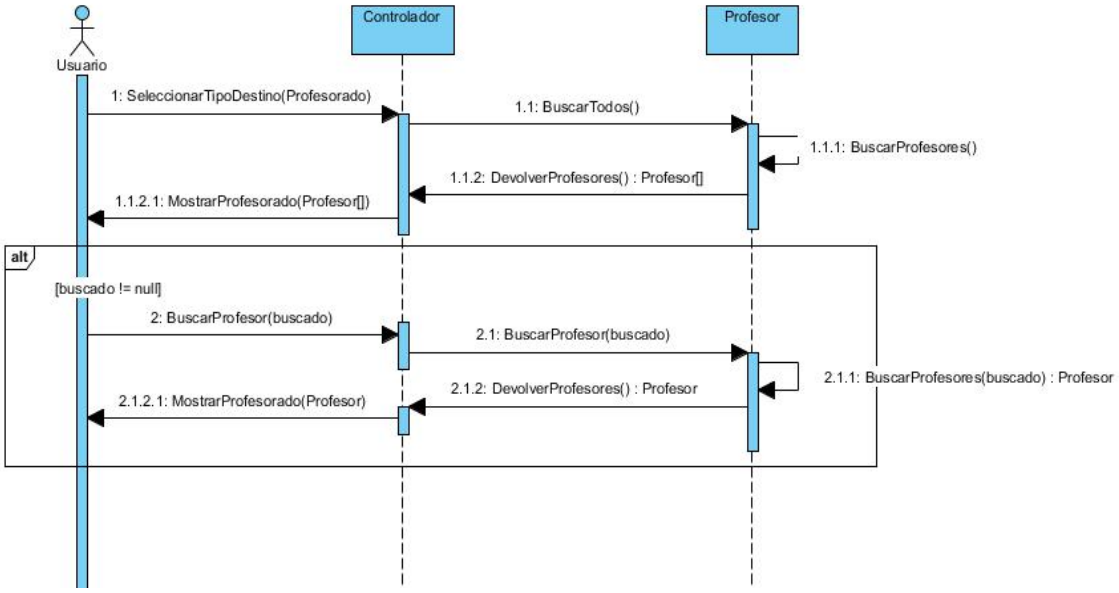


Ilustración 3.2.3.5. Diagrama de secuencia. Mostrar profesorado.

3.3. Diseño de interfaz

En este apartado se detallarán todos los datos correspondientes al diseño de la interfaz de usuario.

3.3.1. Prototipo de interfaz

Para la realización del prototipo se ha utilizado la herramienta de prototipado Axure en su versión 7. Axure es una herramienta orientada a diseñar wireframes y prototipos básicos o avanzados, permitiendo incluso otorgarle funcionalidad para que simulen mejor las interacciones entre pantallas propuestas en el proyecto. Dispone de una amplia gama de widgets para empezar el diseño rápidamente, aunque también es posible crear nuestros propios widgets personalizados. A todo esto cabe añadir que funciona en cualquier navegador por lo que su visualización es muy sencilla. [REF3.3.1]

En las siguientes imágenes se muestra la interfaz que más o menos debería encontrarse el usuario de la aplicación.

Ilustración 3.3.1.1. Pantalla principal.

En esta primera pantalla se mostrará al usuario una pequeña información introductoria acerca de la aplicación. Opcionalmente dispondrá de dos botones que permitirán al usuario realizar su geolocalización a través de GPS o a través de la lectura de un código QR. Debido a que el prototipo se realiza en el interior del edificio A3 la opción desarrollada será la de lectura de códigos QR.

Ilustración 3.3.1.2. Pantalla ubicación

En la siguiente pantalla el usuario podrá conocer su ubicación a través de una imagen o un mensaje informativo. Podrá elegir el tipo de ruta que desea escoger, normal o sin barreras arquitectónicas, realizando la ruta por ascensores o rampas adaptadas. El prototipo se realiza sobre la planta baja del A3 por lo que la opción de sin barreras arquitectónicas permanecerá bloqueada hasta su desarrollo.

Ilustración 3.3.1.3. Pantalla selección

Una vez elegido el tipo de ruta y ubicado el usuario, se pasará a un menú donde se podrá elegir las distintas opciones en las que estarán divididos los distintos destinos a los que podrá solicitar llegar el usuario. Para el desarrollo de la aplicación se utilizará los despachos debido a la mayor cantidad de éstos en la planta baja del A3 y por el mayor interés funcional y pedagógico.

Ilustración 3.3.1.4. Pantalla profesorado **Ilustración 3.3.1.5. Pantalla edificios.**

Las pantallas de selección permitirán elegir el destino del usuario seleccionando las distintas posibilidades. A la izquierda podemos ver la interfaz de búsqueda de despacho y a la derecha la selección del edificio a elegir como destino.

4. Implementación

En este apartado se tratará todos los aspectos que conciernen al tema de la implementación.

4.1. Ubicación

Como se ha mencionado en capítulos anteriores, la ubicación del usuario se realiza a través de la lectura de códigos QR. Para ello se utiliza la biblioteca Zxing. La clase que hace uso de esta biblioteca es PantallaPreescaneado.java y lo único que debemos de hacer para usarla es:

- Importarla en nuestro proyecto

- En el fichero build.gradle de nuestro módulo incluir:

```
compile 'com.journeyapps:zxing-android-embedded:3.2.0@aar'
compile 'com.google.zxing:core:3.2.1'
```

-En el AndroidManifest.xml añadir permiso para usar la cámara del dispositivo

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

Una vez realizado estos pasos, desde la clase que deseemos utilizar el lector debemos incluir los import correspondientes:

```
import com.google.zxing.integration.android.IntentIntegrator;
import com.google.zxing.integration.android.IntentResult;
```

Para finalizar creamos un objeto de la clase IntentIntegrator pasándole una activity.

```
IntentIntegrator integrator = new IntentIntegrator(activity);
```

Configuramos las opciones que deseemos con los métodos set y finalmente procedemos a iniciar la actividad de escaneo.

```
integrator.initiateScan();
```

Para recoger los resultados debemos sobreescribir el método onActivityResult y utilizar el valor de los códigos como se desee.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
   IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
}
```

4.2. Cálculo de rutas

Para el cálculo de rutas se ha desarrollado el algoritmo de Dijkstra. Para ello se han tenido que añadir dos clases más a las establecidas en los diseños anteriores (Grafo.java y Nodo.java). Nodo implementa la interfaz *Comparable* que permitirá comparar nodos entre sí.

Para el cálculo del grafo que forman los distintos pasillos e intersecciones de los edificios se toman los datos almacenados en la tabla códigos QR, donde tenemos almacenados las uniones entre códigos así como sus distancias. Al ser pasillos, se toman las aristas como bidireccionales por lo que ambos nodos deberían almacenar sus conexiones. En este caso, solo se almacenará la conexión en un código QR y no en los dos que forman la arista supliendo esto a la hora de crear el grafo como se describirá posteriormente.

El grafo lo conforma una matriz de tamaño igual al número de códigos QR presentes en una planta de un edificio (en el caso de la planta baja del A3: 18x18). A la hora de marcar las uniones entre nodos se asignarán tanto en columnas como en filas para así generar la bidireccionalidad entre nodos.

Debido a que una ubicación se encontraba entre dos intersecciones (códigos QR) ha sido preciso calcular el algoritmo de Dijkstra dos veces para cada destino. Por ejemplo, si el usuario se encuentra en el código QR “A” y quiere llegar al despacho A3-067, los dos códigos QR destino serán “F” y “G”, por lo que el algoritmo se calculará para el par (A,F) y para el par (A,G). De entre ambas soluciones se escogerá la más corta que será la que se mostrará a través de Realidad Aumentada.

4.3. Realidad Aumentada

En un principio, en la etapa de estudio de tecnologías se consideró Vuforia como una de las alternativas al desarrollo de la Realidad Aumentada. En los ejemplos realizados se consiguió mostrar Realidad Aumentada con la herramienta Unity3D. Esta herramienta generaba un .apk que instalada en nuestro terminal reconocía el target que habíamos configurado y mostraba sobre él la flecha de direccionamiento. El problema consistía en poder integrar ese ejemplo con el código implementado en Java de la aplicación. Tras la lectura de documentación y observando que no se encontraba una solución que cumpliera con nuestras expectativas se procedió al estudio de otras alternativas.

Se empezó el estudio de otra biblioteca, Wikitude. Se encontró numeroso material en el que se podía observar aplicaciones realizadas en Android Studio que era algo más acorde al objetivo. Tras hacer una pequeña prueba con la biblioteca se observó que cumplía todos los aspectos necesarios para integrarla con el proyecto.

Para el uso de Wikitude de manera gratuita es necesario registrarse en la página web y se procede al envío de una clave para poder usar la biblioteca en nuestra aplicación con el único inconveniente de la muestra de una marca de agua. [REF4.3.1]

Otro de los pasos a realizar antes de usar la biblioteca es la generación de los target del proyecto. Para ello, Wikitude en su página web nos permite subir las imágenes que servirán de target (en nuestra aplicación los códigos QR) y generará un archivo extensión .wtc que usaremos en nuestro proyecto. [REF4.3.2]

Para utilizar la librería de Wikitude debemos realizar los siguientes pasos:

- Importarla en nuestro proyecto

- En el fichero build.gradle de nuestro módulo incluir:

