Facultad de Ciencias

REALIDAD AUMENTADA EN ENTORNOS INDUSTRIALES

(Augmented Reality in Industrial Environments)

Trabajo de Fin de Máster
para acceder al

MÁSTER EN INGENIERÍA INFORMÁTICA

Autora: Rebeca Elena Bárcena Orero

Directora: Marta Elena Zorrilla Pantaleón

Junio 2018

Índice de contenidos

Índice de contenidos 3
Índice de tablas 4
Índice de figuras 4
Agradecimientos 6
Resumen 7
Abstract 8
1. Introducción 9
1.1 Introducción a la realidad aumentada 9
1.2 Antecedentes y objetivos 14
1.3 Metodología y Plan de trabajo 15
2. Análisis de tecnologías 18
2.1 Análisis de tecnologías de realidad aumentada 18
2.2 La tecnología elegida: Wikitude 23
2.3 Problemas con Wikitude y la lectura de códigos QR 24
3. Tecnologías y herramientas usadas 27
3.1 Tecnologías 27
3.1.1 Lenguajes de programación 27
3.1.2 Xamarin 27
3.1.3 Lectura de códigos QR 28
3.1.4 Tecnologías complementarias a la realidad aumentada 28
3.1.5 Reconocimiento de voz 28
3.2 Herramientas de desarrollo 29
3.2.1 Wikitude 29
3.2.2 Visual Studio 29
3.2.3 Wikitude 3D Encoder 29
3.2.4 Chrome DevTools 29
3.2.5 Safari Web Inspector 29
3.3 Herramientas de gestión de la configuración 29
3.3.1 Mopa 29
3.3.2 Jira 30
3.3.3 TortoiseSVN 30
4. Especificación de requisitos 31
4.1 Captura de requisitos 31

4.2 Requisitos funcionales..... 31
4.3 Requisitos no funcionales.....32
5. Diseño de la aplicación.....33
5.1 Diseño arquitectónico.....33
5.2 Diseño detallado.....33
6. Implementación 35
6.1 Capa de negocio.....35
6.2 Capa de presentación..... 37
6.3 Gestión de la configuración.....42
7. Pruebas..... 43
8. Integración en IDbox 44
8.1 Pruebas de integración.....47
9. Reconocimiento de voz 48
9.1 Reconocimiento de voz (Speech to text) y Ayuda auditiva (Text to speech).....48
9.2 Implementación.....49
10. Conclusiones y trabajos futuros 51
10.1 Conclusiones.....51
10.2 Trabajos futuros.....52
Referencias..... 54

Índice de tablas

Tabla 1.1. Patentes en el ámbito de la realidad aumentada y realidad virtual.....10
Tabla 1.2. Resumen de las tecnologías de realidad aumentada tenidas en cuenta.....22
Tabla 4.1 Requisitos funcionales..... 31
Tabla 4.2. Requisitos no funcionales..... 32

Índice de figuras

Figura 1.1 Previsión de crecimiento del mercado de AR.....11
Figura 1.2. Industrias que más fácilmente pueden adoptar AR.....11
Figura 1.3. Categorías de dispositivos para AR y VR.....13
Figura 1.4. El ecosistema de la AR y VR.....14

Figura 1.5 Diagrama de Gantt……16
Figura 1.6 Diagrama de Gantt……16
Figura 1.7 Diagrama de Gantt……16
Figura 1.8 Diagrama de Gantt……17
Figura 2.1. Comparativa de frameworks de AR……19
Figura 2.2. Comparación entre el top de las 8 mejores plataformas de AR……20
Figura 2.3. Comparativa de los proveedores de SDKs de AR más populares……21
Figura 2.4. Arquitectura del SDK de Wikitude……23
Figura 2.5. Composición de la vista……24
Figura 5.1. Diagrama de despliegue……34
Figura 6.1. Clases creadas dentro de Xamarin.Droid……35
Figura 6.2. Clases creadas dentro de Xamarin.iOS……35
Figura 6.3. Clase creada dentro de Xamarin……35
Figura 6.4. Código de la WebView en Android……36
Figura 6.5. Código de la WebView en iOS……37
Figura 6.6. Modelo 2D con botón para fijarlo a la pantalla……38
Figura 6.7. Mensaje superior…… 38
Figura 6.8. Botón para modelo 2D…… 39
Figura 6.9. Botones para modelo 3D…… 40
Figura 6.10. Modelo 3D tras haberse fijado a la pantalla y rotarse…… 40
Figura 6.11. Modelo 3D aumentado de tamaño…… 41
Figura 6.12. Modelo 3D con animaciones…… 42
Figura 8.1. Opción de realidad aumentada desde el menú de IDbox Mobile…… 44
Figura 8.2. Módulos usados en ambas plataformas y comparativa……45
Figura 8.3. Toolbar en Android……46
Figura 8.4. Toolbar en iOS……47

Agradecimientos

Me gustaría agradecer, tanto este trabajo como la motivación recibida para aventurarme a realizar este Máster, a esos buenos profesores que he tenido el placer de encontrar a lo largo de mis años en la Universidad, por haberme enseñado a amar el mundo de la Ingeniería Informática y, en especial, a Marta Zorrilla, por saber guiarme, orientarme y ayudarme siempre, por estar siempre ahí y ser un ejemplo a seguir.
En segundo lugar, a CIC Consulting Informático por confiar en mí para desarrollar este proyecto, y a todos los compañeros que he conocido allí, por conseguir ese buen ambiente de trabajo que te hace ir motivado a diario.
Y, por último, pero no menos importante, a mi familia y mis amigos por haberme apoyado siempre, y a mi pareja por ser el que me ha tenido que sufrir en los momentos de estrés y por haber sabido devolverme la confianza en mí misma cuando lo necesitaba.

Resumen

Actualmente la industria está inmersa en la denominada Cuarta Revolución Industrial (*Forth Industral Revolution* o 4IR). Esta tiene por objeto mejorar la eficiencia, productividad, calidad y seguridad de los procesos industriales gracias a la conexión y comunicación digital de todos los recursos implicados. Esta digitalización favorecerá enormemente la creación de escenarios virtuales a bajo coste, la simulación de procesos bajo condiciones reales y el acceso a información relevante en tiempo real, entre otros. En este ámbito es donde la realidad aumentada y la realidad virtual van a jugar un papel importante y, es por ello, que este TFM tiene por objeto estudiar las tecnologías disponibles y su grado de madurez con el fin de incorporarlas en aplicaciones móviles de apoyo a procesos industriales como, por ejemplo, la realización de tareas de mantenimiento del entorno industrial y el control y supervisión de los datos de los diferentes elementos que se encuentren en él.

Como prueba de concepto, se construye un motor de renderizado que permite ser fácilmente integrado en *Smartphones*. Este incluye las siguientes funcionalidades: escanear códigos QR, mostrar contenidos en realidad aumentada, mostrar datos en tiempo real, usar el giroscopio y acelerómetro del móvil y utilizar reconocimiento de voz para acciones sobre el motor.

El módulo de realidad aumentada se ha desarrollado para teléfonos móviles con sistema operativo iOS y Android utilizando Xamarin.Forms como *framework* para la integración de tecnologías.

Palabras clave: industria 4.0, realidad aumentada, códigos QR, app

Abstract

Currently, industry is immersed in the so-called Forth Industrial Revolution (4IR). It aims at improving efficiency, productivity, quality and safety of industrial processes thanks to the digital connection and communication among all the involved resources. This digitization will greatly benefit virtual scenes' creation at low cost, simulations of processes under real conditions and access to relevant information in real time, among others. In this area is where augmented reality and virtual reality will play an important role and, therefore this TFM aims at studying available technologies and its maturity level in order to incorporate them into mobile applications which support industrial processes such as carrying out maintenance tasks in the industrial environment or control and monitor the different elements deployed on it.

As a proof of concept, a rendering engine, which could be easily integrated into *Smartphones*, is built. It includes the following functionalities: scan QR codes, show contents in augmented reality, show data in real time, use the gyroscope and the accelerometer of the mobile phone and use voice recognition for actions on the engine.

The augmented reality module has been development for mobile phones with iOS and Android operating systems using Xamarin.Forms as framework for the integration of technologies.

Keywords: industry 4.0, augmented reality, QR codes, app

1. Introducción

En este capítulo se realiza una introducción al concepto de la realidad aumentada, ámbito en el que se desarrolla este TFM. Asimismo, se describen los antecedentes que promovieron este trabajo y los objetivos que se persiguen. Por último, se detalla el plan de trabajo y la metodología seguida en su desarrollo.

1.1 Introducción a la realidad aumentada

Hoy en día el uso de *Smartphones* y *Tablets* está muy extendido, no solo en el ámbito personal, sino en el profesional y empresarial. El desarrollo de nuevas tecnologías para aumentar sus capacidades y lograr una mejor experiencia de usuario no deja de crecer, siendo la realidad aumentada y la realidad virtual dos herramientas que han adquirido una gran importancia estos últimos años. Basta con observar en la Tabla 1.1 [1] cómo ha incrementado el número de patentes en este sector de acuerdo a cuatro categorías: a) las patentes de rastreo ocular, técnicas y desarrollos que tratan de capturar el movimiento ocular para realizar acciones en función de ello, como puede ser bajar el brillo de la pantalla para ahorrar energía si el usuario no está observando; b) *force feedback* y *tactile feedback* aglutina patentes que hacen referencia a simulaciones que replican características o comportamientos de objetos reales con el fin de asemejarse más a los objetos que emulan, como puede ser un volante de una videoconsola que vibra cuando se pisan las líneas externas de una carretera u ofrece resistencia; c) bajo modelado 3D se encuentran cualquier *hardware* o *software* que emplean modelos 3D, ya sea para realidad virtual o aumentada; y, por último, d) SLAM (*Simultaneous Localization and Mapping*) que engloba la funcionalidad que proporciona la realidad aumentada de escanear objetos reales de tres dimensiones y mostrar contenido en 360°.

Tabla 1.1. Patentes en el ámbito de la realidad aumentada y realidad virtual

Año	Rastreo ocular	*Force feedback* y *tactile feedback*	Modelado 3D	SLAM
2016	178	215	1091	48
2015	115	134	888	54
2014	109	99	602	32
2013	76	74	538	21
2012	74	75	418	31
2011	22	66	348	10
2010	16	56	261	11
2009	8	34	195	7
2008	8	39	186	5
2007	7	24	119	6
2006	11	24	114	4
2005	6	12	96	5
2004	4	16	91	2
2003	2	23	79	2
2002	4	19	86	0
2001	9	17	75	0
2000	6	21	86	0
1999	5	13	58	0
1998	3	10	57	0

Como se puede comprobar, la tecnología SLAM es una tecnología bastante nueva, y que está creciendo rápidamente, llevando el resto más tiempo en desarrollo.
En la Figura 1.1 se muestra la previsión del crecimiento del mercado de la realidad aumentada según el informe de la PTC [2], que lo estima en un 118% para el periodo 2016-2021.

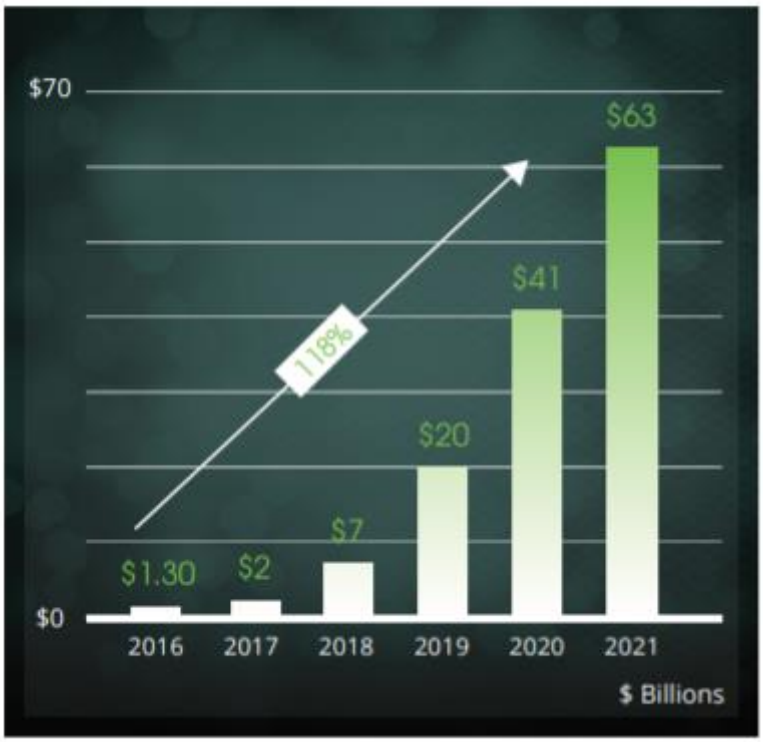

Figura 1.1 Previsión de crecimiento del mercado de AR [2]

En la Figura 1.2 se muestran los sectores que más se pueden beneficiar de las características de la realidad aumentada, en las que destacan el entorno industrial y empresas relacionadas con el software.

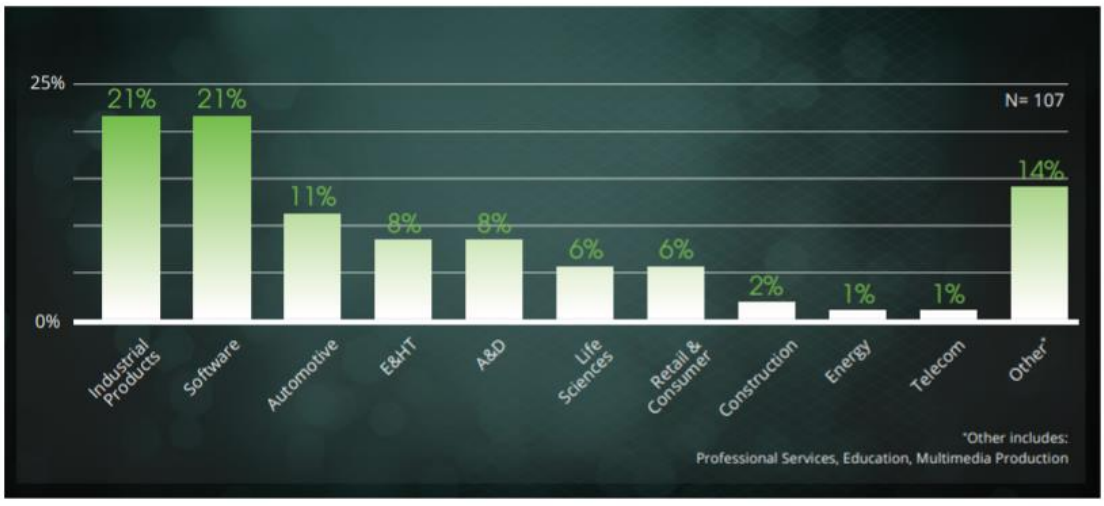

Figura 1.2. Industrias que más fácilmente pueden adoptar AR [2]

¿Pero qué es la realidad aumentada? El término realidad aumentada (*Augmented Reality*, AR), que fue utilizado por primera vez en 1990 por Tom Caudell [3], se refiere a la combinación de tecnologías que permiten la generación de contenido en tiempo real combinada con la visión de un entorno físico del mundo real.
La realidad aumentada no sólo tiene finalidades relacionadas con el entretenimiento, sino que puede ser usada en los negocios [4], en la educación utilizando *apps* como *Google Sky Map* o *Elements 4D,* o *Anatomy 4d* para explorar el cuerpo humano, en el diseño de interiores permitiendo ver los resultados antes de llevarse a cabo, en el turismo, proporcionando información de los lugares de interés que se están visitando, o creando nuevos modelos de marketing que permitan que el usuario interaccione, por ejemplo, con las revistas, catálogos o pósteres.[4] [3][5].
La realidad aumentada tiene su propia terminología y es preciso conocerla para poder utilizarla. Uno de los elementos principales de la realidad aumentada es el *target* o *marker*. Este es un objeto del mundo real que, una vez capturado por la cámara del *Smartphone* o *Tablet*, permite mostrar la realidad aumentada que se especifique. En función de si la aplicación cuenta con *target* o no, se diferencia entre tecnologías *Marker-based* o *Marker-less.* Esta última es más costosa y difícil de implementar pues debe reconocer objetos que no están predefinidos. [6].
Dentro de las tecnologías *Marker-less* se encuentran las *Location-based* [7], que son aquellas que usan el GPS y otros detectores de posición, como el acelerómetro, para mostrar objetos de realidad aumentada dependiendo de la posición del usuario, o el *Object Recognition* [8], en la que el *tracker* no es una imagen u objeto 2D, sino un objeto 3D.
Dentro de las aplicaciones *Marker-based* se está empezando a incluir también una nueva opción, que es *Multiple Targets Recognition*, que permite reconocer simultáneamente varios *targets* y mostrar, a la vez, la realidad aumentada correspondiente a cada uno [6].
Por último, *Reliable Target Recognition* es una característica ofrecida por las librerías de realidad aumentada que indica que el reconocimiento de los *targets* funciona correctamente, siempre y cuando estos tengan un tamaño y calidad adecuados y se escaneen desde una distancia no excesiva, siendo especificada por cada librería [6]. Esta característica es muy importante a la hora de analizar qué librería o librerías se desean usar, si lo que se busca es construir una aplicación *Marker-based*.
En la Figura 1.3 se muestran algunos tipos y ejemplos de dispositivos que permiten incorporar realidad virtual y realidad aumentada. La realidad virtual está algo más restringida, puesto que requiere de dispositivos que sumerjan al usuario en un nuevo mundo virtual, completamente digital y que no interactúa con el mundo real. En cambio, la realidad aumentada sólo requiere de un *Smartphone* o *Tablet* con cámara, por lo que su uso progresivamente se generalizará.

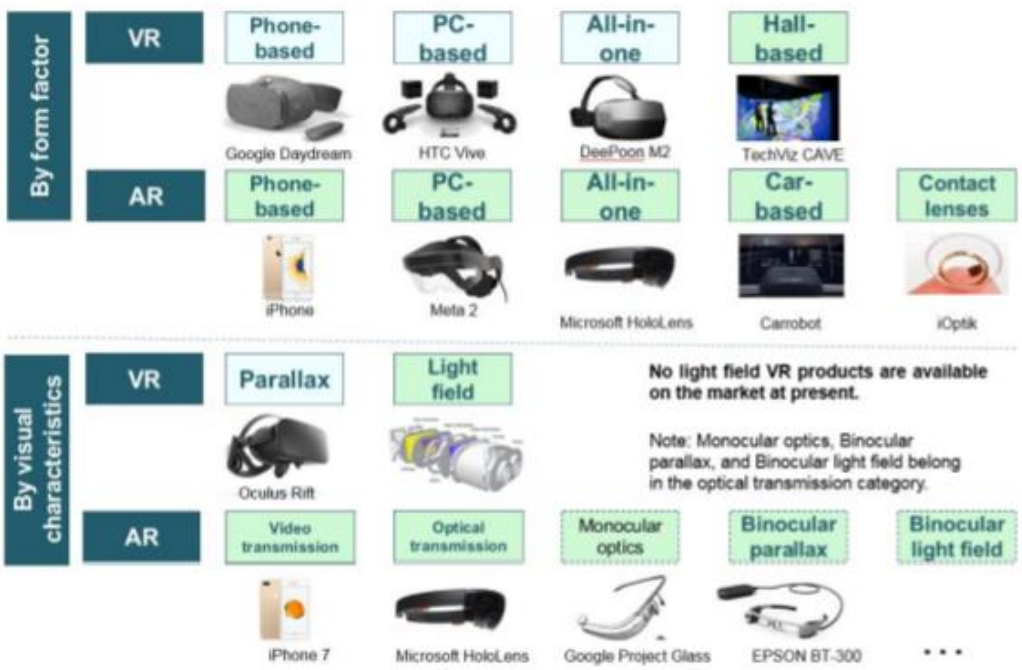

Figura 1.3. Categorías de dispositivos para AR y VR [9]

Por último, la Figura 1.4 recoge de forma resumida los distintos dispositivos y elementos que constituyen el ecosistema de la realidad aumentada y la realidad virtual, así como posibles aplicaciones de dichas tecnologías.

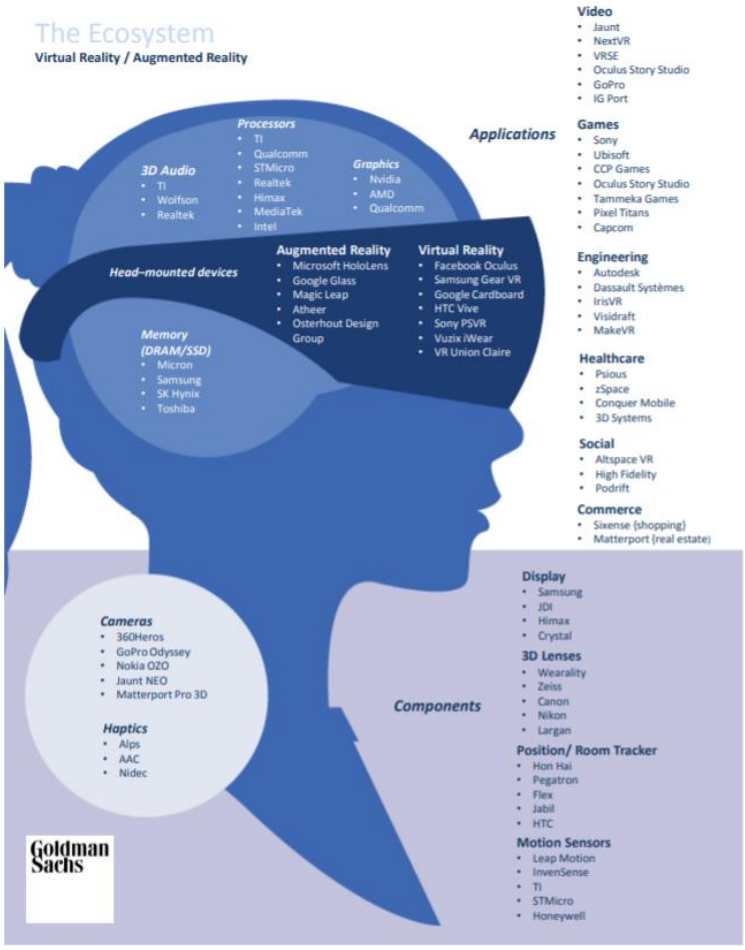

Figura 1.4. El ecosistema de la AR y VR [10]

1.2 Antecedentes y objetivos

Este proyecto se enmarca como un proyecto de I+D dentro de la empresa CIC Consulting Informático, a propuesta de la subscritora de este TMF, por su interés en estudiar y analizar las posibilidades que permite la realidad aumentada e implementar un motor de renderizado capaz de crear modelos 2D y 3D, así como mostrar datos en tiempo real asociados a códigos QR.

Dicha empresa cuenta con un producto, denominado IDbox, que incluye un software SCADA (*Supervisory Control And Data Acquisition*) que permite controlar y supervisar datos y procesos industriales a distancia. La incorporación de AR permitiría visualizar datos en tiempo real dentro de las propias instalaciones facilitando la realización de tareas de mantenimiento y comprobaciones rutinarias, así como disponer de simulaciones de zonas de la planta con restricciones de acceso.

Esta empresa ya cuenta con una app de IDbox para *Smartphones*, desarrollada en Xamarin.Forms, por lo que es necesario trabajar con dicha tecnología y, posteriormente, integrar el motor dentro de su app.

En resumen, el objeto de este TFM es construir un motor de renderizado para *Smartphones* que pueda ser integrado en IDbox y que proporcione al menos la posibilidad de facilitar las tareas de mantenimiento en entornos industriales al poderse

emplear la realidad aumentada en ellos, mostrando elementos 2D, 3D y datos en tiempo real.

1.3 Metodología y Plan de Trabajo

Como se ha mencionado, este proyecto se concibió como un desarrollo I+D y, por tanto, no sujeto a resultados finales bien establecidos ya que se partía de cero, esto es sin conocimientos en la materia y sin expertos en este ámbito en la empresa. El trabajo ha sido realizado exclusivamente por la subscritora del este TFM tratando de proporcionar al motor de renderizado del mayor número de funcionalidades. Dado que sí había una delimitación temporal, se realizó una exhaustiva Gestión de Proyectos.
Inicialmente se definieron y estimaron las tareas. y, semanalmente, se iba contabilizando y detallando el tiempo diario que se empleaba en su desarrollo, usando las herramientas Mopa y Jira, (ver sección ***3. Tecnologías y herramientas usadas***.) A medida que se iban cubriendo objetivos se abordaban nuevas funcionalidades.
Se establecieron reuniones frecuentes con el experto en temas relacionados con apps móviles y Xamarin y de carácter mensual con los responsables del proyecto si se habían tenido avances significativos.
En cuanto al plan de trabajo, se estableció en unas 658h, correspondientes a 4 meses a jornada completa, para realizar un estudio y análisis de tecnologías de realidad aumentada, desarrollar el motor de renderizado con el mayor número de funcionalidades e integrarlo en IDbox.
Debido a que las tareas fueron desarrolladas por una única persona, en ocasiones se realizaron en otro orden o, incluso, se paralelizaron. En las Figuras 1.1, 1.2, 1.3 y 1.4 se recoge el diagrama de Gantt, observando las tareas desglosadas y su estimación en horas (se han eliminado las relacionadas con el acelerómetro y el giroscopio por no haberse desarrollado finalmente).

1	**Desarrollo del motor de AR**	**594 horas**
2	Definición de objetivos y requisitos	5 horas
3	Reuniones	40 horas
4	Iniciación con Xamarin	8 horas
5	**Investigación de librerías de AR**	**3,5 días**
6	Búsqueda de librerías de AR para dispositivos	4 horas
7	Búsqueda de librerías de AR para Xamarin.Forms	4 horas
8	Análisis de las librerías (con o sin pruebas)	20 horas
9	Análisis de las tareas a realizar y definición de tiempos requeridos	10 horas
10	Pruebas con la librería elegida	30 horas
11	Investigación de cómo leer códigos QR	30 horas

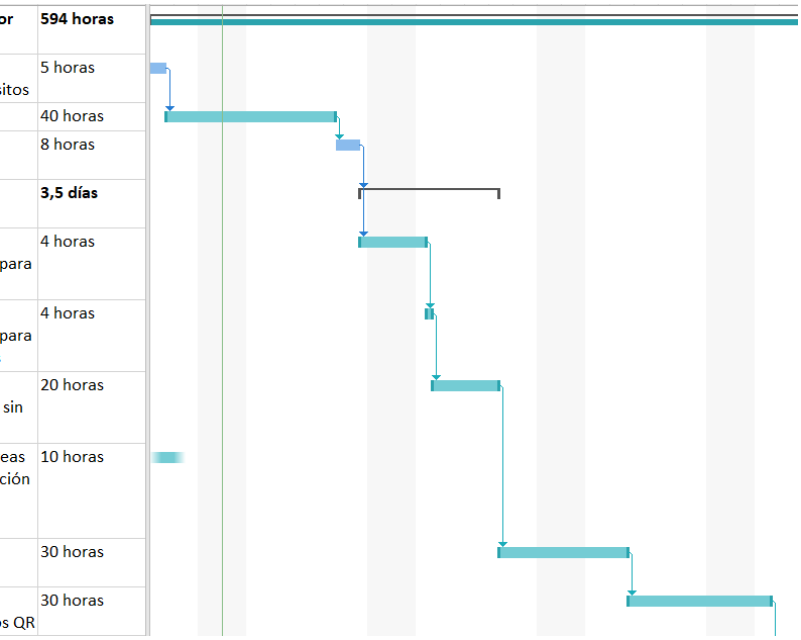

Figura 1.5 Diagrama de Gantt

11	Investigación de cómo leer códigos QR	30 horas
12	**Diseño de una prueba de concepto**	**10 días**
13	Introducción de la funcionalidad de lectura de códigos QR en la librería	40 horas
14	Carga de SVGs como elementos de AR	10 horas
15	Carga de SVGs en función de la URL decodificada del QR	30 horas
16	**Definición e implementación del motor**	**18,75 días**
17	Inclusión de modelos en 2D y	34 horas
18	Inclusión de videos	33 horas
19	Botones para interactuar con el elemento	30 horas
20	Inclusión de datos en tiempo real	33 horas

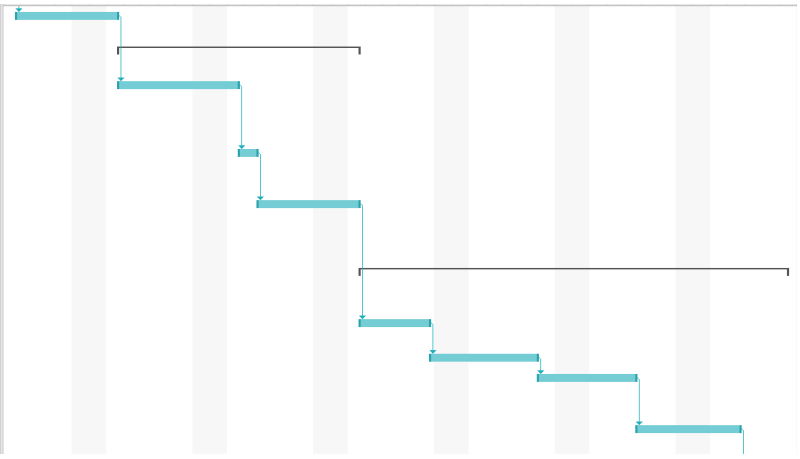

Figura 1.6 Diagrama de Gantt

20	Inclusión de datos en tiempo real	33 horas
21	Pruebas	20 horas
22	**Integración del motor en Idbox**	**10 días**
23	Integración	20 horas
24	Añadir botón de atrás	10 horas
25	Pruebas	50 horas
26	**Extensibilidad del motor con nuevas características**	**9 días**
27	Mostrar tabla con datos en tiempo	24 horas
28	Cargar gráfica a partir de los datos de la tabla	16 horas
29	Botones para detener/reanudar los datos en tiempo real	5 horas
30	Análisis de librerías de reconocimiento de voz para Xamarin	3 horas