```
compile (name: 'wikitudesdk', ext:'aar')
```

-Añadir en el archivo AndroidManifest.xml las siguientes líneas de código:

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

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

-Debemos crear en la estructura de ficheros, en la carpeta assets de nuestro proyecto los siguientes elementos:

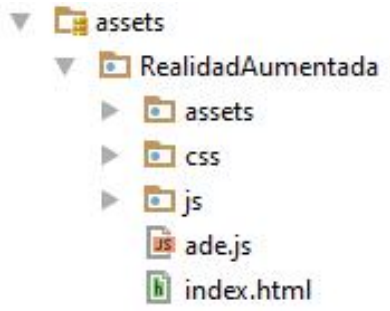


Ilustración 4.3. Imagen estructura carpeta assets.

En la carpeta assets almacenaremos el archivo .wtc generado desde la página web de Wikitude.

En la carpeta css guardaremos el archivo .css que se encargue del estilo de la página index.html

En la carpeta js se incluirá el archivo .js que será el encargado de asignar los target a las imágenes que queramos mostrar en formato javascript.

Por último, en las clases controladores de nuestro código debemos utilizarlo de la siguiente manera:

-Añadir los import en las clases:

```
import com.wikitude.architect.ArchitectView;
import com.wikitude.architect.StartupConfiguration;
```

-Creamos el architectView que será el encargado de generar la Realidad Aumentada en el index.html.

```
/*Asignamos la ruta del architectView en nuestro XML*/
this.architectView =
(ArchitectView)this.findViewById( R.id.architectView );
/*Añadimos nuestra autenticación para el uso de Wikitude*/
final StartupConfiguration config = new StartupConfiguration("
Autenticación");
this.architectView.onCreate( config );
```

-Sobreescribimos el método onPostCreate para cargar los datos en el index.html y en nuestro caso pasamos a través del metodo callJavaScript() los datos en formato JSON que se utilizarán para asignar a cada código QR (target) la flecha correspondiente. Para ello, llamamos a la función escrita en javascript cargaDatos.

```
/*Cargamos el html donde se proyectará la realidad aumentada*/
this.architectView.load("RealidadAumentada/index.html");
/*Cargamos la ruta y las instrucciones en el javascript que
controlará el html*/
PantallaMostrarRuta.this.callJavaScript("World.cargarDatos",
new String[] {instrucciones.toString()});
```

-En el fichero RealidadAumentada.js que se llamará cuando se cargue el index.html deberemos asociar el fichero .wtc y las imágenes obtenidas en formato JSON a través de la función callJavaScript().

```
/*Asignamos los códigosQR creados como targets*/
this.tracker = new AR.ClientTracker("assets/codigos.wtc", {
                    onLoaded: this.worldLoaded
                });


var imgA = new AR.ImageResource("assets/flecha6.png");
```

4.4. Base de datos

La implementación de la base de datos se realizó de acuerdo al patrón DAO. Para poder trabajar con la base de datos, Android Studio incorpora una biblioteca que permite realizar las operaciones propias del manejo de éstas. Para ello lo único que tenemos que hacer es incorporar una serie de líneas de código a nuestro proyecto.

-En el fichero build.gradle de nuestro módulo aplicación debemos añadir la siguiente línea:

```
compile 'com.readystatesoftware.sqliteasset:sqliteassethelper:+'
```

-Se creará una clase que será la encargada de la creación, copia y comprobación de la base de datos. Deberá extender de la interfaz SQLiteAssetHelper. En nuestro caso se denomina BBDDHelper.

-Almacenaremos en la carpeta assets de nuestro proyecto la base de datos creada previamente con la aplicación SQLiteDatabaseBrowserPortable.

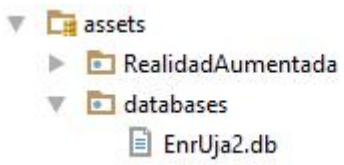


Ilustración 4.4. Imagen estructura carpeta assets/databases.

-Para obtener una instancia de la base de datos y poder consultar datos de la misma debemos añadir el siguiente código a nuestras clases:

```
BBDDhelper usdbh = new BBDDhelper(getApplicationContext());
final SQLiteDatabase db = usdbh.getWritableDatabase();
```

-Una vez obtenida la instancia de la base de datos, a través de un cursor podemos recorrer los datos obtenidos de una consulta SQL. En nuestro caso se ha hecho una consulta SQL parametrizada.