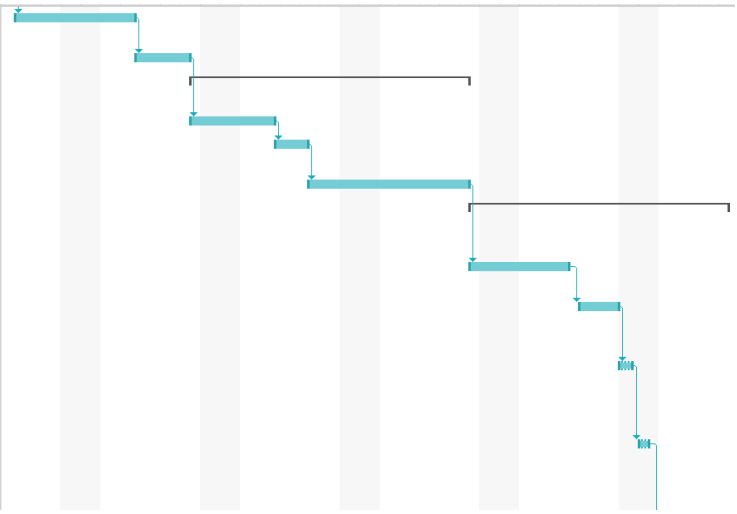

Figura 1.7 Diagrama de Gantt

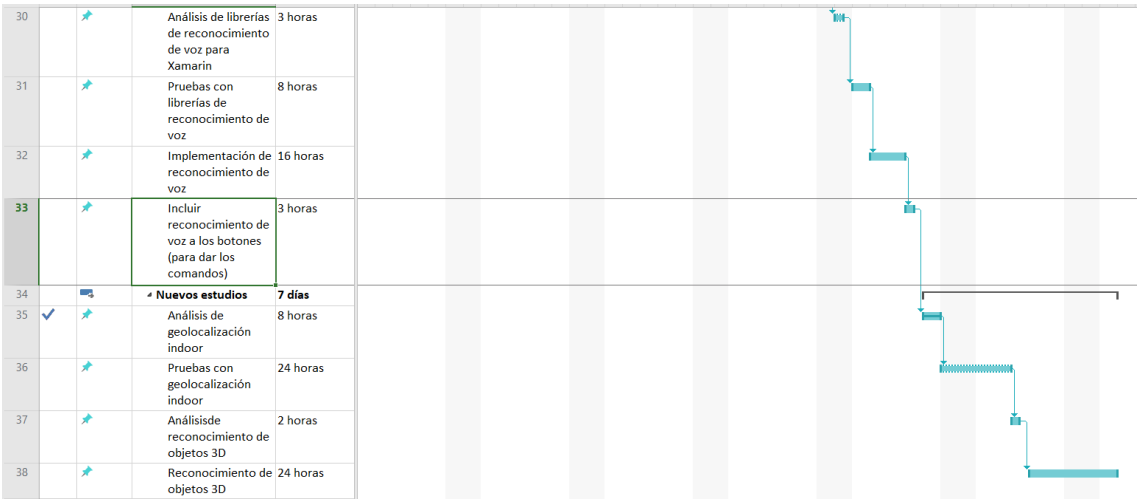

Figura 1.8 Diagrama de Gantt

2. Análisis de tecnologías

En este capítulo se recoge el proceso seguido para analizar las tecnologías de realidad aumentada disponibles en el mercado, así como una breve explicación de la tecnología escogida y la problemática encontrada.

2.1 Análisis de tecnologías de realidad aumentada

Tras los últimos avances en *Smartphones*, son muchas las librerías disponibles para prestar ayuda a aquellos que quieran incorporar las utilidades de la realidad aumentada en sus propias aplicaciones. Para evaluar sus capacidades y grado de madurez, se llevó a cabo un estudio de las tecnologías disponibles en el mercado, centrándose en aquellas que fuesen compatibles con Android y/o iOS, puesto que teniendo en cuenta ambos sistemas operativos se cubre prácticamente todo el mercado de dispositivos móviles. De acuerdo con [11], en el año 2017 el número de dispositivos vendidos que usaban uno de estos dos S.O. era de un 99,6%.
Además de tener en cuenta ambos sistemas operativos, el estudio se particularizó en tecnologías que fuesen soportadas por Xamarin.Forms, tecnología utilizada para desarrollar aplicaciones móviles dentro de la empresa, con objeto de reducir problemas de integración.
Fueron varias las tecnologías que se estudiaron y tuvieron en cuenta. Debido a que las tecnologías de realidad aumentada son tecnologías relativamente actuales, con pocos años en el mercado, no hay muchos estudios ni *benchmarks* que las comparen. Por ello se decidió realizar una búsqueda online de las últimas tecnologías disponibles, teniendo en cuenta artículos, blogs relacionados con tecnologías y páginas de las propias casas de software para introducirme en este campo y, seguidamente, realizar una búsqueda de comparativas sobre ellas. Básicamente, lo que se buscaba era encontrar tecnologías AR compatibles con Xamarin.Forms y que contasen con todas las prestaciones necesarias para cumplir los requisitos del proyecto, que se detallan en el ***Apartado 4. Especificación de requisitos***. Asimismo, se tuvieron en cuenta también tecnologías propias de Android o iOS, así como otras compatibles con ambos sistemas operativos, ya que podría llegar a implementarse en Xamarin.Forms, aunque fuera un proceso mucho más costoso.
A continuación, se muestran las figuras correspondientes a las comparativas tenidas en cuenta. Cabe señalar que se fijaron las siguientes premisas:

- La tecnología permite mostrar, como realidad aumentada, modelos 2D, 3D, contenido web, vídeos y texto.
- La tecnología es compatible con Xamarin.Forms.
- La tecnología ofrece soporte actual y futuro. Esta premisa es importante, puesto que se comprobó que muchas tecnologías han estado durante unos años en el mercado y actualmente no son soportadas, aconsejando ser sustituidas por otras nuevas. Este es el ejemplo de Project Tango, que cerraba el proyecto en marzo de 2018 [12] y recomendaba el uso de ARCore, tecnología que sólo era compatible con un par de dispositivos Android.
- La tecnología debe poder ser soportada por multitud de dispositivos, siempre que tengan un sistema operativo Android o iOS.

Además, se tuvieron en cuenta otros requisitos, aunque menos relevantes, que fueron:

- La tecnología puede ser implementada en *Smart Glasses*, ya que hay empresas que apuestan por ellas.
- La tecnología permite el reconocimiento de objetos 3D.
- La tecnología permite decodificar códigos QR. Esta premisa era la menos importante, porque si no se encontraba, se incorporaría un lector de códigos QR que hiciera esa función.

En cuanto a las comparativas, en la figura 2.1 se presentan varias tecnologías soportadas por Xamarin, como son Wikitude y Vuforia, contándose con algo de experiencia en esta última antes de desarrollar el proyecto. A partir de esto se profundizó el estudio en ambas tecnologías, aunque sin dejar de lado el resto.

	ARToolkit	Vuforia	ARmedia	XZIMG	Kudan AR Engine	Wikitude
Type	Open Source	Free + Commercial SDK option	Free + Commercial SDK option	Free + Commercial SDK option	Free + Commercial SDK option	Free + Commercial SDK option
iOS	✓	✓	✓	✓	✓	✓
Android	✓	✓	✓	✓	✓	✓
Marker	✓ Basic	✓ Advanced + VUMark	✓	✓	✓	✓ Advanced
NaturalFeature	✓	✓	✓	✓	✓	✓
Windows Mobile	✓	✓ Vuforia now available for windows app dev & also for MS Hololens	− Soon		✓	✓
Web		✗	✓	✓ HTML5	✗	
PC/Mac/Linux	PC/Mac/Linux		ARmedia supports PC/Mac and Linux	via Unity/Native	Cros-platform development Unity3D	
3D Object Tracking	✗	✓ Only on box and cylinder and small size 3D objects too	✓	✓ Beta	✓ SLAM	✓
GPS		✓	✓		✓	✓
IMU Sensors		✓	✓		✓	✓
VisualSearch		✓		✓	✓ unlimited local visual search (no network connection required)	✓ Cloud Recognition and Offline (on device)
FaceTracking		✗	✗	✓	− Extensible with FT/FR Plugins	− Face Detection
ContentAPI		✓ With Vuforia Cloud		✓		✓ with Wikitude Studio and Cloud Recognition
Unity (3D)	✓	✓	✓	✓	✓ With SLAM	✓ 3D Tracking Included
Plugins API						Integration with 3D party libraries available
Website	artoolkit.org	vuforia.com	inglobetechnologies.com	xzimg.com	kudan.eu	wikitude.com

Figura 2.1. Comparativa de frameworks de AR [13]

En la figura 2.2 se muestra otra comparativa similar realizada por [7]. Como se puede observar las tecnologías más completas son Vuforia y Wikitude. En este caso, se comprobó que las diferencias principales eran que Vuforia podría ser usado desde UWP (Universal Windows Platform) y que Wikitude permitía el uso de SLAM. Dentro de la empresa no se implementan aplicaciones móviles para Windows, por lo que el hecho de tener soporte para UWP era irrelevante. Sin embargo, el SLAM sí que podría resultar útil al permitir escanear objetos 3D y mostrar la realidad aumentada en función de ellos. Como este proyecto se orientaba a fines industriales, se consideró mejor opción Wikitude al contar con esta funcionalidad, puesto que podría facilitar labores de mantenimiento al poder escanear una máquina y mostrar sobre ella las actividades a realizar.

	Vuforia	Wikitude	EasyAR	Kudan	ARToolKit	Maxst	Apple ARKit	XZIMG
Licence	Free, Commercial	Commercial	Free, Commercial	Free, Commercial	Free Open Source	Free, Commercial	Free	Free, Commercial
Supported platforms	Android, iOS, UWP	Android, iOS	Android, iOS, UWP, macOS	Android, iOS	Android, iOS, Linux, Windows, macOS	Android, iOS, Windows, macOS	iOS	Android, iOS, Windows
Smart glasses support	+	+	-	-	+	+	+	-
Unity support	+	+	+	+	+	+	+	+
Cloud recognition	+	+	+	-	-	-	+	-
3D recognition	+	+	+	+	-	+	+	-
Geolocation	+	+	-	-	+	-	+	-
SLAM	-	+	+	+	-	+	+	-

Figura 2.2. Comparación entre el top de las 8 mejores plataformas de AR [7]

Por último, se tuvo en consideración la Figura 2.3, que considera sólo a los tres proveedores de realidad aumentada más populares, comprobando una vez más que Wikitude y Vuforia son líderes en este sector. Tras volver a comprobar que Wikitude se ajustaba mejor a las necesidades, se realizó una búsqueda detallada dentro de la página web de la compañía sobre todas sus funcionalidades, así como en foros técnicos para conocer la opinión de otros usuarios y desarrolladores. Tras ello, Wikitude fue elegida, definitivamente, como principal herramienta de realidad aumentada del proyecto.

Features	Wikitude	Vuforia	MetaIO
Platform	Android, iOS, Titanium, Xamarin, PhoneGap	iOS, Android, Windows, Unity	iOS, Android, Windows, Unity
2D target tracking	Yes	Yes	Yes
3D target tracking	None	Yes	Yes
2D and 3D Content	Yes	Yes	Yes
Video Content	Yes	Yes	Yes
Multiple target recognition	Yes	Yes	Yes
Cloud target recognition	Yes	Yes	Yes
Face tracking	None	None	Yes
GPS	Yes	None	Yes
SLAM support	Yes	None	Yes
Target Generation	Yes	Yes	None
Apparatus support	Mobile Devices, Smart Glasses	Mobile Devices, Smart Glasses (Beta)	Mobile Devices

Figura 2.3. Comparativa de los proveedores de SDKs de AR más populares [6]

Para llegar a esta conclusión, previamente se elaboró una tabla, donde se anotaron las tecnologías de realidad aumentada que se consideraron más relevantes. En ella se recogieron datos sobre si las tecnologías podían ser integradas, si soportaban SLAM o si permitían reconocer objetos de tres dimensiones. También se recogieron datos acerca de si presentaban limitaciones en cuanto a dispositivos que podían usarlas o si tenían futuro, refiriéndose a que si eran proyectos que se sabía que iban a cerrarse o se preveía seguir trabajando en él. Esta tabla es la siguiente:

Tabla 1.2. Resumen de las tecnologías de realidad aumentada tenidas en cuenta

Tecnología	Soportada por Xamarin	SLAM	Reconocimiento de objetos 3D	Variedad de dispositivos soportados	Con futuro	Comentarios
Project Tango	No	Sí	Sí	No, sólo permite dos dispositivos Android	No, se cierra en marzo de 2018	
ARCore	No	Sí	Sí	Disponible para pocos dispositivos	Sí	Existe una versión compilada para Android 7 Nougat y posteriores, pero sólo ofrece la opción *Markerless*
Vuforia	Sí, pero da problemas al integrarse en algunas versiones (Xamarin 2015)	No	Sí	Sí	Sí	
ARToolKit	No	No	No	Sí	Sí	
Wikitude	No directamente, pero puede realizarse	Sí	Sí	Sí	Sí	
Kudan	No	Sí	Sí	Sí	Sí	
EasyAR 2.0	No	Sí	Sí	Sí	Sí	
ARKit		Sí				
MetaIO	No, al igual que Vuforia da fallos	Sí	Sí	Sí	Sí	
Aurasma	No, da problemas al integrarse en Xamarin	No				Está orientado más a personas sin conocimiento de programación

En un primer lugar, la intención era elegir las mejores y realizar pruebas con ellas, para observar cuál o cuáles eran las óptimas para este caso, integrando incluso varias y aprovechando lo mejor de cada una.
Sin embargo, al comprobar que Wikitude poseía todo lo necesario, se comenzaron a realizar pruebas con esta tecnología. Al realizarlas, se descartó la idea de usar varios proveedores de realidad aumentada simultáneamente, ya que no se puede usar la cámara de un dispositivo móvil por dos aplicaciones paralelamente, como pueden ser dos tecnologías de realidad aumentada o una de realidad aumentada y un escáner de códigos QR. Esto ocurre porque la cámara es un recurso que no se puede compartir, sino que se accede a él de forma única. Tras realizar pruebas, incluso de intercalar el uso de la cámara desde varias tecnologías, se descartó el integrar varias tecnologías de realidad aumentada, eligiendo única tecnología a Wikitude, de la que se hablará más detalladamente en el siguiente apartado.