```
 /*Consulta parametrizada para evitar inyecciones de código SQL*/
c = db.query("profesor", campos, "correo like '%" + correo + "%'",
null, null, null, null);
if (c.moveToFirst()) {
   nuevo = new Profesor(c.getString(0), c.getString(1),
c.getString(2), "A3-" + c.getString(3), c.getString(4));
}
c.close();
```

5. Pruebas

En este apartado de la memoria se van a especificar las pruebas que se han realizado para comprobar el funcionamiento de cada unas de las funcionalidades de la aplicación y que de está manera cumpla los requisitos previstos.

El orden de presentación de pruebas no corresponde con el orden de realización de éstas. Se han ordenado en función de los requisitos funcionales que permiten comprobar.

5.1. Calcular ruta con menor distancia

Id.	Prueba realizada	Resultado	Solución
P6-1	Seleccionado despacho: A3-040 Inicio: “F”	Correcto. La aplicación muestra la ruta F G H I J	
P6-2	Seleccionado despacho: A3-014 Inicio: “P”	Correcto. Muestra como camino más corto P Q G F y como segundo camino P M N C D E	
P6-3	Seleccionado despacho: A3-067 Inicio: “A”	Correcto. Muestra el camino A M P Q G con una distancia de 38 metros. Segundo camino A M N C D E F con 65 metros.	
P6-4	Seleccionado despacho: A3-031 Inicio: “K”	Correcto. Muestra el camino K J I H con 28 metros.Segundo camino K O Q G con 39 metros.	
P6-5	Seleccionado despacho: A3-023 Inicio: “C”	Correcto. Muestra el camino C D E F con 28 metros. Segundo camino C N Q G con 39 metros.	
P6-6	Seleccionado despacho: A3-074 Inicio: “H”	Correcto. Muestra el camino H G Q N C con 70 metros. Segundo camino H Q O N C B con 88 metros	
P6-7	Seleccionado despacho: A3-009	Correcto. Muestra el camino M N C D con 39 metros. Segundo camino M N C D E	

	Inicio: “M”	con 45 metros.	

Tabla 5.1. Pruebas sobre el cálculo de rutas.

Las pruebas realizadas permiten comprobar que todos los códigos QR están conectados con sus respectivos nodos y que las distancias son correctas. La selección de objetivos e inicios permite observar el cálculo de las rutas a ambos extremos del destino.

5.2. Mostrar ruta con Realidad Aumentada

Id.	Prueba realizada	Resultado	Solución
P2-1	Seleccionado despacho: A3-028 Inicio: “O”	Error. La flecha correspondiente al código “H” muestra la dirección sin indicar la proximidad del usuario a su destino.	El problema se encuentra en la elección de ruta. Al escoger la segunda ruta como la más cercana, asigna los extremos mal.
P2-1-1	Seleccionado despacho: A3-028 Inicio: “O”	Correcto. La aplicación muestra el camino correcto hasta el destino.	
P2-2	Seleccionado despacho: A3-009 Inicio: “P”	Error. La aplicación muestra mensaje de error de destino no alcanzable.	No se habían añadido correctamente los datos en la base de datos, dejando el nodo “P” sin uniones con sus otros dos nodos.
P2-2-1	Seleccionado despacho: A3-009 Inicio: “P”	Correcto. La aplicación muestra el camino correcto hasta el destino.	
P2-3	Seleccionado despacho: A3-073 Inicio: “G”	Correcto. La aplicación muestra el camino correcto hasta el destino.	
P2-4	Seleccionado despacho: A3-040 Inicio: “D”	Correcto. La aplicación muestra el camino correcto hasta el destino.	

P2-5	Seleccionado despacho: A3-014 Inicio: “B”	Correcto. La aplicación muestra el camino correcto hasta el destino.	

Tabla 5.2. Pruebas de Realidad Aumentada. Mostrar imagen.

Las pruebas realizadas en este apartado son complementarias a las realizadas en la tabla 5.1, debido a que es posible comprobar también la secuencia de nodos a visitar hasta el objetivo. Se han escogido nodos de inicio diferentes para poder comprobar todo el conjunto de códigos QR.

5.3. Escanear código QR

Id.	Prueba realizada	Resultado	Solución
P1-1	Lectura de todos los códigos QR tras pulsar botón de escaneado.	Correcto. La aplicación permite escanear los códigos QR que forman el grafo del edificio.	
P1-2	Lectura de código QR ajeno a los correspondientes al grafo del edificio.	Error. La aplicación deja de funcionar al obtener una lectura de un código no esperado.	Se comprueba la lectura del código QR, si el valor es válido se continua con el programa, sino se solicita volver a escanear otro código QR.
P1-2-1	Lectura de código QR ajeno a los correspondientes al grafo del edificio	Correcto. La aplicación solicita volver a escanear un código QR válido.	

Tabla 5.3. Pruebas escanear código QR.

Estas pruebas se han realizado abarcando todos los códigos QR que conforman la planta primera del edificio A3. Una vez comprobado que todos fueran escaneados correctamente se ha pasado al caso contrario, escanear algún código QR que no se encontrara en el dominio de la aplicación. De esta manera, con las pruebas realizadas se completa el abanico de posibilidades a introducir por el usuario.

5.4. Seleccionar tipo destino

Id.	Prueba realizada	Resultado	Solución
P3-1	Seleccionar aulas	Correcto. La aplicación muestra un mensaje informativo	
P3-2	Seleccionar edificios	Correcto. La aplicación muestra un mensaje informativo	
P3-3	Seleccionar laboratorios	Correcto. La aplicación muestra un mensaje informativo	
P3-4	Seleccionar profesorado	Correcto. La aplicación muestra la pantalla Profesorado.	
P3-5	Seleccionar servicios	Correcto. La aplicación muestra la pantalla Servicios.	

Tabla 5.4. Pruebas seleccionar tipo destino.

Los casos de prueba escogidos corresponden con las cinco posibles opciones de elección en la pantalla que permite seleccionar el tipo de destino.

5.5. Mostrar profesorado

Id.	Prueba realizada	Resultado	Solución
P4-1	Seleccionar profesorado	Error. La aplicación muestra el profesorado pero por duplicado.	La aplicación realizaba una doble búsqueda del profesorado y la mostraba ambas en el listado.
P4-2	Seleccionar profesorado	Correcto. La aplicación muestra todo el profesorado con su nombre y apellidos y el despacho correspondiente	

Tabla 5.5. Pruebas de mostrar profesorado.

Se ha realizado una prueba al acceder a la sección de profesorado que ha dado error. Tras solucionar el problema el resto de veces seleccionado el resultado ha sido correcto.

5.6. Buscar profesor por criterio

Id.	Prueba realizada	Resultado	Solución
P5-1	Texto de búsqueda: algarcia	Error. La aplicación muestra el profesor duplicado.	Se realizaba la búsqueda en dos lugares y rellenaba el vector de profesores dos veces.
P5-2	Texto de búsqueda: algarcia	Correcto. La aplicación muestra el profesor.	
P5-3	Texto de búsqueda: Ángel	Correcto. La aplicación muestra los profesores que cumplen el criterio.	
P5-4	Texto de búsqueda: lor	Correcto. La aplicación muestra los profesores que cumplen el criterio.	
P5-5	Texto de búsqueda: asdga	Correcto. La aplicación muestra un mensaje diciendo que no existen profesores con ese criterio.	
P5-6	Texto de búsqueda: ale	Correcto. La aplicación muestra los profesores que cumplen el criterio.	

Tabla 5.6. Pruebas búsqueda profesor por criterio.

Con las seis pruebas realizadas se completa el abanico de posibilidades disponibles en la búsqueda de un profesor. Búsqueda por correo, búsqueda por nombre o apellido, búsqueda por una subcadena y búsqueda con resultado negativo.

5.7. Mostrar ficha profesorado

Id.	Prueba realizada	Resultado	Solución
P7-1	Seleccionado: Antonio Cano Ortega	Correcto. Muestra nombre, apellidos y despacho. No dispone de imagen.	
P7-2	Seleccionado: Manuel José Barranco García	Correcto. Muestra nombre, apellidos y despacho. Imagen disponible.	
P7-3	Seleccionado: José Ramón Balsas Almagro	Correcto. Muestra nombre, apellidos y despacho. Imagen disponible.	
P7-4	Seleccionado: Francisco Javier Cardenal Escarcena	Correcto. Muestra nombre, apellidos y despacho. Imagen disponible.	

Tabla 5.7. Pruebas mostrar ficha profesorado.

Se han realizado cuatro pruebas para comprobar la situación de la ficha, el correcto mostrado de los datos y las alternativas de profesores con imagen disponible y profesores sin imagen (descarga de la imagen comprobada con este hecho).

6. Métricas

Para este aparcado se ha trabajado con la biblioteca MetricsReloaded.

Metrics Reloaded es un plugin que nos proporciona una gran cantidad de métricas sobre nuestro proyecto. Desde la complejidad ciclomática a la cohesión entre clases, lo que realmente nos ofrece una gran visión de la estructura de nuestro proyecto. El código fuente está disponible en GitHub. [REF6.1] [REF6.2]

Las métricas se pueden ejecutar para un proyecto completo, un módulo específico, el archivo actual... por lo que es extremadamente flexible en cuanto a qué es lo que queremos analizar. Incluso nos proporciona ala capacidad de filtrar nuestras pruebas.

Algunas de las métricas disponibles son:

-Chidamber-Kmerer. Nos muestra información sobre acoplamiento de objetos, profundidad, herencia, cohesión entre métodos (o falta de ella), número de subclases...

-Dependencias. Proporciona información acerca de las dependencias cíclicas y dependencias transitivas.

-Complejidad ciclomática.

-Cobertura Javadoc.

-Líneas de código

6.1. Métricas de complejidad