2.2 La tecnología elegida: Wikitude

Wikitude es una tecnología móvil de realidad aumentada, cuyo SDK es usado por más de 10.000 aplicaciones [14], entre las que destacan aplicaciones de empresas como Epson, Ford, SAP, Sony, Harvard University, 20th Century Fox o TripAdvisor.
Gracias a su *core* basado en C++ es multiplataforma, pudiendo ser usada en *wearables* (dispositivos que se incorporan en alguna parte de nuestro cuerpo [15] como pueden ser relojes, pulseras o gafas inteligentes), *Smartphones* o *Tablets*. Para facilitarlo cuenta con un SDK desde el que se puede acceder mediante APIs nativas o JavaScript.
Como podemos ver en la Figura 2.4, cuenta con dos APIs principales, una de Java, que permite conectar a Unity, y otra desde JavaScript, que permite que Cordova, Titanium y Xamarin puedan trabajar con Wikitude. Además, cuenta con una tercera API, relacionada con *plugins*. Estos permiten crear módulos que puedan acceder a la cámara, por ejemplo, para leer códigos QR, evitando el problema anteriormente comentado. Sin embargo, no son accesibles desde todas las tecnologías.

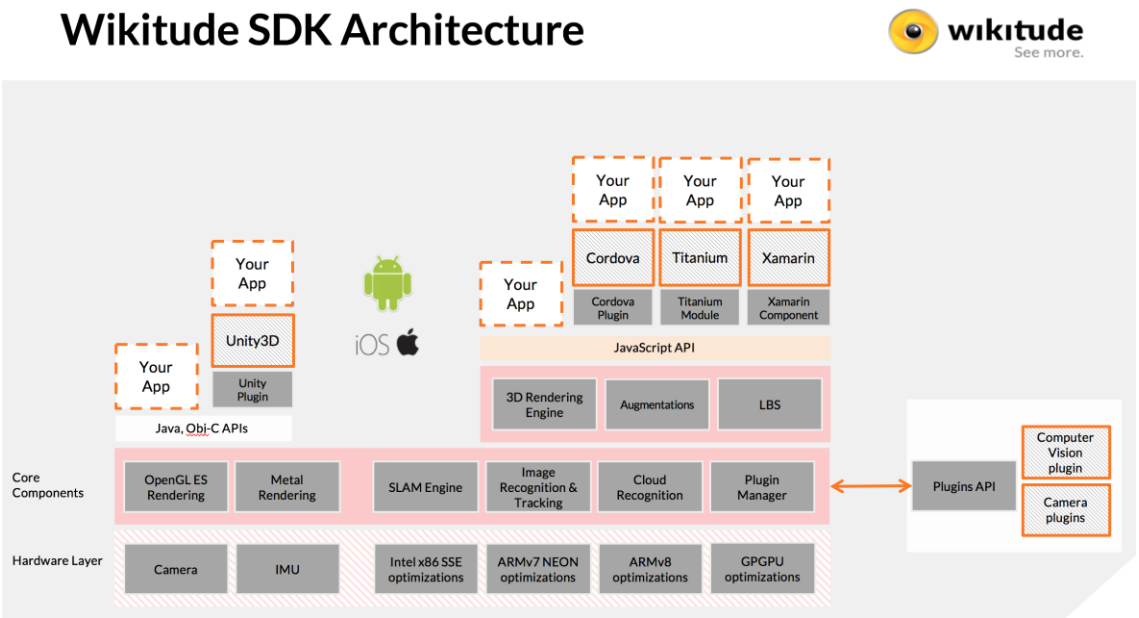

Figura 2.4. Arquitectura del SDK de Wikitude [16]

El uso de esta tecnología se basa en crear una web (*WebView*) en el dispositivo, la cual cuenta con el fondo transparente. Una vez la cámara es accedida y el *WebView* mostrado sobre la pantalla, se permite observar lo que está capturando la cámara. Sobre este *WebView* se renderizan los elementos de realidad aumentada correspondientes en cada caso.
Un ejemplo de cómo funcionaría esta arquitectura se muestra en la Figura 2.5. Como se puede ver son dos capas, una correspondiente al renderizado de la cámara y otra al *WebView*, encargado de renderizar la realidad aumentada.

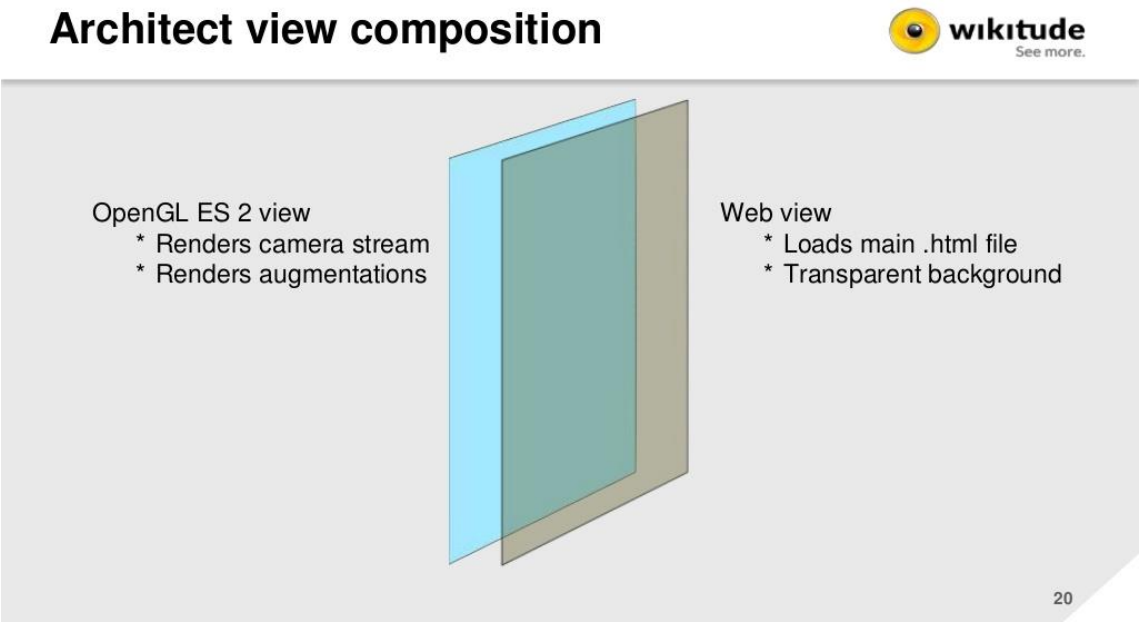

Figura 2.5. Composición de la vista [14]

Para trabajar con estos elementos de realidad aumentada, se emplean ficheros .html, .js y .css, lo que facilita tanto la creación de los propios elementos como su estilo.
Los tipos de datos que permite mostrar son imágenes, vídeos, modelos 3D, vistas de páginas web y etiquetas, que pueden contener cualquier tipo de texto, como pueden ser datos en tiempo real.

2.3 Problemas con Wikitude y la lectura de códigos QR

Aunque Wikitude indica que da soporte a Xamarin.Forms, no es cierto. Sólo da soporte a Xamarin.Android y Xamarin.iOS, por lo que fue necesario trabajar con todas estas tecnologías para poder utilizarla desde Xamarin.Forms.
Dado que Wikitude no soportaba la decodificación de códigos QR, se realizaron bastantes pruebas con diversas tecnologías.
En un primer lugar, se estudió la tecnología Anyline, puesto que Wikitude comentaba en su página web que era posible emplearla para decodificar códigos QR, códigos de barras, … Sin embargo, incluso con los programas de ejemplo que ofrecían, esto no funcionaba. Tras ello se intentó acceder a la cámara con algunas tecnologías de lectura de códigos QR, como ZXing (ya usada en la aplicación de la empresa, donde se

integrará después el motor de realidad aumentada), pero los dispositivos no permiten el acceso simultáneo de varias librerías o tecnologías a la cámara, y si se añadiesen soluciones como semáforos o similares, con el fin de liberar y usar recursos, no se conseguiría un uso eficiente de la cámara, ni el rendimiento requerido de la RA.
Por ello, se investigó y descubrió que Wikitude permitía el uso de "*plugins*", de modo que el programador podría crear módulos que, extendiendo del módulo "*plugin*" de Wikitude, permitía usar la cámara para lo que se desease, sin los problemas que las pruebas anteriores habían acarreado. El problema encontrado fue que estos *plugins* no se encontraban disponibles para ser implementados desde Xamarin, ni Xamarin.Forms, ni Xamarin.Android ni Xamarin.iOS, pero sí existían los módulos para ello dentro del propio SDK de Wikitude, por lo que se intentó implementar. Tras múltiples pruebas e intentos tanto en nativo como pasando los propios módulos a nativo (*Binding libraries* o JNI), no se obtuvieron los resultados deseados.
Finalmente, se decidió realizar capturas de pantalla cuando fuese necesario, y así decodificar la imagen con el código QR. También hubo que solucionar problemas en esto, ya que la captura de la pantalla congelaba la misma, haciendo que lo que mostraba la cámara se quedase varios segundos estático, por lo que se emplearon *threads* o hilos para mejorar esta situación.
Para decodificar esta imagen se pensó en dos propuestas, por un lado, la propia librería usada en la empresa, ZXing, y por otro lado Jsqrcode, librería basada en ZXing, pero usada desde JavaScript.
Se realizaron pruebas con la segunda, puesto que al realizar el código en JavaScript podría ser reutilizado por todos los sistemas operativos, pero se comprobó que fallaba demasiado la decodificación de las imágenes. Por ello, se realizaron pruebas con ZXing, desde C#, pero al ser librerías similares, los fallos eran parecidos.
No se emplearon más librerías puesto que, para JavaScript no había de hecho ninguna más capaz de decodificar códigos QR desde una imagen y, para C#, ZXing era la que mejor funcionaba.
La solución fue emplear Jsqrcode y estudiar los fallos, para tratar de mejorarlos en la medida de lo posible. Con algunas mejoras se consiguió pasar de una tasa de aciertos de cerca de un 10% a una tasa de aciertos mayor del 70%.
El último inconveniente tratado fue la propia captura de pantalla. Podría realizarse periódicamente, pero el rendimiento no sería tan apropiado. Por ello, era necesario sólo realizarse cuando hubiese un código QR. El problema hallado fue que, para que Wikitude detectase una imagen, era necesario crear previamente un fichero especial con dicha imagen o imágenes. Introducir todos los códigos QR era algo inviable, puesto que, si la aplicación creciese, sería costoso estar actualizando dicho fichero cada vez que se introdujese un nuevo código QR.
Por ello, se diseñó una imagen de IDbox, para poder ser detectada, que sería colocada sobre los códigos QR. Se realizaron varias pruebas antes y durante este diseño, desde tratar de detectar simplemente un icono introducido en el centro del código QR hasta varios tipos de imágenes. La primera opción mostró que el icono era demasiado pequeño para ser detectado, por lo que se eligió colocar un diseño de IDbox, que se fue refinando hasta ser lo suficientemente detectable.
Obviamente, esta imagen puede ser sustituida por cualquier otra para poder reutilizar el motor en cualquier otro contexto.
Se pensó, en un primer momento, realizar la captura de pantalla en nativo, sin embargo, las capturas no mostraban nada, ya que el sistema no tenía acceso a los recursos de Wikitude. Tras una investigación, se encontró que el propio Wikitude contaba con la posibilidad de realizar capturas de pantalla desde JavaScript, ya fuese sólo de lo que

mostraba la cámara o de ello junto con los elementos de realidad aumentada. Tras realizar pruebas uniendo esto con la librería de lectura de QRs, se comprobó que todo funcionaba correctamente.
En cuanto a la comunicación necesaria entre el código de Xamarin.Android o Xamarin.iOS en C# y el código en JavaScript, se probó la librería XLabs de Xamarin, pero no se obtuvieron buenos resultados, ya que dicha librería parece estar abandonada (lleva desde antes del 2017 sin actualizarse) y posee muchas de sus funcionalidades en Beta. Por último, se analizaron las opciones de Wikitude, puesto que permitía la comunicación bilateral entre JavaScript y Xamarin.Android/Xamarin.iOS, y al comprobarse que era rápida y eficaz, decidió emplearse.

3. Tecnologías y herramientas usadas

En este capítulo se recogen y describen las tecnologías y herramientas empleadas a lo largo del proyecto.

3.1 Tecnologías

3.1.1 Lenguajes de programación

En el desarrollo del presente TFM ha sido necesario emplear los siguientes lenguajes de programación:

- **JavaScript (JS)**: Es el lenguaje de programación de alto nivel para HTML y Web. Junto con HTML y CSS, es una de las tres tecnologías que componen el núcleo de la ingeniería de contenido *World Wide Web*. [17] [18]
- **C Sharp (C#)**: Lenguaje de programación multi-paradigma, desarrollado por Microsoft con su iniciativa .NET. Es de propósito general y orientado a objetos. [19]
- **HTML (*Hypertext Markup Language*):** Lenguaje de construcción más básico para una página web que se emplea para crear y representar visualmente los elementos que la componen, sin incluir su funcionalidad [20].
- **CSS (*Cascading Style Sheets*):** Lenguaje de diseño gráfico que describe el estilo de un documento estructurado escrito en un lenguaje de marcado, generalmente HTML o XHTML [21] [22]. No sólo permite aplicar estilos visuales, sino que también permite otros tipos, como las hojas de estilo auditivas.

3.1.2 Xamarin

Xamarin es una tecnología que permite el desarrollo multiplataforma que permite cubrir todas las necesidades de una aplicación para *Smartphone* desde un único código base, que puede ser usado por los diferentes sistemas operativos, permitiendo ahorrar costes y tiempo de desarrollo [23]. Es propiedad de Microsoft desde el 2016 y está basado en C# [24]. Dentro de ella cabe destacar Xamarin.Forms, Xamarin.Android y Xamarin.iOS.

En primer lugar, Xamarin.Forms es un *framework*, basado en C#, que permite construir aplicaciones para *Smartphones*, creando código que se puede compartir entre los diferentes sistemas operativos (*cross platform*). No sólo permite crear la lógica de la aplicación, sino también la definición de interfaces [25]. Cuenta con varios controles y *layouts* que son renderizados a sus correspondientes nativos [26].

Por su parte, Xamarin.Android o Xamarin.Droid es la herramienta de la que dispone Xamarin.Forms para exponer el SDK de Android a los desarrolladores de .NET, con el fin de que puedan crear aplicaciones nativas de Android usando C# en Visual Studio en vez de Java en Android Studio [27].

Por último, similar a Xamarin.Android o Xamarin.Droid, Xamarin.iOS expone el SDK de iOS para los desarrolladores de .NET [28].

3.1.3 Lectura de códigos QR

Como uno de los requisitos era permitir lanzar la realidad aumentada en función de códigos QR, fueron necesarias las tecnologías que se describen a continuación.