Tras pasar la métrica de complejidad por el código obtenemos los resultados que se pueden observar en la Ilustración 6.1.1.

Method metrics | Class metrics | Package metrics | Module metrics | Project metrics

class	OCavg	WMC
com.tfg.ahurt.enruja2.Operaciones	14,50	58
com.tfg.ahurt.enruja2.BBDD.CodigoQRDAO	3,00	9
com.tfg.ahurt.enruja2.Grafo	2,86	20
com.tfg.ahurt.enruja2.BBDD.ProfesorDAO	2,50	15
com.tfg.ahurt.enruja2.pantalla.PantallaMostrarRuta	2,25	18
com.tfg.ahurt.enruja2.BBDD.UbicacionDAO	2,25	9
com.tfg.ahurt.enruja2.BBDD.BBDDhelper	2,00	8
com.tfg.ahurt.enruja2.pantalla.PantallaPreEscaneado	2,00	14
com.tfg.ahurt.enruja2.pantalla.PantallaProfesorado	1,70	17
com.tfg.ahurt.enruja2.Nodo	1,25	5
com.tfg.ahurt.enruja2.Lista_Adaptador	1,20	6
com.tfg.ahurt.enruja2.pantalla.PantallaServicios	1,00	2
com.tfg.ahurt.enruja2.pantalla.PantallaSeleccion	1,00	8
com.tfg.ahurt.enruja2.objetosDTO.CodigoQR	1,00	5
com.tfg.ahurt.enruja2.objetosDTO.Servicio	1,00	8
com.tfg.ahurt.enruja2.ApplicationTest	1,00	1
com.tfg.ahurt.enruja2.objetosDTO.Union	1,00	7
com.tfg.ahurt.enruja2.objetosDTO.Profesor	1,00	9
com.tfg.ahurt.enruja2.pantalla.MainActivity	1,00	6
com.tfg.ahurt.enruja2.ExampleUnitTest	1,00	1
com.tfg.ahurt.enruja2.objetosDTO.Ubicacion	1,00	9
com.tfg.ahurt.enruja2.pantalla.PantallaPreEscaneado.Ca	1,00	3
Total		**238**
Average	1,97	10,82

Ilustración 6.1.1. Resultados métricas de compejidad.

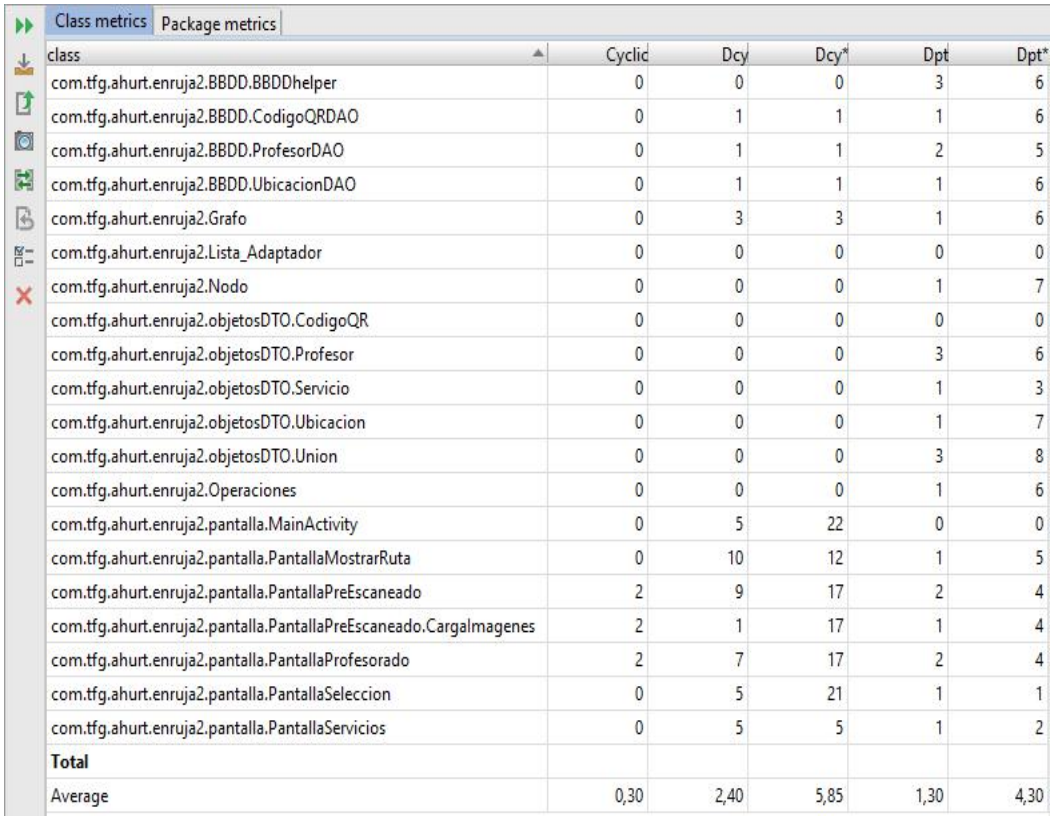
Para comprender esta métrica debemos conocer el significado de los valores de WMC (Weighted Methods per Class). WMC devuelve como resultado la suma de la complejidad de cada método.

Por lo que podemos ver en la Ilustración 6.1.1. la complejidad en la clase Operaciones de nuestra aplicación es muy superior al nivel máximo establecido en el plugin. Esto es debido a que la clase Operaciones se encarga de la asignación de las flechas adecuadas a cada código QR para realizar la Realidad Aumentada y existe un gran número elevado de opciones que incrementan la complejidad del método.

El resto de clases muestras una complejidad de sus clases y métodos óptima.

6.2. Métricas de dependencia

Las métricas de dependencia proporcionan información acerca de las dependencias cíclicas y transitivas de clases, así como las dependencias en cada clase.



Class metrics | Package metrics

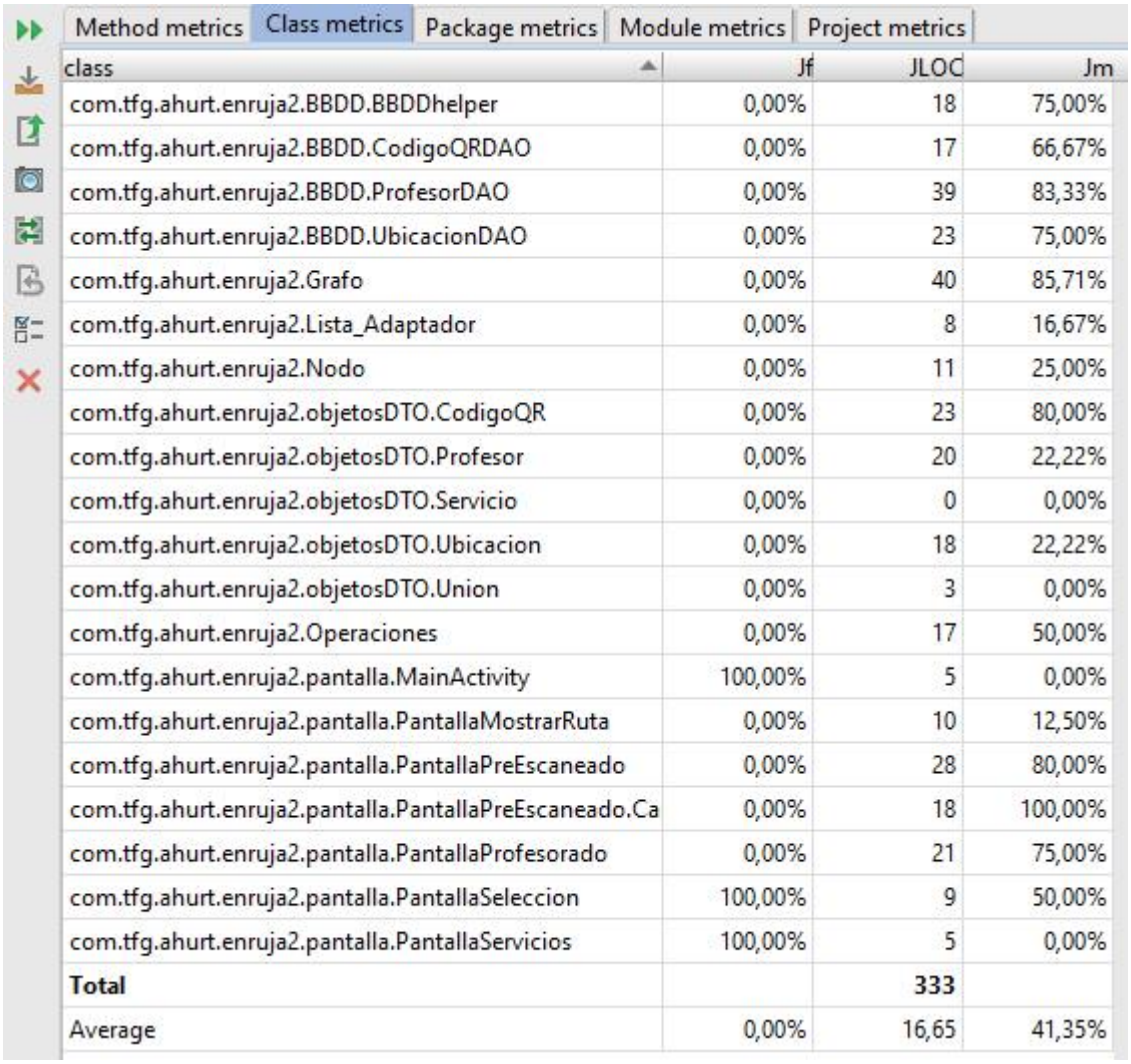
class	Cyclic	Dcy	Dcy*	Dpt	Dpt*
com.tfg.ahurt.enruja2.BBDD.BBDDhelper	0	0	0	3	6
com.tfg.ahurt.enruja2.BBDD.CodigoQRDAO	0	1	1	1	6
com.tfg.ahurt.enruja2.BBDD.ProfesorDAO	0	1	1	2	5
com.tfg.ahurt.enruja2.BBDD.UbicacionDAO	0	1	1	1	6
com.tfg.ahurt.enruja2.Grafo	0	3	3	1	6
com.tfg.ahurt.enruja2.Lista_Adaptador	0	0	0	0	0
com.tfg.ahurt.enruja2.Nodo	0	0	0	1	7
com.tfg.ahurt.enruja2.objetosDTO.CodigoQR	0	0	0	0	0
com.tfg.ahurt.enruja2.objetosDTO.Profesor	0	0	0	3	6
com.tfg.ahurt.enruja2.objetosDTO.Servicio	0	0	0	1	3
com.tfg.ahurt.enruja2.objetosDTO.Ubicacion	0	0	0	1	7
com.tfg.ahurt.enruja2.objetosDTO.Union	0	0	0	3	8
com.tfg.ahurt.enruja2.Operaciones	0	0	0	1	6
com.tfg.ahurt.enruja2.pantalla.MainActivity	0	5	22	0	0
com.tfg.ahurt.enruja2.pantalla.PantallaMostrarRuta	0	10	12	1	5
com.tfg.ahurt.enruja2.pantalla.PantallaPreEscaneado	2	9	17	2	4
com.tfg.ahurt.enruja2.pantalla.PantallaPreEscaneado.CargaImagenes	2	1	17	1	4
com.tfg.ahurt.enruja2.pantalla.PantallaProfesorado	2	7	17	2	4
com.tfg.ahurt.enruja2.pantalla.PantallaSeleccion	0	5	21	1	1
com.tfg.ahurt.enruja2.pantalla.PantallaServicios	0	5	5	1	2
Total					
Average	0,30	2,40	5,85	1,30	4,30

Ilustración 6.2.1. Resultados métrica de dependencias.

Tras pasar el código por la prueba no se muestran valores anómalos. Los valores de dependencias de todas las clases se muestran por debajo del umbral máximo.

6.3. Recuento de lineas Javadoc

Esta métrica nos permite comprobar el porcentaje de líneas Javadoc de cada clase en nuestra aplicación. Es muy útil para ver el nivel de documentación de código realizado.



Method metrics | Class metrics | Package metrics | Module metrics | Project metrics

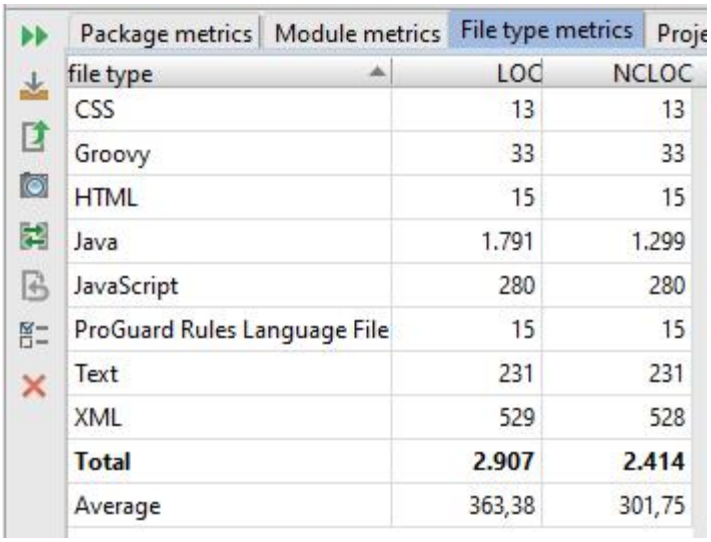
class	Jf	JLOC	Jm
com.tfg.ahurt.enruja2.BBDD.BBDDhelper	0,00%	18	75,00%
com.tfg.ahurt.enruja2.BBDD.CodigoQRDAO	0,00%	17	66,67%
com.tfg.ahurt.enruja2.BBDD.ProfesorDAO	0,00%	39	83,33%
com.tfg.ahurt.enruja2.BBDD.UbicacionDAO	0,00%	23	75,00%
com.tfg.ahurt.enruja2.Grafo	0,00%	40	85,71%
com.tfg.ahurt.enruja2.Lista_Adaptador	0,00%	8	16,67%
com.tfg.ahurt.enruja2.Nodo	0,00%	11	25,00%
com.tfg.ahurt.enruja2.objetosDTO.CodigoQR	0,00%	23	80,00%
com.tfg.ahurt.enruja2.objetosDTO.Profesor	0,00%	20	22,22%
com.tfg.ahurt.enruja2.objetosDTO.Servicio	0,00%	0	0,00%
com.tfg.ahurt.enruja2.objetosDTO.Ubicacion	0,00%	18	22,22%
com.tfg.ahurt.enruja2.objetosDTO.Union	0,00%	3	0,00%
com.tfg.ahurt.enruja2.Operaciones	0,00%	17	50,00%
com.tfg.ahurt.enruja2.pantalla.MainActivity	100,00%	5	0,00%
com.tfg.ahurt.enruja2.pantalla.PantallaMostrarRuta	0,00%	10	12,50%
com.tfg.ahurt.enruja2.pantalla.PantallaPreEscaneado	0,00%	28	80,00%
com.tfg.ahurt.enruja2.pantalla.PantallaPreEscaneado.Ca	0,00%	18	100,00%
com.tfg.ahurt.enruja2.pantalla.PantallaProfesorado	0,00%	21	75,00%
com.tfg.ahurt.enruja2.pantalla.PantallaSeleccion	100,00%	9	50,00%
com.tfg.ahurt.enruja2.pantalla.PantallaServicios	100,00%	5	0,00%
Total		**333**	
Average	0,00%	16,65	41,35%

Ilustración 6.3.1. Resultados de métrica de recuento de líneas Javadoc.

Como podemos observar en la Ilustración 6.3.1. la media de líneas Javadoc por clase es de 16,65. Esta métrica realiza el recuento únicamente de los comentarios en dicho formato, por lo que los comentarios que no siguen el fomato Javadoc no los tiene en cuenta.

6.4. Métrica de líneas de código

Esta métrica ofrece un recuento total de líneas de código. Como se puede observar en la Ilustración 6.4.1. muestra el total de líneas de código así como el tipo de lenguaje utilizado.

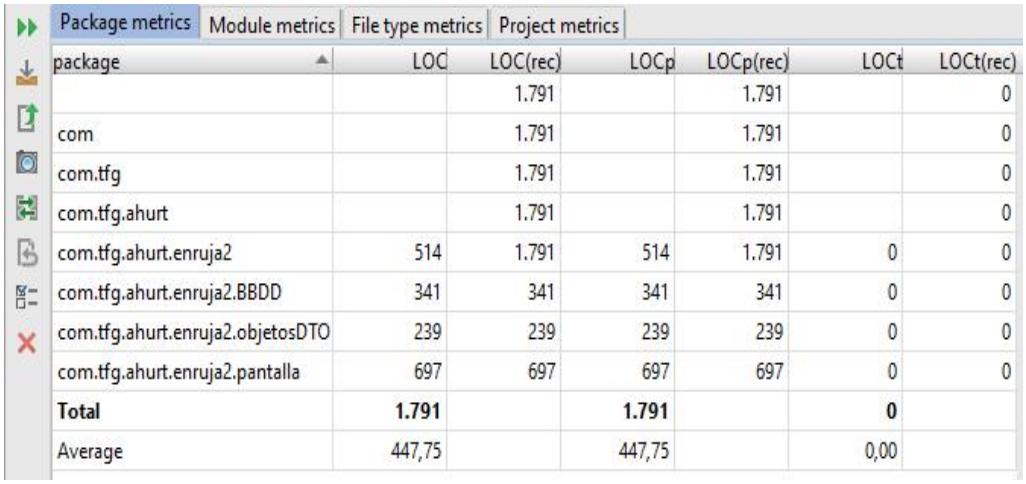


Package metrics | Module metrics | File type metrics | Proje

file type	LOC	NCLOC
CSS	13	13
Groovy	33	33
HTML	15	15
Java	1.791	1.299
JavaScript	280	280
ProGuard Rules Language File	15	15
Text	231	231
XML	529	528
Total	**2.907**	**2.414**
Average	363,38	301,75

Ilustración 6.4.1. Resultados métrica de líneas de código.

También es posible mostrar las líneas de código por paquetes como se muestra en la Ilustración 6.4.2.



Package metrics | Module metrics | File type metrics | Project metrics

package	LOC	LOC(rec)	LOCp	LOCp(rec)	LOCt	LOCt(rec)
		1.791		1.791		0
com		1.791		1.791		0
com.tfg		1.791		1.791		0
com.tfg.ahurt		1.791		1.791		0
com.tfg.ahurt.enruja2	514	1.791	514	1.791	0	0
com.tfg.ahurt.enruja2.BBDD	341	341	341	341	0	0
com.tfg.ahurt.enruja2.objetosDTO	239	239	239	239	0	0
com.tfg.ahurt.enruja2.pantalla	697	697	697	697	0	0
Total	**1.791**		**1.791**		**0**	
Average	447,75		447,75		0,00	

Ilustración 6.4.2. Resultados métrica de líneas de código por paquetes.

7. Conclusiones y trabajo futuro

El objetivo principal de este trabajo era la realización de una aplicación que permitiera guiarnos a través del campus de la Universidad de Jaén utilizando realidad aumentada. Se han conseguido todos los objetivos marcados al inicio de este proyecto y se han dejado sentadas las bases para la ampliación y mejora de otras funcionalidades.

Se realizó un análisis profundo para permitir que el proyecto pudiera avanzar en un futuro y no fuera una mera resolución de un problema. Es por ello que partiendo de esta base, se pueden desarrollar algunas mejoras sustanciales:

- Ampliación del dominio de la aplicación. Se pueden incluir otras plantas del edificio A3, así como otros edificios del campus.

-Uso de posicionamiento GPS. Se puede implementar la búsqueda de rutas entre edificios, donde el usuario pueda ser localizado a través de GPS.

-Búsquedas a través de aulas, laboratorios, instalaciones... Aumentar la funcionalidad de la aplicación desarrollando los distintos tipos de destino.

-Uso de imágenes 3D para el direccionamiento entre puntos. La biblioteca Wikitude nos permite realizar esta mejora así como otras relacionadas con el reconocimientos de puntos, puntos de interés...

-Diseño de interfaz mejorado con elementos interactivos.

Otro de los puntos en los que se ha hecho especial hincapié es en la documentación. Como todo software, desde el momento en que termina su desarrollo comienza la etapa quizás más dura de este proceso, el mantenimiento. Se ha intentado que todas las decisiones de diseño estén perfectamente justificadas y que el código implementado se encuentre correctamente documentado, con el fin de poder facilitar su mejora y entendimiento a futuros desarrolladores.

Dejando atrás el resultado obtenido a través de la aplicación, se han conseguido también otros hitos que se plantearon al inicio de éste.

-Conocimiento sobre nuevas técnicas y herramientas.

Uno de los objetivos personales en este proyecto era la adquisición de conocimientos que no hubiesen sido tratados durante mi periodo universitario, por lo que la oportunidad de trabajar en el desarrollo de una aplicación Android me parecía muy interesante. Entorno de desarrollo nuevo, diferente manera de trabajar, conceptos con los que familiarizarse (xml, sdk, actividades, emuladores...), pequeñas cosas que parecen un mundo al principio.