Por un lado, los **códigos QR (*Quick Response Code*)** o código de respuesta rápida puede ser considerado como la evolución de un código de barras. Se trata de una matriz de puntos que almacena información, de forma que un *Smartphone* puede leerlo y decodificarlo empleando la cámara y un decodificador de QRs. Se caracteriza por permitir leer su contenido a alta velocidad y por tener tres cuadrados en las esquinas superiores e inferior izquierda, que permiten detectar su posición para así poder decodificarlo [29].

Por otro lado, **Jsqrcode**, una librería de JavaScript que permite leer códigos QR, captados por la cámara o desde una imagen almacenada. Está basado en el conocido decodificador de códigos QR ZXing (Zebra crossing), un decodificador open-source implementado en Java. En este caso, Jsqrcode está desarrollado por LazarSoft [30].

3.1.4 Tecnologías complementarias a la realidad aumentada

Como tecnologías complementarias a la realidad aumentada se encuentran dos, la cámara y los *heatmaps* o mapas de calor.

El uso de la **cámara** trasera de los *Smartphones* y *Tablet* ha sido esencial para el proyecto, puesto que es la forma que se tiene de interactuar con el mundo real y mostrar la realidad aumentada que corresponda. Así mismo, ha sido necesario su uso para poder escanear los códigos QR.

En cuanto a los ***Heatmaps* o mapas de calor**, son una forma de representación de datos, en los que se usa una termografía, es decir, se establece una jerarquía de dos polos (uno con colores cálidos y otro con fríos). Los puntos con colores cálidos son los que más atención pueden tener, mientras que los de colores fríos no son útiles [31].

Han sido usados para analizar las imágenes que se elegían como *tracker*, comprobando así las partes que no eran útiles y tratando de mejorarlas, con el fin de conseguir que la librería de realidad aumentada, Wikitude, reconociese más fácilmente dichos *tracker*.

3.1.5 Reconocimiento de voz

Como nuevo requisito surgió el de permitir que la aplicación reconociese la voz del usuario para poder interactuar con ella, pronunciando comandos. Fueron necesarias, para permitir el cumplimiento de este requisito, las tecnologías explicadas a continuación.

Microsoft Bing Speech API es la API de Microsoft que permite procesar y reconocer palabras pronunciadas por un ser humano. Permite procesar contenido procedente de un fichero de voz o de un discurso pronunciado en tiempo real. Está dentro de los Cognitive Services de Microsoft, que permiten, además de procesar lenguaje, habilitar características de visión, voz, lenguaje y conocimiento [32] [33].

Xamarin.Cognitive.BingSpeech y **Plugin.AudioRecorder** son dos paquetes de NuGet, el gestor de paquetes de .NET, desarrollados por el mismo usuario, Nate Rickard, que facilitaron el uso de la API de Bing Speech.

Por un lado, Xamarin.Cognitive.BingSpeech facilita la comunicación con dicha API, permitiendo usar todas sus funcionalidades. Sin embargo, está pensada para funcionar con Xamarin.Forms y diseñada para que funcione desde *Smartphones* y dispositivos móviles [34].

A su vez, si se requiere enviar a la API voz en tiempo real, es necesario trabajar con el micrófono del dispositivo y enviar en *streaming*. Por ello, el paquete Xamarin.Cognitive.BingSpeech está pensado para ser usado junto con el paquete

Plugin.AudioRecorder (aunque se puede emplear cualquiera), y por eso se decidió usarlo.
Por último, **Xam.Plugins.TextToSpeech**, es un paquete de NuGet, pensado para usar desde Xamarin.Forms, que permite pronunciar las palabras que se desee, indicando el idioma y estas.

3.2 Herramientas de desarrollo

3.2.1 Wikitude
Tecnología de realidad aumentada empleada para construir el motor. Se habla de ella en el ***Apartado 2.2. La tecnología elegida: Wikitude***.

3.2.2 Visual Studio
Visual Studio es un IDE (*Integrated Development Environment*) de Microsoft usado para el desarrollo de programas de ordenador, sitios web, aplicaciones web, servicios web y aplicaciones móviles. Permite integrar Xamarin, por lo que ha sido el IDE empleado [35].

3.2.3 Wikitude 3D Encoder
Herramienta para ordenador que permite optimizar los modelos 3D, para que puedan ser usados por Wikitude. Se encarga de transformar los modelos 3D, con formato FBX (.fbx), en el formato de Wikitude (.wt3), que se trata de un formato binario de compresión optimizado para la carga rápida de los modelos en *Smartphones* [36] [37].

3.2.4 Chrome DevTools
Es un conjunto de herramientas, integradas en Google Chrome, cuya finalidad es facilitar la creación y depuración de contenido web [38]. Al emplear varios lenguajes de programación dentro del proyecto, fue necesario emplear una herramienta de depuración externa a Visual Studio para depurar los lenguajes relacionados con la web, como JavaScript, ya que las herramientas de depuración de Visual Studio no lo permiten.

3.2.5 Safari Web Inspector
Similar a Chrome DevTools, se tratan de las herramientas integradas en el navegador de Safari que permiten la depuración de contenido web [39]. Fue empleada como herramienta de depuración externa a Visual Studio para depurar los lenguajes web en dispositivos iOS.

3.3 Herramientas de gestión de la configuración

3.3.1 Mopa
Herramienta web, implantada en la propia empresa, que permite la gestión de tareas y la imputación de estas en el parte de trabajo, llevando un seguimiento diario de estas.

3.3.2 Jira

Software, usado en la empresa, para rastreo de *bugs*, problemas y gestión de proyectos. Permite crear proyectos con tareas, estimaciones de tiempo, asignar responsables y empleados a cada tarea, crear nuevas tareas para la corrección de bugs, … [40]
Al igual que Mopa, cuenta con un servicio web.

3.3.3 TortoiseSVN

Cliente de Apache Subversion (SVN), implementado como una extensión de la *Shell* de Windows [41]. Se trata de la herramienta de control de versiones usada en la empresa, que permite tener repositorios para las aplicaciones que se están desarrollando, así como un control de las versiones y que varios empleados trabajen simultáneamente, asegurándose que las versiones subidas al repositorio tienen siempre los cambios correspondientes y necesarios.

4. Especificación de requisitos

Esta sección recoge los requisitos funcionales y no funcionales de la prueba de concepto finalmente desarrollada, ya que al tratarse de un proyecto de I+D, la lista de requisitos se fue incrementando a medida que se cumplía su implementación. Así mismo, algún requisito tuvo que ser cancelado porque la tecnología elegida no lo soportaba.

4.1 Captura de requisitos

La fase de captura de requisitos comenzó con el planteamiento de las prestaciones que se querían disponer en el producto. Estas se recogieron en alto nivel, como se describieron en los objetivos. A partir de ahí, se desarrolló la fase de estudio y análisis de herramientas y se detallaron y extendieron los requisitos, tras algunas reuniones con los responsables del proyecto dentro de la empresa. En todo momento se mantuvieron estas listas como abiertas, puesto que se permitían las inclusiones de nuevos requisitos según iba avanzando el proyecto.
Los listados definitivos de requisitos funcionales y no funcionales se detallan en los siguientes apartados.

4.2 Requisitos funcionales

Los requisitos funcionales son aquellos que establecen el comportamiento de un sistema, definiendo una función de este o de alguno de sus componentes. En la Tabla 4.1 se recogen todos los requisitos funcionales implementados, con un identificador y una pequeña descripción. El único requisito no cubierto ha sido la inclusión de vídeos ya que la tecnología no soportaba la posibilidad de cargar vídeos de un servidor o página web, solo del dispositivo.

Tabla 4.1 Requisitos funcionales

Identificador	Descripción
RF01	Permitirá escanear códigos QR
RF02	Permitirá mostrar realidad aumentada 1. Mostrará texto, imágenes, modelos 3D, vídeos, sinópticos, gráficas, contenido web, etiquetas con datos en tiempo real.
RF03	Permitirá ver datos en tiempo real
RF04	Contará con un botón para reproducir o detener las animaciones o la visualización de datos que se reciben en tiempo real
RF05	Contará con un botón para animar los modelos, permitiendo que giren sobre sí mismos
RF06	Permitirá usar el acelerómetro del *Smartphone* para mejorar la visualización de los modelos
RF07	Permitirá usar el giroscopio del *Smartphone* para mejorar la visualización de los modelos
RF08	Permitirá ver una tabla con datos leídos en tiempo real
RF09	Permitirá usar reconocimiento de voz para dar órdenes

RF10	Permitirá abrir una gráfica, con los datos leídos en tiempo real
RF11	Contará con un botón para fijar el modelo a la pantalla, de tal forma que una vez cargado en función del QR, pueda mostrarse sin necesidad de que siga apareciendo dicho QR.
RF12	Permitirá agrandar, reducir y rotar el modelo una vez se ha fijado a la pantalla.
RF13	Si el tamaño u orientación del modelo se modifica cuando se fija a la pantalla, si se vuelve a fijar sobre el QR, mantendrá las modificaciones sólo esa vez. Si vuelve a escanearse, volverá a tener el tamaño inicial.
RF14	Cuando el modelo es fijado a la pantalla y se modifica su tamaño, los botones que aparezcan también han de redimensionarse.

Debe señalarse que, los requisitos iniciales son los identificados con el RF01 hasta el RF07, incluidos; el resto fueron añadidos por iniciativa propia o sugeridos por la empresa.

4.3 Requisitos no funcionales

Los requisitos no funcionales, imponen restricciones en el diseño e implementación de una aplicación. Entre otros, hacen referencia a su rendimiento, disponibilidad, operatividad, escalabilidad, mantenibilidad, interfaz y seguridad.
Los requisitos no funcionales, junto con un identificador y el tipo al que hacen referencia, se muestran en la Tabla 4.2.

Tabla 4.2. Requisitos no funcionales

Identificador	Descripción	Tipo
RNF01	La *app* debe poder ejecutarse en Android e iOS	Portabilidad
RNF02	La *app* será programada usando Xamarin.Forms (Xamarin.Droid y Xamarin.iOS)	Portabilidad
RNF03	Los botones seguirán un estilo similar a los de la *app* de la empresa	Interfaz
RNF04	Deberá poder visualizarse en modo vertical y horizontal	Interfaz
RNF05	Debe ser fácilmente reutilizable	Mantenibilidad
RNF06	Servirá como modelo extensible de renderizado	Operatividad
RNF07	Será integrado en la *app* de la empresa	Portabilidad
RNF08	Debe ser fácilmente extensible	Extensibilidad
RNF09	Las librerías de AR deben ser, en la mayor medida posible, compatibles con Xamarin.Forms	Compatibilidad
RNF10	La *app* tendrá una estructura en tres capas	Mantenibilidad

5. Diseño de la aplicación

Esta sección se divide en dos apartados en los que se describe el diseño arquitectónico de la solución software y de cada componente de la aplicación.

5.1 Diseño arquitectónico

El diseño arquitectónico del motor de renderizado ha sido creado bajo las siguientes premisas: a) que sea fácilmente reutilizable y extensible y b) siguiera un patrón 3-capas.

- La capa de presentación: es el componente con el que interactúa el usuario. En este caso, la interfaz es muy sencilla. Muestra la visión de la cámara del móvil y los elementos de realidad aumentada que se superpongan en cada caso. Puede incluir botones para interactuar con los elementos y elementos textuales para aportar información al usuario.
- La lógica de negocio: es el componente que recoge las funcionalidades y procesos necesarios entre la capa de presentación y la persistencia. Dentro de esta se distinguen tres bloques:
    - Uno implementado en Xamarin con C# y que incluye código compartido entre los diferentes sistemas operativos.
    - Un bloque específico con funcionalidades propias para cada S.O., como puede ser el almacenamiento de ficheros. Estas funcionalidades están escritas también en C#.
    - Un bloque para construir la web que requiere Wikitude para poder crear los elementos de realidad aumentada. La mayor parte de la funcionalidad será recogida mediante JavaScript, por lo que podrá ser reutilizada, sin tener que realizar modificaciones, por cualquier sistema operativo.
- La capa de persistencia o datos: en un principio, no habrá almacenamiento de datos, ya que los elementos que se carguen serán elementos web o que estén en un servidor, por lo que solo será necesario que haya conexión a Internet. Sin embargo, como es un requisito que sea una arquitectura de tres capas y a medida que el proyecto avance puede ser necesario implementar una capa de persistencia, para mostrar información de los elementos si se pierde la conexión y fuera necesario trabajar offline, o para mostrar otro tipo de datos.

5.2 Diseño detallado

Tras la elección de la arquitectura, se comenzó con el diseño detallado, descomponiendo en unidades que luego serían traducidas a código.
En la Figura 5.1 se muestra el diagrama de despliegue de este proyecto:

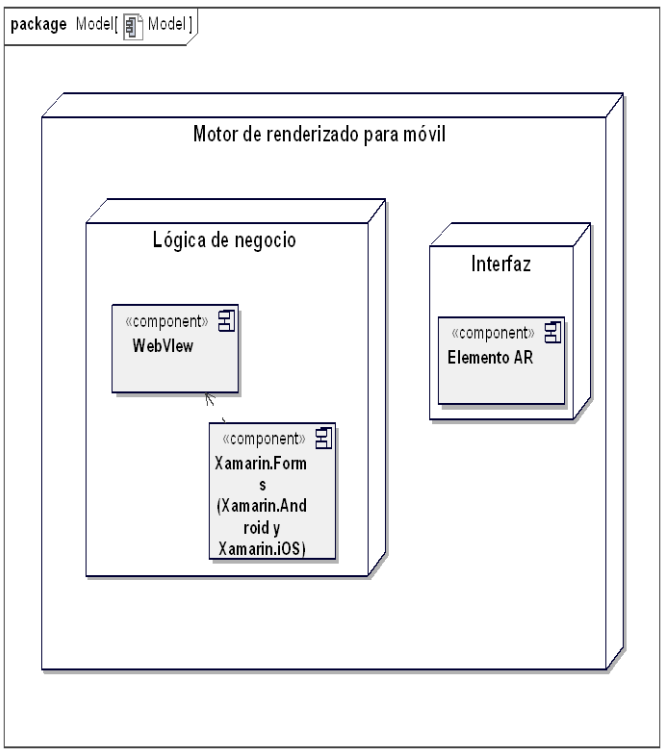

Figura 5.1. Diagrama de despliegue

6. Implementación

Una vez escogidos los requisitos y elegida una arquitectura que permita satisfacerlos, se procede a la implementación de los módulos necesarios, los cuales se explican en este capítulo.

6.1 Capa de negocio

La implementación de la capa de negocio ha sido desarrollada, principalmente, en JavaScript, aunque ha sido necesario también C# (Xamarin.Android y Xamarin.iOS), HTML y CSS.
El uso de todos estos lenguajes de programación ha sido fundamental, puesto que cada uno se ha encargado de dar soporte a un par de funciones, debido a la complejidad del problema a tratar. C# ha permitido crear *WebViews* desde Android o iOS. También ha permitido la captura de la pantalla, para poder decodificar la imagen posteriormente.
Como se comprueba en las Figuras 6.1, 6.2 y 6.3, no ha sido necesario crear multitud de clases. Dentro de Xamarin ha sido necesario crear una clase, *ARPage*, que es renderizada por otras dentro de Xamarin.Droid o Xamarin.iOS, según lo que corresponda.
Han sido necesarias también crear clases auxiliares para tomar una captura de pantalla y almacenarla, para que la librería de JavaScript pueda decodificarla. Esto ha sido necesario implementarlo en Xamarin.Droid y Xamarin.iOS, puesto que al ser dos sistemas operativos diferentes estas tareas se deben de hacer de forma diferente, no puede usarse un código común.

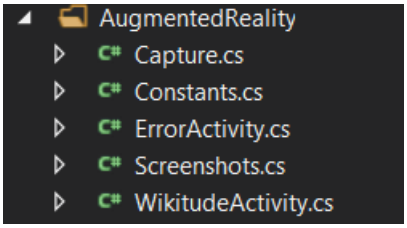

Figura 6.1. Clases creadas dentro de Xamarin.Droid

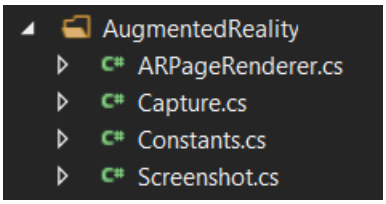

Figura 6.2. Clases creadas dentro de Xamarin.iOS

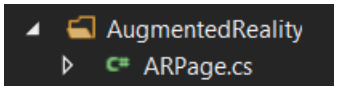

Figura 6.3. Clase creada dentro de Xamarin

Por su parte, el código HTML y CSS ha sido necesario para dar formato al *WebView* y permitir algunas funcionalidades, como la de fijar el elemento escaneado a la pantalla, y que se siga mostrando, aunque ya no se capte el código QR.
Por último, el código JavaScript ha permitido la decodificación de las imágenes capturadas, con el fin de encontrar códigos QR y decodificarlos. También ha sido posible la creación de todos los elementos de realidad aumentada de forma dinámica gracias a este lenguaje.
Como se aprecia en las Figuras 6.4 y 6.5, los elementos de código que da formato al *WebView* y funciona dentro de este son más numerosos, por desarrollarse la mayor parte de la complejidad aquí. Este código ha sido reutilizado en iOS y Android, puesto que no ha sido necesario modificar nada de él. Sí que se ha almacenado en diferentes rutas, estando dentro de *Assets* en Xamarin.Android y dentro de *Resources* en Xamarin.iOS, pero esto ha sido debido a la forma de operar de los sistemas operativos.

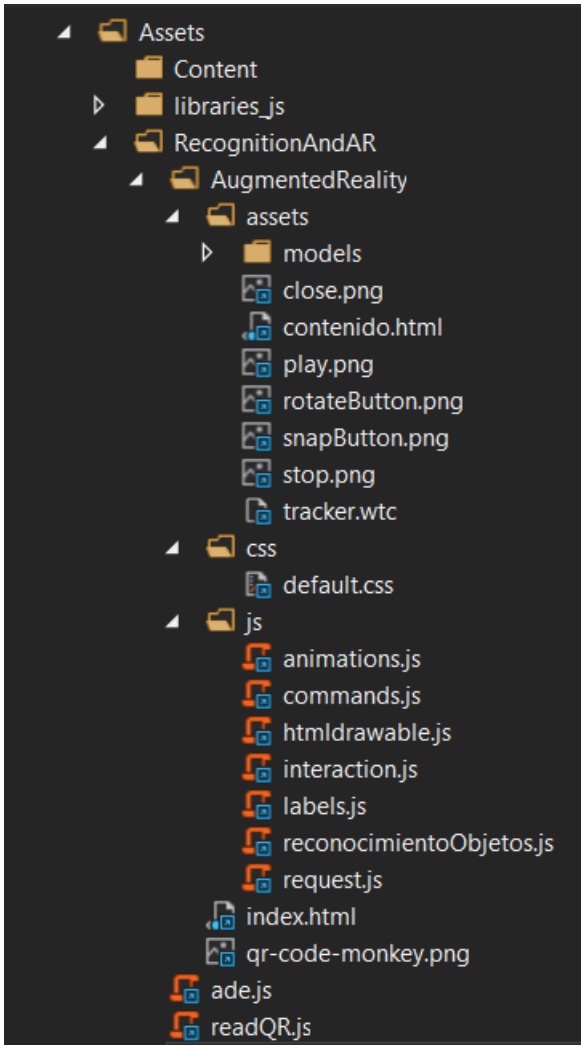

Figura 6.4. Código de la WebView en Android

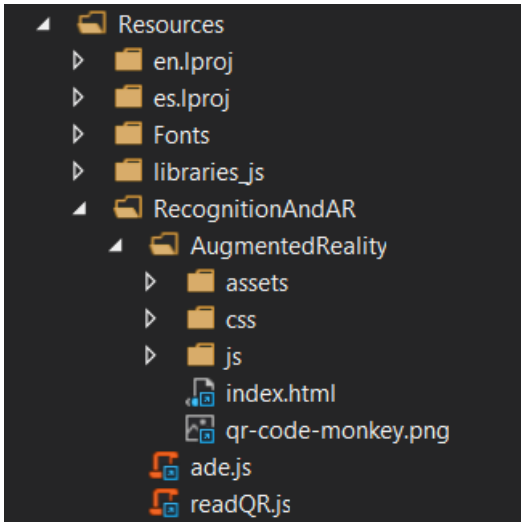

Figura 6.5. Código de la WebView en iOS

Dentro de los módulos de JavaScript, cabe destacar que *readQR* se encarga de realizar la comunicación entre la realización de la captura de pantalla y las librerías de decodificación de códigos QR, recibiendo el resultado y decidiendo qué hacer.
Por su parte, los módulos *htmldrawable*, *interaction* y *animations* permiten crear elementos de realidad aumentada 2D y 3D, interaccionar con ellos a través de botones y reproducir las animaciones en los modelos 3D que las posean, respectivamente.
Request y *labels* permiten conectarse con el servidor correspondiente para recibir los datos en tiempo real y crear una tabla con ellos y, por último, *commands* contiene las funciones que pueden ser activadas mediante reconocimiento de voz.
ReconocimientoObjetos es un nuevo módulo que se está incluyendo actualmente, con el fin de extender su funcionalidad para reconocer objetos 3D también, siendo una línea futura del motor de renderizado.
Por último, si se desea integrar el motor de renderizado dentro de una aplicación, basta con crear una instancia de *WikitudeActivity* en Xamarin.Android o de *ARPageRenderer* en iOS. Estas ya se encargan de pedir permisos al usuario y de instanciar todos los módulos disponibles.

6.2 Capa de presentación

Respecto a la capa de presentación, en este proyecto se trata de una capa muy sencilla, que cumple con los requisitos básicos para probar el componente.
Una vez la cámara detecta la imagen y se lee y decodifica el código QR, se muestra la realidad aumentada correspondiente. En cuanto a los casos de elementos mostrados en 2D y 3D no ha sido necesario darles formato, más que el tamaño. Sin embargo, sí que ha sido necesario trabajar con CSS, ya que se les ha añadido un botón que permite fijar el modelo a la pantalla, para así poder mostrarlo independientemente de que se esté leyendo un QR o no, y ha sido necesario crear un contenedor para ellos y darle formato mediante una hoja de estilos.

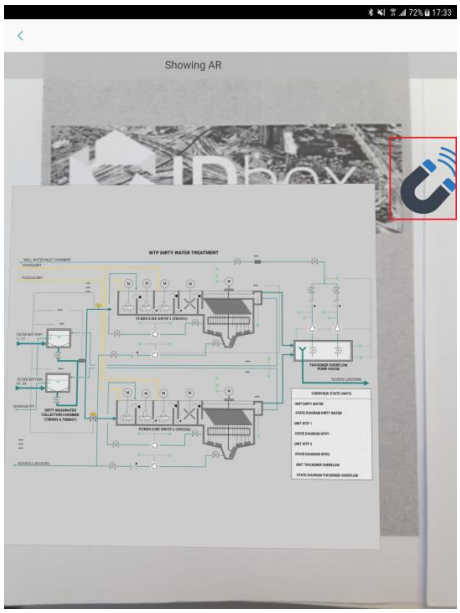

Figura 6.6. Modelo 2D con botón para fijarlo a la pantalla

Con el fin de facilitar el uso del motor por parte del usuario, se ha colocado una pequeña barra superior informativa con la tarea que se está realizando, ya sea escanear un QR, decodificando o mostrando la realidad aumentada. Esto ha sido implementado mediante JavaScript y HTML.

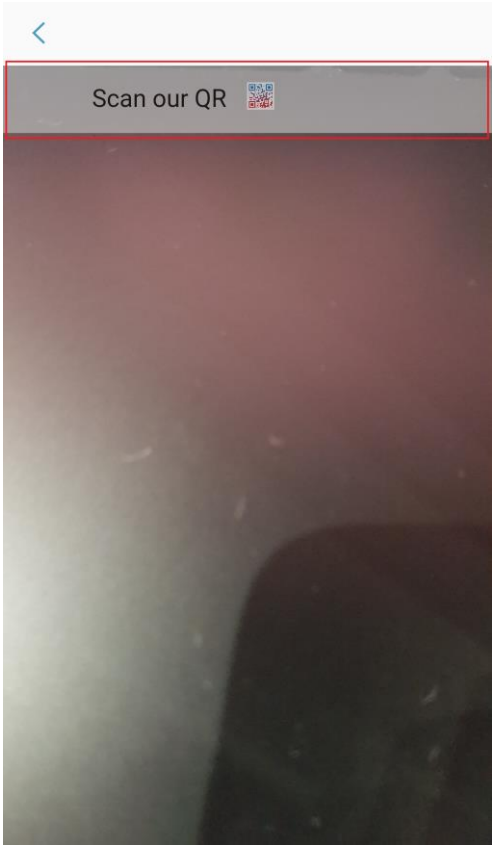

Figura 6.7. Mensaje superior

Por último, los botones que se muestran para interactuar con los modelos han sido creados también mediante JavaScript, y de igual modo, se les ha asignado su estilo.
Tanto en los modelos 2D como 3D se ha añadido un botón que permite fijar el modelo a la pantalla (enmarcado en rojo en la figura 6.8), para que aparezca indistintamente de si se sigue capturando con la pantalla el QR o no.

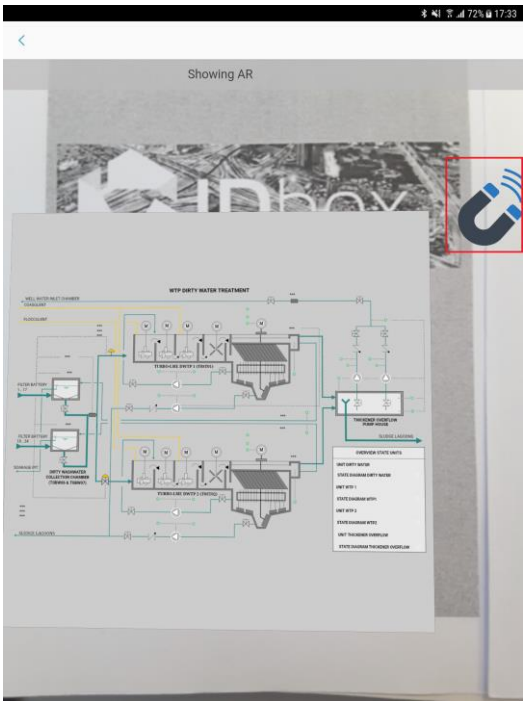

Figura 6.8. Botón para modelo 2D

Cuando se ha seleccionado este botón, se puede modificar el tamaño del modelo o incluso girarlo manualmente. Si se vuelve a presionar y se sigue capturando con la cámara el QR volverá a su lugar. Si se ha girado o modificado su tamaño se mantendrán estos cambios. En caso de que ya no se esté capturando el QR, desaparecerá.
En las siguientes figuras, 6.9, 6.10 y 6.11, podemos ver un modelo 3D escaneado a partir de un QR junto con este botón. La primera figura muestra simplemente el modelo al ser cargado en función del QR. La siguiente figura muestra el modelo una vez se ha seleccionado el botón, permitiendo que se quede fijado en la pantalla, aunque el QR ya no se está captando. Como se aprecia, el modelo ha sido girado por el usuario. En la última figura el modelo, además, ha sido agrandado por el usuario.

Figura 6.9. Botones para modelo 3D

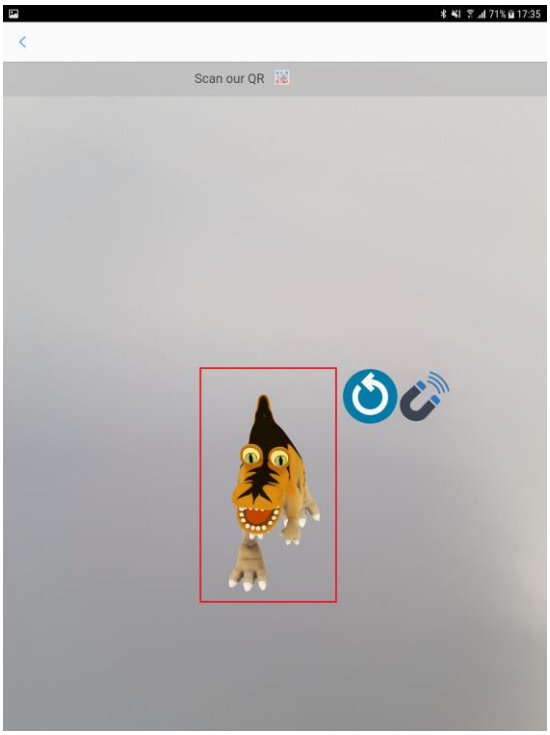

Figura 6.10. Modelo 3D tras haberse fijado a la pantalla y rotarse

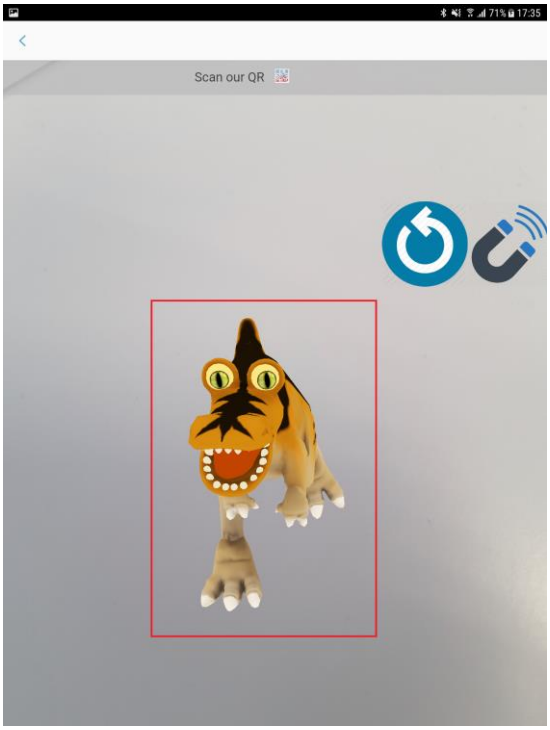

Figura 6.11. Modelo 3D aumentado de tamaño

Como se aprecia en las figuras, los modelos 3D también cuentan con otro botón, que hace que los modelos giren sobre sí mismos indefinidamente, o hasta que se vuelva a presionar.
Por último, se ha incluido la funcionalidad de permitir interactuar con los modelos 3D que disponen de animaciones. En un principio se pensó en incluir un botón que reprodujese las animaciones, pero más tarde se pensó que sería más útil si se activaban cuando el usuario presionase sobre el modelo. En caso de que haya varias animaciones, se activará una u otra en función de en qué parte del modelo presione el usuario. Estas pueden reproducirse indistintamente de si el modelo se está mostrando sobre el código QR o se ha fijado a la pantalla.