La realidad aumentada era para mí un terreno desconocido, del cual he aprendido cosas muy interesantes. El resultado de la aplicación no es más que un pequeño y sencillo ejemplo de la cantidad de cosas que se pueden realizar con ella. Es una de esas tecnologías, que una vez que la usas, te incita a realizar grandes proyectos.

-Capacidad de resolución y aprendizaje.

Durante nuestro periodo en la Universidad los ejemplos prácticos que desarrollamos suelen ser muy guiados y establecidos por los profesores. La realización de este proyecto permitía profundizar con las herramientas, buscar alternativas y ser muy selectivo a la hora de buscar información que resultará útil para el desarrollo de la aplicación.

He aprendido que una mala elección puede dar muchos quebraderos de cabeza y que una buena documentación de un proyecto, una biblioteca o una simple función, puede ser la clave entre usar (comprar en el mejor de los casos), o no, tu desarrollo.

Como conclusión final, mostrar mi satisfacción personal por haber finalizado un proyecto comenzado desde cero y haber podido demostrar los muchos conocimientos adquiridos durante estos años.

8. Apéndice

8.1. Manual de usuario

Este manual le permitirá aprender a utilizar todas las funcionalidades básicas de EnrUja2.

Ilustración 8.1.1. Pantalla principal

En la ilustración 8.1.1. se puede observar la pantalla principal de la aplicación, en ella se hace una pequeña introducción a la funcionalidad de ésta. En la parte inferior de la pantalla se muestra un botón que al ser pulsado nos permite dirigirnos a la pantalla de selección de ubicaciones.

Ilustración 8.1.2. Pantalla Selección tipo ruta.

La ilustración 8.1.2. muestra los distintos tipos de destinos entre los que se puede elegir. Esta pantalla está compuesta por cinco botones, uno por cada tipo de destino disponible. Los botones correspondientes a Aulas, Edificios y Laboratorios, al ser pulsados muestran un mensaje informando al usuario del no desarrollo de estas opciones por el momento (Ilustración 8.1.3). Sin embargo, si pulsamos sobre los botones de Profesorado o Servicios seremos dirigidos hacia la pantalla de Profesores (Ilustración 8.1.5) o a la pantalla de Servicios (Ilustración 8.1.4) respectivamente.

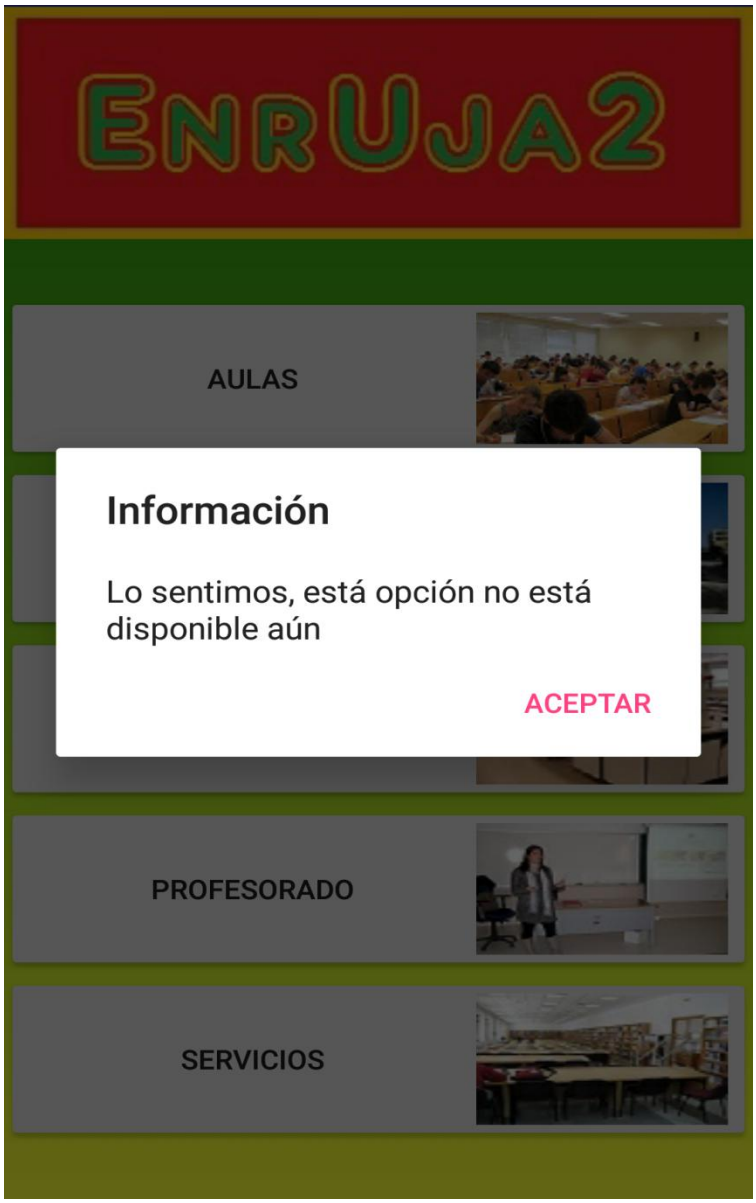


Ilustración 8.1.3. Mensaje informativo.

La ilustración 8.1.3. muestra el mensaje informativo al seleccionar alguna opción que aún no ha sido desarrollada. Si pulsamos sobre el texto Aceptar volveremos a la pantalla donde nos encontrabamos anteriormente.

Ilustración 8.1.4. Pantalla servicios.

La ilustración 8.1.4. muestra la información acerca de los servicios ofrecidos en la Universidad de Jaén. No dispone de ninguna funcionalidad adicional.

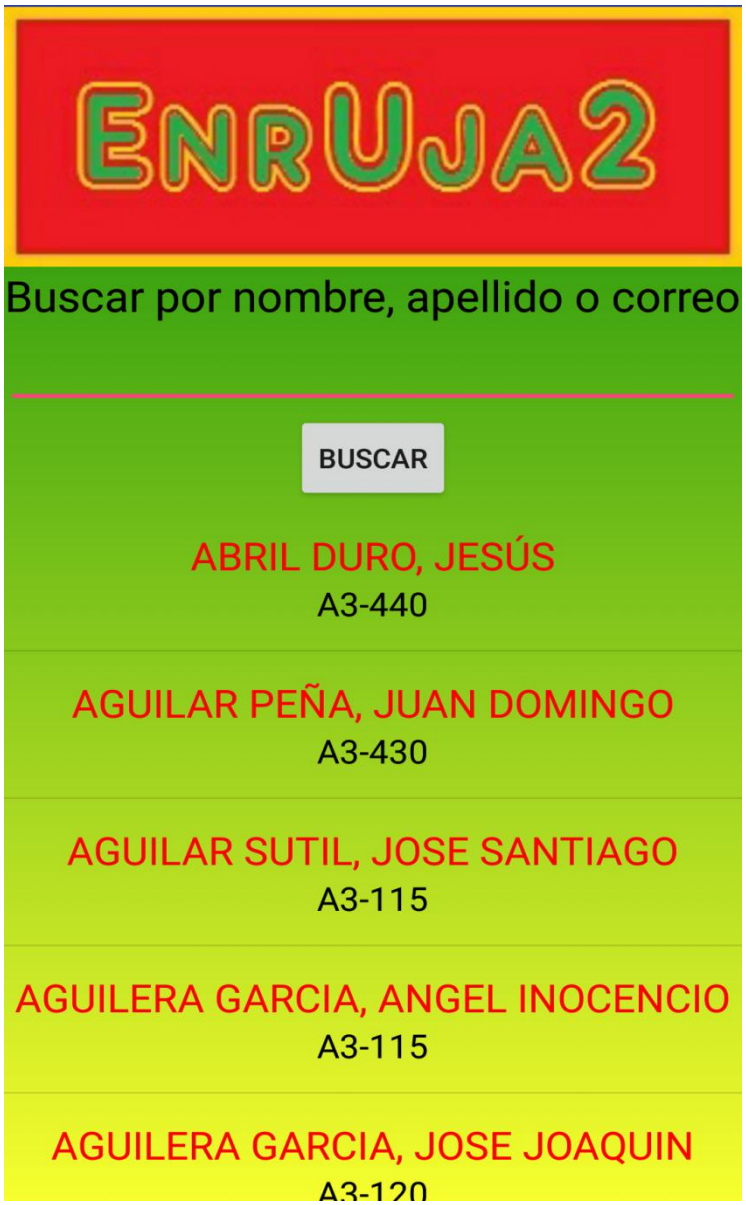


Ilustración 8.1.5. Pantalla profesorado.

En la pantalla profesorado podemos encontrar un listado con todos los profesores con despacho en el edificio A3 del campus Las Lagunillas. En la parte superior de la pantalla encontramos una caja de texto donde se puede introducir el nombre, apellido o correo de un profesor para realizar su búsqueda. El buscador tambien permite introducir parte del nombre o los apellidos y la búsqueda se realizará a partir de esa subcadena. Una vez introducido el texto si pinchamos sobre el botón Buscar aparecerán los resultados (Ilustración 8.1.6). En el caso que el campo de texto se encontrase vacío aparecerá todo el profesorado (Ilustración 8.1.5).

Una vez encontrado el profesor deseado se puede pinchar sobre él para pasar a la confirmación de profesor (Ilustración 8.1.8).

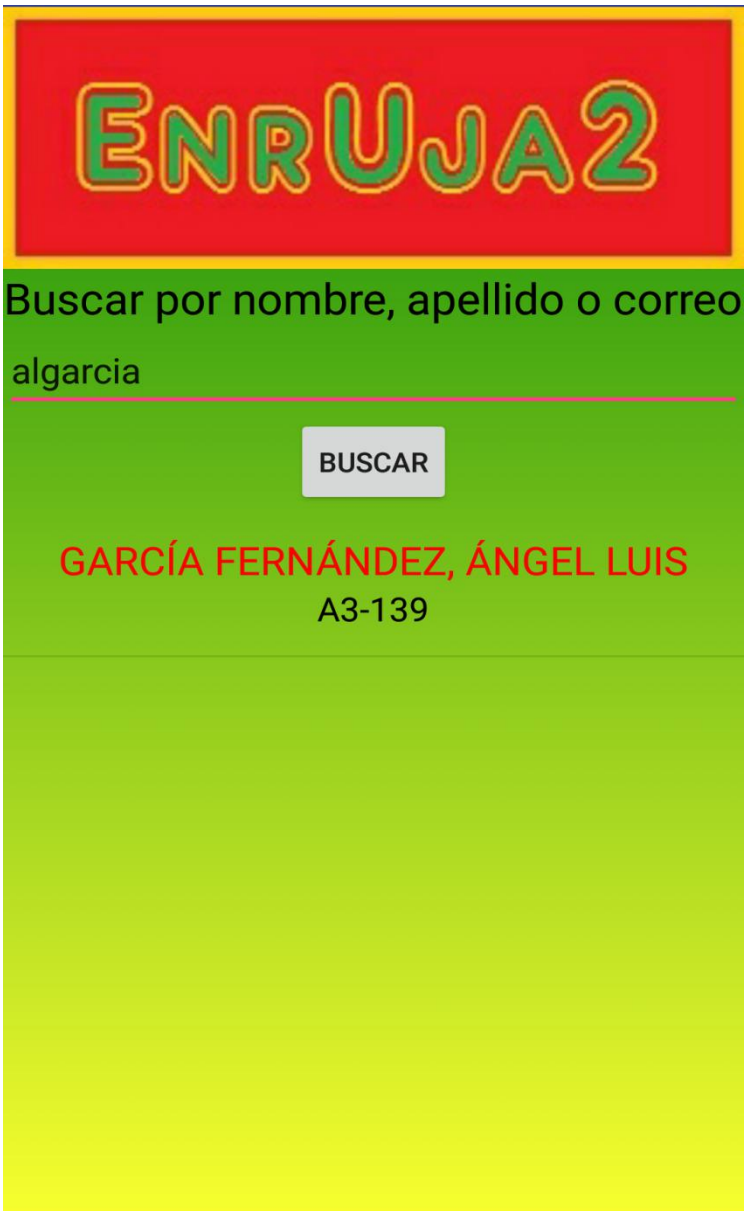


Ilustración 8.1.6. Mostrar profesor, búsqueda por correo.

La ilustración 8.1.6. muestra el profesor con el correo indicando en la caja de texto. No es necesario especificar si la búsqueda se realiza por correo, por nombre o por apellidos. La aplicación realizará la búsqueda con todos los criterios y devolverá los resultados obtenidos.

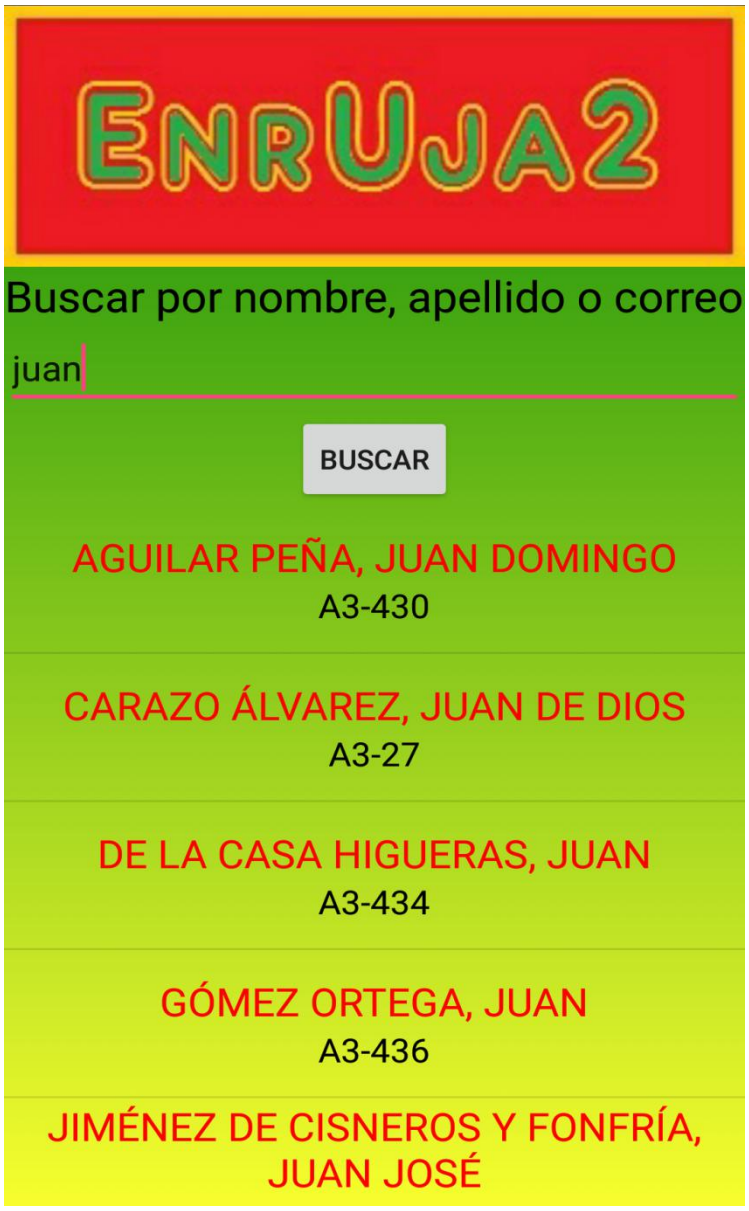


Ilustración 8.1.7. Mostrar profesor, todos los criterios.

En la ilustración 8.1.7. podemos ver como se muestran todos los profesores en cuyo nombre, apellidos o correo se encuentra la palabra “juan”.

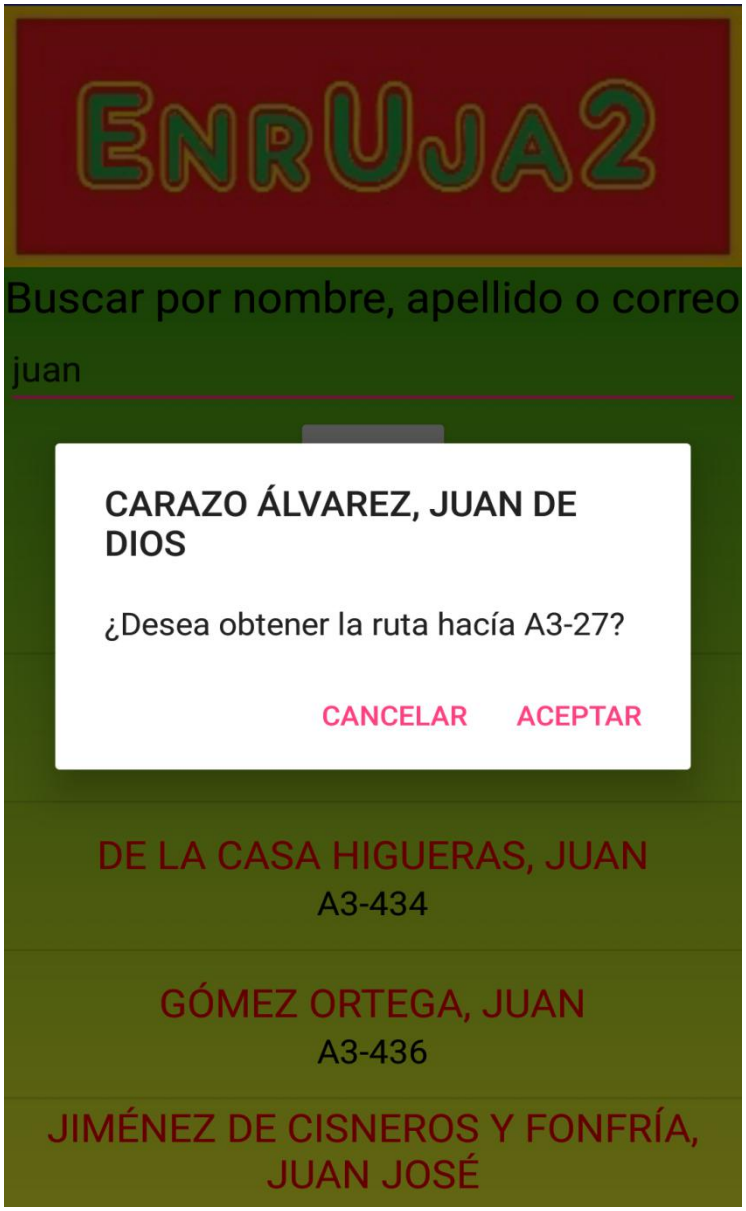