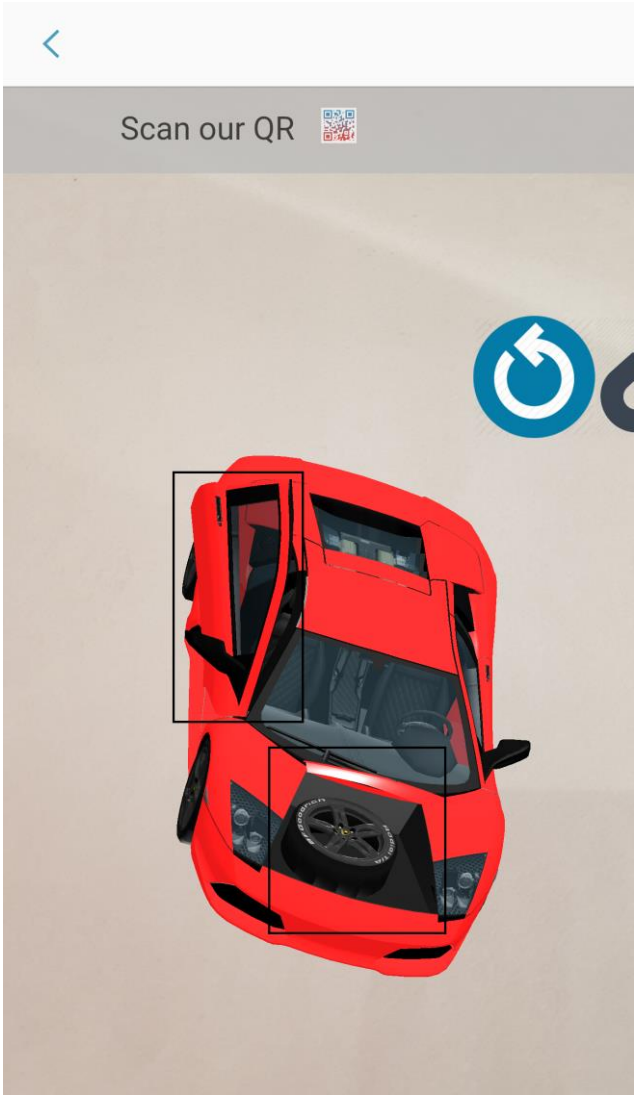

Figura 6.12. Modelo 3D con animaciones

6.3 Gestión de la configuración

La gestión de la configuración es un aspecto clave en cualquier empresa o institución dedicada al desarrollo de software, ya sea para tener una mejor organización y mantenimiento, un mejor control de los cambios y una forma segura de guardar todas las versiones.
Con estos fines, en este proyecto también se ha gestionado la configuración, empleando para ello TortoiseSVN, ya que es la herramienta usada dentro de la empresa.

7. Pruebas

Cualquier aplicación software no está libre de fallos, por lo que es necesario realizar pruebas con el fin de minimizar el número de fallos presentes. Hay varios tipos de pruebas, destacando las unitarias, de sistema, de aceptación y de integración (estas últimas se comentan en el ***Apartado 8.1. Pruebas de integración***).
Sin embargo, debido a que este proyecto empleaba varios lenguajes de programación, estas no han sido automatizadas siendo, las realizadas, de caja blanca, para comprobar los flujos de ejecución de las distintas unidades. Además, al tratarse de un prototipo, se ha preferido emplear el tiempo en estudiar las tecnologías y llevarlas a la práctica, más que en realizar una completa batería de pruebas.
Pero, por supuesto, estas no han sido obviadas en este proyecto. Sí que han sido llevadas a cabo, lo más completas posible, pero de forma manual (al fin y al cabo, la forma más simple y rápida de ver si se está mostrando la realidad aumentada correspondiente es así). Hay que destacar que las pruebas no han sido llevadas a cabo por una tercera persona, detalle que suele ser recomendable, puesto que es más fácil ver los errores de otra persona o pensar de forma diferente al programador y considerar casos de uso que él ha obviado. Además, se ha realizado un esfuerzo especial al emplearse varios dispositivos físicos, en lugar de emuladores, distinguiendo dos *Tablets* Samsung, un Samsung S7, un *Smartphone* Asus, dos iPhone y un iPad, para realizar las pruebas. Con ello, no sólo se ha comprobado y mejorado su funcionamiento, sino que se ha probado cómo se reproduce en varios tamaños y resoluciones de pantalla.

8. Integración en IDbox

IDbox es la herramienta de monitorización industrial SCADA desarrollada en la empresa [42], y cuenta con una aplicación para *Smartphones* que usen Android o iOS. La aplicación cuenta con un menú lateral con las opciones disponibles, por lo que se decidió añadir a este menú la opción de realidad aumentada, para permitir así la integración del motor de renderizado creado.

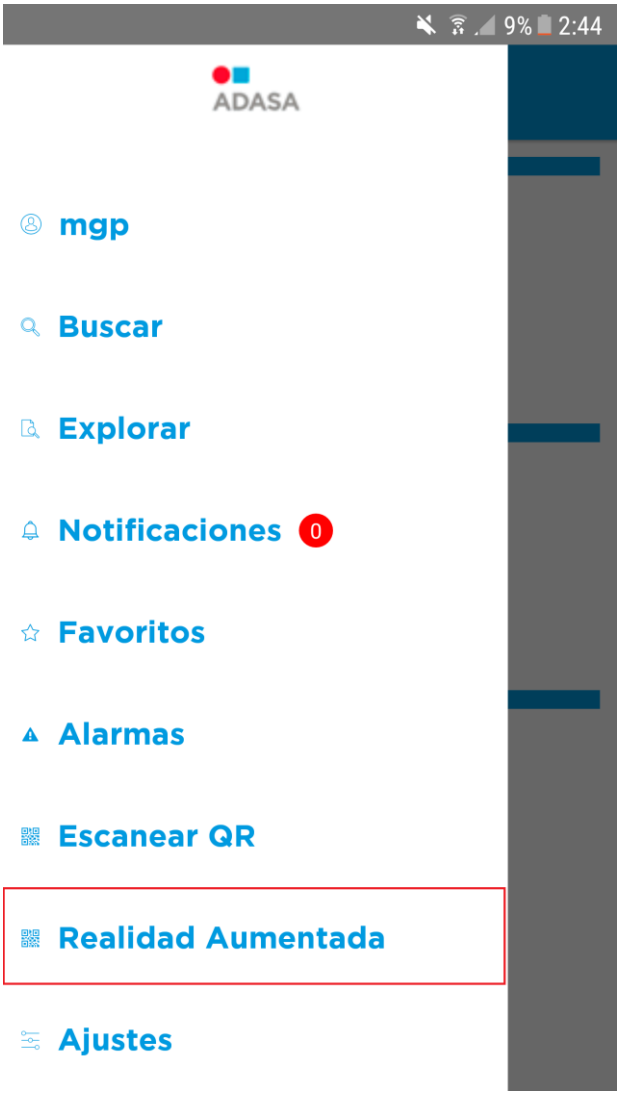

Figura 8.1. Opción de realidad aumentada desde el menú de IDbox Mobile

Para lanzar la realidad aumentada es necesario crear una Web (*WebView*) dentro de la aplicación, que en Android se lanza mediante una *Activity* y en iOS mediante una *View*, ya que son equivalentes en ambos sistemas operativos. En la Figura 8.2 se ve un resumen de ambos elementos, *Activity* y *View*, y las funciones que los componen. Estas han sido tenidas en cuenta por el motor de renderizado a la hora de mostrar la realidad aumentada, deteniendo por ejemplo los recursos y ejecuciones en *background* cuando sea necesario, para no repercutir negativamente en el rendimiento.

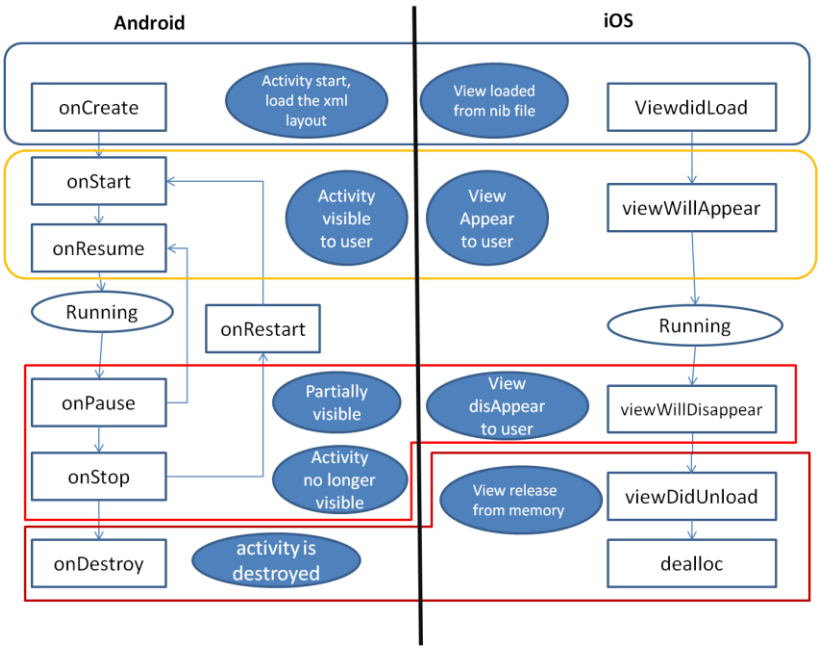

Figura 8.2. Módulos usados en ambas plataformas y comparativa

Al emplearse ambos elementos, ha sido necesario crear una *toolbar* o barra de navegación en la parte superior de la pantalla, para dar la opción de volver atrás, regresando a la pantalla principal de IDbox. En Android, además, también se ha dado dicha funcionalidad al botón físico de "atrás". Esta barra puede verse en las dos siguientes figuras y, como se puede apreciar, se muestra independientemente de la orientación del dispositivo.

Figura 8.3. Toolbar en Android

Figura 8.4. Toolbar en iOS

Cabe destacar que como se ha desarrollado todo usando una versión de prueba, la marca de agua ha podido ser eliminada en Android, pero no en iOS, de ahí que la mayoría de las capturas realizadas e incluidas en este documento sean con Android.
En cuanto a los QR usados, IDbox ya empleaba códigos QR para facilitar la carga de URLs de contenidos disponibles dentro de su aplicación, por lo que estos han sido reutilizados. En el caso de integrar el modelo en otra aplicación, bastaría con crear códigos QR con cualquier URI, ya sea una URL de un contenido web, de un modelo 3D en un servidor, o una URI correspondiente a un fichero almacenado en el dispositivo. En cuanto a las tablas con señales en tiempo real, no se ha creado en IDbox aún un protocolo para trabajar con ello, por lo que se estaban codificando los nombres de las señales como texto dentro del código QR para permitir una simulación.

8.1 Pruebas de integración

De igual forma que se ha comentado en el ***Apartado 7. Pruebas***, las pruebas de integración han sido realizadas manualmente y únicamente por la propia desarrolladora, una vez se logró integrar el motor dentro de la aplicación de IDbox.

9. Reconocimiento de voz

Una de las ideas iniciales de este proyecto era que se pudiese extender su funcionalidad empleando los sensores del dispositivo. Como se explica a continuación, el requisito inicial era emplear el giroscopio y el acelerómetro, pero la empresa prefirió abordar la incorporación de reconocimiento de voz.

9.1 Reconocimiento de voz (*Speech to text*) y Ayuda auditiva (*Text to speech*)

Al inicio del proyecto se planteó aportar nuevas características al motor, una vez implementadas las funcionalidades mínimas exigidas (requisitos RF01 al RF07). Entre ellas se encontraba el uso del giroscopio y del acelerómetro del *Smartphone* o *Tablet*, uniendo ambas características para trabajar con el modelo mostrado. Con su uso se pretendía que el modelo permaneciese estático, aunque el usuario girase alrededor de él, o que se aumentase o redujese respecto a la cercanía del usuario.
Sin embargo, al fijar el elemento a un punto del espacio, en función de una imagen o "*target*" que reconoce la librería de Wikitude, no era necesario programar estas características, puesto que las realiza por defecto la herramienta.
La empresa consideró entonces interesante la incorporación de reconocimiento de voz dentro del motor de realidad aumentada, por lo que se cedió el tiempo del proyecto destinado a trabajar con el giroscopio y el acelerómetro al reconocimiento de voz. La aplicación de esta sería con el objeto de facilitar el control sobre los elementos de realidad aumentada que se muestran en la pantalla, en especial en la funcionalidad con la que se muestran datos en tiempo real.
En un principio se pensó incluir un botón, creado con realidad aumentada, que permitiese grabar audio para procesarlo posteriormente y, con el tiempo, grabar sonido ininterrumpidamente. Sin embargo, se consiguió grabar sonido continuo desde el primer momento, por lo que se descartó el botón.
Es por ello que se han usado las librerías Xamarin.Cognitive.BingSpeech y Plugin.AudioRecorder para poder comunicarse con la API de Microsoft Cognitive Services Bing Speech, enviando a esta el audio grabado en tiempo real y recibiendo el texto correspondiente a este. De esta forma, en función del comando pronunciado, se llama a una u otra función. Esta característica, por el momento, sólo puede funcionar si se dispone de conexión a Internet. Por ahora se han implementado cuatro comandos de voz:

- Detener: detiene la actualización de los datos de la tabla, dejándose fijos los últimos datos recibidos.
- Reanudar: reanuda la actualización de los datos de la tabla, por lo que vuelven a verse en tiempo real.
- Abrir gráfica: abre una gráfica con las señales que aparecen en la tabla.
- Cerrar gráfica: cierra la gráfica.