Ilustración 8.1.8. Mensaje de confirmación. Profesor elegido.

En la Ilustración 8.1.8. aparece un mensaje informativo tras haber seleccionado un profesor. En el mensaje aparecen los datos del profesor solicitando si es el despacho elegido. Tenemos la opción de pulsar Aceptar que mostraría la pantalla de preescaneado (Ilustracion 8.1.9) o pulsar la opción Cancelar, volviendo a la pantalla de búsqueda de profesores (Ilustración 8.1.5).

Ilustración 8.1.9. Pantalla preescaneado.

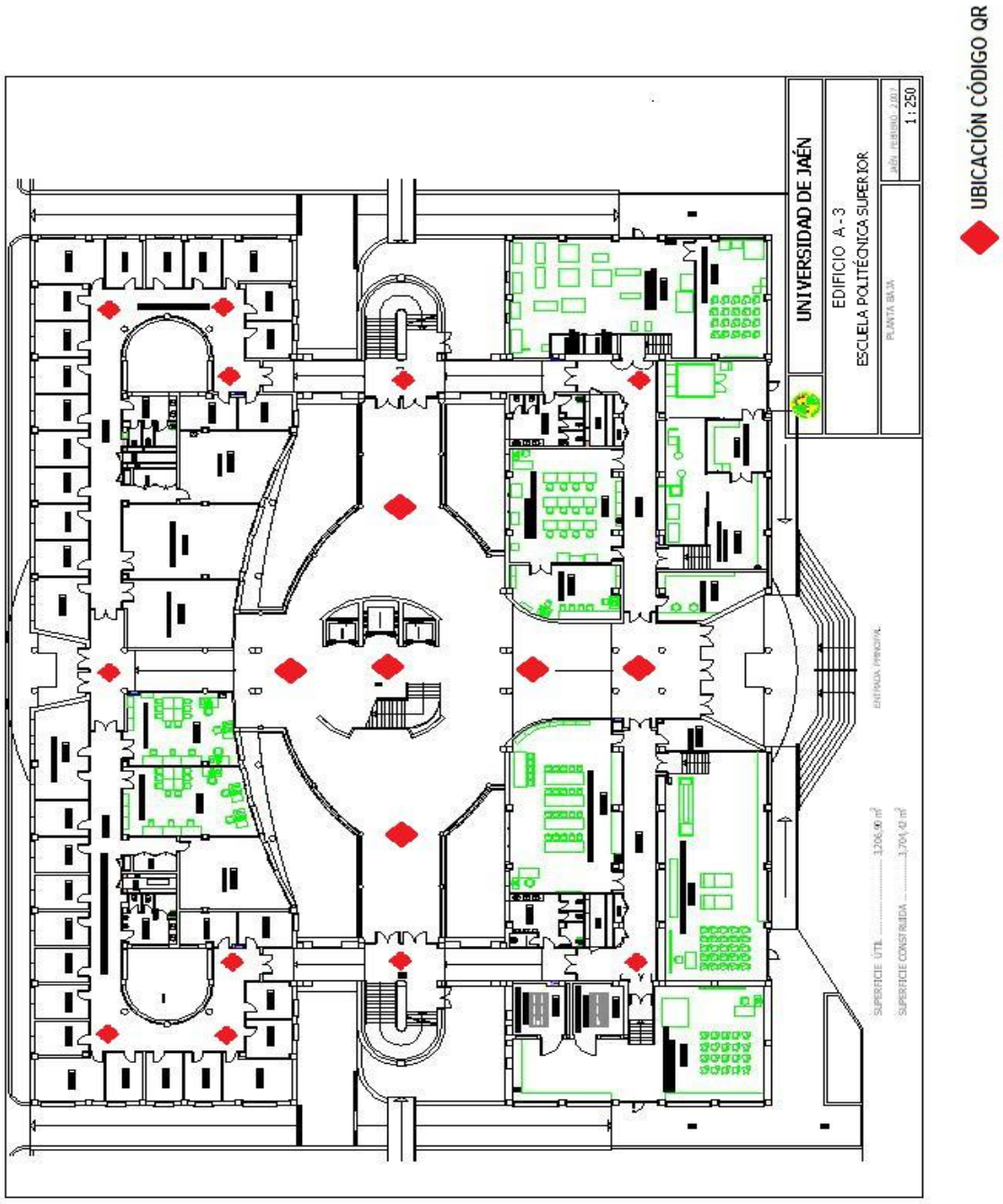
La Illustración 8.1.9. muestra la pantalla preescaneado. En ella se muestra la ficha completa del profesor seleccionado (nombre, apellidos y despacho) acompañado de una imagen del profesor (si está disponible). La opción ruta normal estará marcada por defecto, mientrás que la opción “Ruta sin barreras arquitectónicas” no será seleccionable. En la parte inferior aparece el botón “Escanear código QR” que al pulsarlo abrirá la cámara de nuestro terminal para escanear el código QR que marcará donde nos encontramos.

Ilustración 8.1.10. Mostrar Realidad Aumentada.

En la Ilustración 8.1.10. podemos observar la pantalla mostrar ruta. En ella, una vez escaneado el código QR que tengamos más cercano, nos mostrará la dirección hacía donde debemos dirigirnos para llegar a nuestro objetivo. Lo único que tenemos que hacer es continuar con nuestra pantalla abierta buscando el próximo código QR al que nos lleve la flecha. Cuando el fondo de la flecha se vuelva rojo indicará que el destino está próximo y que se encuentra antes de llegar al próximo código QR (Ilustración 8.1.11).

Ilustración 8.1.11. Realidad Aumentada. Objetivo cerca.

8.2. Mapa de la distribución de códigos QR



Bibliografía

Todas las referencias incluidas se encuentran disponibles a fecha 02/09/2016.

[REF1.2.1]

www.theappdate.es/blog/informe-sobre-las-apps-en-espana-2015-la-era-appcommerce/

[REF1.2.2]

es.wikipedia.org/wiki/Realidad_aumentada

[REF1.2.3]

www.durangomas.mx/2016/07/guia-para-entender-por-que-todo-el-mundo-se-ha-vuelto-loco-con-pokemon-go/

[REF1.2.4]

http://cincodias.com/cincodias/2016/07/11/lifestyle/1468235688_941889.html

[REF1.2.5]

http://www.elmundo.es/economia/2016/07/12/5784e1dde2704ed0578b45ce.html

[REF2.2.1.1]

http://www.expansion.com/economia-digital/companias/2016/06/15/5761265522601db42e8b4577.html

[REF2.2.1.2]

https://es.wikipedia.org/wiki/Android

[REF2.2.1.3]

https://developer.android.com/about/dashboards/index.html

[REF2.2.3.1]

https://es.wikipedia.org/wiki/Sistema_de_posicionamiento_global

[REF2.2.3.2.1]

https://situm.es/

[REF2.2.3.2.2]

https://github.com/situmtech/situm-android-getting-started

[REF2.2.3.2.3]

http://www.elmundo.es/economia/2015/01/22/54bff268268e3efa718b4583.html

[REF2.2.3.3.1]

https://es.wikipedia.org/wiki/Código_QR

[REF2.2.3.3.2]

https://github.com/zxing/zxing

[REF2.2.3.3.3]

http://zomwi.blogspot.com.es/2012/09/zxing.html

[REF2.2.4]

https://es.wikipedia.org/wiki/Realidad_aumentada#Cronolog.C3.ADa

[REF2.2.4.1]

https://es.wikipedia.org/wiki/Realidad_aumentada#Tecnolog.C3.ADa

[REF2.2.4.2]

https://es.wikipedia.org/wiki/Realidad_aumentada#Niveles

[REF2.2.4.3.1]

http://www.vuforia.com/

[REF2.2.4.3.2]

https://unity3d.com/es

[REF2.2.4.3.3]

https://www.wikitude.com/external/doc/documentation/5.1/android/samples.html

[REF2.2.4.3.4]

http://www.desarrollolibre.net/blog/tema/48/android/desarrollando-aplicaciones-de-realidad-aumentada-con-wikitude-parte-1#.Vqs5k5rhBdg

[REF2.3.3.1]

http://www.ccooservicios.es/archivos/capgemini/20090404_XVI_ConvenioColectivoTIC.pdf

[REF2.3.3.2]

http://blogs.salleurl.edu/ingenieria/%C2%BFcuanto-cobra-un-ingeniero-it-en-espana/

[REF2.3.4.1]

http://eshop.asus.com/es-ES/asus-f556ua-eur-es-es-90nb09s2-m00850.html

[REF2.3.4.2]

https://www.amazon.es/dp/B003PAKL7O/ref=asc_df_B003PAKL7O36094159/?tag=googshopes-21&creative=24514&creativeASIN=B003PAKL7O&linkCode=df0&hvdev=c&hvnetw=g&hvqmt=

[REF2.3.5]

http://www.movistar.es/particulares/internet/adsl-fibra-optica/oferta-fibra-optica-30mb/

[REF3.3.1]

http://www.axure.com/

[REF4.3.1]

http://www.wikitude.com/developer/licenses?p_p_id=58&p_p_lifecycle=0&p_p_state=maximized&p_p_mode=view&p_p_col_id=column-1&p_p_col_pos=1&p_p_col_count=2&saveLastPath=0&_58_struts_action=%2Flogin%2Fcreate_account

[REF4.3.2]

https://targetmanager.wikitude.com/projects

[REF6.1]

https://blog.jetbrains.com/idea/2014/09/touring-plugins-issue-1/

[REF6.2]

https://github.com/BasLeijdekkers/MetricsReloaded