Tanto en "abrir gráfica" como "cerrar gráfica" se ha codificado el comando de forma que permita cualquier palabra entre "abrir" o "cerrar" y "gráfica", por si el usuario pronuncia "abrir la gráfica" en lugar de "abrir gráfica".

Por el momento no se han implementado más comandos, pero en el futuro es posible que se añadan sinónimos o bien extenderlo con nuevas acciones.
Además, las librerías permiten reconocer lenguaje en varios idiomas, por lo que se puede leer el idioma de la aplicación de IDbox (si el motor está integrado) o del propio dispositivo móvil y se configuran con este idioma por defecto.
Sin embargo, cualquier persona que haya utilizado en alguna ocasión una aplicación con reconocimiento de voz puede comprender que esto es algo complejo, puesto que dependiendo de la entonación y acento del hablante o incluso del ruido que puede haber en los alrededores, pueden detectarse palabras similares, pero no iguales a las pronunciadas.
Es por ello que, como ayuda para el usuario, se decidió incorporar la reproducción de sonidos. Estos son reproducidos cuando se ha detectado un comando, de la forma siguiente:
Suponiendo que el usuario pronuncia el comando "Detener", la aplicación captura dicho comando y detiene los datos en tiempo real, a la vez que se escucha la frase "Deteniendo datos en tiempo real". De esta forma, el usuario reconoce que su comando ha sido ejecutado satisfactoriamente.
Esta funcionalidad puede no resultar útil, por ejemplo, al abrir o cerrar la gráfica, puesto que es algo perceptible por el usuario a simple vista, pero sí es muy útil a la hora de detener o reanudar los datos en tiempo real, ya que puede ser que se estén recibiendo valores de la señal iguales durante un cierto periodo de tiempo y el usuario no sepa realmente si se están recibiendo datos nuevos o no.
En el futuro, al incluir nuevas funcionalidades y comandos, puede resultar conveniente añadir esta facilidad.

9.2 Implementación

Para abordar la implementación de la nueva funcionalidad de reconocimiento de voz en el motor de renderizado se comenzó con un estudio de las tecnologías disponibles para Xamarin. Para ello se tuvieron en cuenta blogs y foros relacionados con el tema [43] [1] [2] [44] [45], así como páginas web de las casas de las propias tecnologías [46] [47].
En un principio los esfuerzos se enfocaron en librerías de código abierto y gratuitas, pero se encontraron dos grandes problemas: por un lado, algunas de las tecnologías eran demasiado novedosas y poco estables, contando incluso con días desde su publicación, como fueron Voice Recognition System for Android and iOS [48] y PocketSphinx [49], o parecían estar abandonadas y ya no darse soporte, como eran XLabs [50].
Se estudiaron entonces otras librerías gratuitas como Annyang y ACR Speech Recognition Plugin For Xamarin & Windows. Se realizaron pruebas con ellas, pero hubo problemas al tratar de integrarlas en Xamarin o con los componentes que ya empleaba el motor de renderizado, por lo que se descartaron y se recurrió a las librerías de pago como Bing Speech API, de Microsoft, y Google Cloud Speech-to-text.
Del estudio, se comprobó que Google Cloud Speech-to-text contaba con librerías cliente para poder trabajar offline, pero no eran del todo definitivas y no se recomendaban para producción. Como entre los trabajos futuros estaba el de trabajar offline, se escogió Bing Speech API.
Para realizar la implementación de la funcionalidad de reconocimiento de voz se creó, inicialmente, un nuevo módulo en C# que se encargaba de iniciar la grabación con el micrófono del *Smartphone* y realizar el reconocimiento. Este se realizaba a la vez que se continuaba grabando, de forma que el *Smartphone* está continuamente escuchando, no

es necesario presionar un botón o realizar ninguna acción para ello. Para facilitar tanto la interacción con el micrófono como el grabado de sonido se emplearon dos librerías, Xamarin.Cognitive.BingSpeech y Plugin.AudioRecorder.
Fue necesario también crear un segundo módulo, esta vez en JavaScript, para recoger las funciones que podían ser accionadas por comandos de voz.
Posteriormente, se incluyó en el módulo escrito en C# la ayuda auditiva o Text-to-speech, de forma que al recibirse un comando por voz se narraba en voz alta el comando indicado, haciendo uso del altavoz del *Smartphone*, de forma que el usuario sepa en todo momento que se ha reconocido y ejecutado satisfactoriamente el comando. Para ello se hizo uso de la librería Xam.Plugins.TextToSpeech.
En cuanto a las pruebas, estas se han realizado de forma manual y utilizando la voz únicamente de la desarrolladora del módulo.
Por último, esta nueva funcionalidad, como quedaba integrada dentro del motor de renderizado, no supuso realizar ninguna acción en el apartado de integración del motor con la app.

10. Conclusiones y trabajos futuros

10.1 Conclusiones

Este proyecto ha tenido por objeto el estudio de las tecnologías existentes de realidad aumentada para desarrollar un motor de renderizado capaz de mostrar diferentes tipos de contenidos en función de la decodificación de códigos QR fácilmente integrable en aplicaciones móviles. Su fin último es el de poder emplear esta tecnología para facilitar las labores de mantenimiento en una industria, permitiendo ver en cada lugar señalado con un código QR, los datos más relevantes en tiempo real.

Se ha de señalar que este TFM ha logrado cumplir todos los requisitos funcionales iniciales especificados, así como algún otro que surgió a medida que el proyecto era desarrollado, como es el reconocimiento de voz para controlar las acciones del módulo de realidad aumentada construido.

Dentro de este proyecto se diferencian dos grandes apartados: (I, el trabajo de estudio e investigación y (II, la implementación del motor de renderizado y su integración en IDbox. En la primera parte se ha abordado la búsqueda y el estudio de librerías de lectura de códigos QR, realidad aumentada y reconocimiento de voz, tecnologías muy novedosas, y aún poco maduras que ha conducido a tener que buscar soluciones para su integración en la solución. La segunda parte, se trata de un desarrollo software que ha seguido todas las fases del ciclo de vida, construyendo una aplicación funcional y fácilmente extensible y mantenible.

En un proyecto de esta índole, los problemas han sido la tónica habitual del día a día, enfrentándose a nuevos lenguajes, tecnologías y herramientas poco maduras de las que aún no hay suficiente comunidad. Pero la constancia y la confianza en mí misma me permitieron avanzar y conseguir exitosamente mi objetivo.

Este trabajo, además, me ha servido para introducirme en el mundo del *Cross platform*, de la mano de Xamarin, aspecto que veo muy útil e interesante para desarrollar aplicaciones en varios sistemas operativos de forma rápida y fácilmente mantenible. También me ha permitido profundizar en la realidad aumentada, un tema que siempre me ha fascinado, debido a sus múltiples usos, desde temas de ocio hasta temas industriales, como se muestra en este proyecto. Además, al haberme permitido estudiar e investigar tanto acerca de ella, me he dado cuenta de todos los tipos de contenido que permite mostrar, desde modelos 3D, 2D, vídeos, … y cargados en función de un punto geoposicional, de un punto del espacio en función del usuario, de códigos, de imágenes, de objetos 3D, … Las posibilidades de aplicación de la realidad aumentada no parecen tener límite.

10.2 Trabajos futuros

El desarrollo de este proyecto ha logrado cumplir con los requisitos iniciales, así como aquellos que fueron surgiendo durante su desarrollo.
Sin embargo, como trabajos futuros, entre otros, pueden encontrarse:

- Mejora del rendimiento: la decodificación de códigos QR no es fluida, por lo que podría mejorarse empleando *threads* y/o paralelizando partes de la codificación o, incluso, creando una nueva librería que decodifique los códigos QR.
- Inclusión de otros tipos de códigos, como pueden ser códigos de barras.
- Aprovechamiento de BIM: podría emplearse los BIM (*Building Information Modeling*) de algún edificio dentro de la propia *app*. Por ejemplo, podrían mostrarse planos en 3D correspondientes a la habitación donde se está.
- Trabajar offline: por cuestiones de tiempo, no se ha realizado. El motor de realidad aumentada está preparado para trabajar sin conexión a Internet, ya que se pueden decodificar los códigos QR y mostrar la realidad aumentada sin requerir de conexión, pero los datos que carga sí la requieren. Sin embargo, aprovechando que la aplicación de IDbox puede trabajar offline, cargando gráficas, sinópticos y datos almacenados en una pequeña base de datos, podrían aprovecharse para cargarse si no se dispone de conexión a Internet.
- Por otro lado, por el momento, el reconocimiento de voz funciona con una API que requiere de conexión a Internet. Microsoft, sin embargo, permite descargarse librerías para trabajar offline, por lo que puede mejorarse esta característica también y funcionar offline en el futuro.
- Rediseño de los iconos: por cuestiones de tiempo, los iconos de los botones no han sido diseñados concienzudamente, sino que se han escogido unos que cumpliesen bien la metáfora de su funcionalidad.
- Mostrar la realidad aumentada en función de la geoposición: es un aspecto en el que se está trabajando en este momento, especialmente geoposicionamiento *indoor*, puesto que *outdoor* es fácil al emplear el GPS, pero *indoor* tiene más aplicaciones.
- Reconocimiento de objetos 3D: otro aspecto en el que también se está comenzando a trabajar. Reconocer objetos 3D y mostrar la realidad aumentada en función de ellos es muy útil también, pudiendo obtener de estos la información a cargar. Sin embargo, se están realizando pruebas de cuan exacto es este tipo de reconocimiento, para que, si se obtienen buenos resultados, se incluya en el motor. El problema por resolver sería también la diferenciación entre dos modelos iguales, a la hora de cargar los datos.

Referencias

[1] *ACR Speech Recognition Plugin for Xamarin & Windows* (01/09/2016). Recuperado el 18/04/2018
https://forums.xamarin.com/discussion/76341/acr-speech-recognition-plugin-for-xamarin-windows
[2] *How to implement voice to text* (01/08/2017). Recuperado el 18/04/2018
https://forums.xamarin.com/discussion/101308/how-to-implement-voice-to-text
[3] *Augmented Reality: Applications, Challenges and Future Trends* (s.f.). Recuperado el 01/02/2018
http://www.wseas.us/e-library/conferences/2014/Malaysia/ACACOS/ACACOS-29.pdf
[4] *How to Create an Augmented Reality Experience for Your Business* (s.f.). Recuperado el 01/02/2018
https://codetiburon.com/create-augmented-reality-experience-for-business/
[5] *Here's where – and why – Markerless Augmented Reality works* (24/02/2017). Recuperado el 01/02/2018
https://www.wikitude.com/blog-wikitude-markerless-augmented-reality/
[6] *Factors to Consider before Building the Augmented Reality (AR) Application* (03/06/2015). Recuperado el 01/02/2018
https://www.rapidvaluesolutions.com/tech_blog/factors-to-consider-before-building-the-augmented-reality-ar-application/
[7] *Best Tools for Building Augmented Reality Mobile Apps* (s.f.). Recuperado el 01/02/2018
https://www.upwork.com/hiring/for-clients/building-augmented-reality-mobile-apps/
[8] *Augmented Reality 101: all about the technology behind ARCore and ARKit* (01/09/2017). Recuperado el 01/02/2018
https://www.wikitude.com/blog-augmented-reality-google-arcore-arkit-apple/
[9] *Virtual Reality/Augmented Reality White Paper* (12/2017). Recuperado el 28/04/2018
http://www-file.huawei.com/-/media/CORPORATE/PDF/ilab/vr-ar-en.pdf
[10] *Profiles in Innovation, Virtual & Augmented Reality: Understanding the race for the next computing platform* (13/01/2016). Recuperado el 28/04/2018
http://www.goldmansachs.com/our-thinking/pages/technology-driving-innovation-folder/virtual-and-augmented-reality/report.pdf
[11] *PC Magazine* (17/02/2017). Recuperado el 12/03/2018.
http://latam.pcmag.com/sistemas-operativos-moviles/18490/news/el-996-del-mercado-movil-le-pertenece-a-android-y-ios
[12] *Después de tres años de desarrollo, Google cierra Project Tango para centrarse exclusivamente en ARCore* (15/12/2017). Recuperado el 01/02/2018.
https://www.xataka.com/tag/project-tango
[13] *Social Compare* (s.f.). Recuperado el 10/02/2018.
http://socialcompare.com/en/comparison/ar-frameworks-388frkga
[14] *Deep-Dive into the Wikitude SDK* (07/11/2016). Recuperado el 29/04/2018
https://www.slideshare.net/AugmentedWorldExpo/philipp-nagele-cto-wikitude-an-insider-deepdive-into-the-wikitude-sdk
[15] *¿Qué es Wearable? – Los dispositivos vestibles* (s.f.). Recuperado el 28/04/2018
http://www.dispositivoswearables.net/

[16] *Getting started* (s.f.). Recuperado el 01/02/2018
http://www.wikitude.com/external/doc/documentation/latest/epson/
[17] *JavaScript tutorial* (s.f.). Recuperado el 30/03/2018.
https://www.w3schools.com/js/default.asp
[18] *JavaScript* (s.f.). Recuperado el 30/03/2018.
https://en.wikipedia.org/wiki/JavaScript
[19] *C Sharp* (s.f.). Recuperado el 30/03/2018
https://en.wikipedia.org/wiki/C_Sharp_(programming_language)
[20] *HTML* (s.f.). Recuperado el 31/03/2018.
https://developer.mozilla.org/es/docs/Web/HTML
[21] *CSS Tutorial* (s.f.). Recuperado el 31/03/20118
https://www.w3schools.com/css/
[22] *Hoja de estilos en cascada* (s.f.). Recuperado el 31/03/2018
https://es.m.wikipedia.org/wiki/Hoja_de_estilos_en_cascada
[23] *Xamarin la gran candidata a ganadora del desarrollo de aplicaciones para móvil* (30/11/2015). Recuperado el 01/04/2018
https://www.xataka.com/aplicaciones/xamarin-la-gran-candidata-a-ganadora-del-desarrollo-de-aplicaciones-para-movil
[24] *Xamarin* (s.f.). Recuperado el 01/04/2018
https://en.wikipedia.org/wiki/Xamarin
[25] *Introduction to Xamarin.Forms* (s.f.). Recuperado el 01/04/2018.
https://www.edx.org/course/introduction-xamarin-forms-microsoft-dev215x-3
[26] *Xamarin.Forms – Write Once, Run Everywhere, AND Be Native?* (s.f.). Recuperado el 01/04/2018
https://www.hanselman.com/blog/XamarinFormsWriteOnceRunEverywhereANDBeNative.aspx
[27] *Xamarin.Android* (s.f.). Recuperado el 01/04/2018
https://docs.microsoft.com/en-us/xamarin/android/
[28] *Xamarin.iOS* (s.f.). Recuperado el 01/04/2018
https://docs.microsoft.com/en-us/xamarin/ios/
[29] *Código QR* (s.f.). Recuperado el 28/04/2018.
https://es.wikipedia.org/wiki/C%C3%B3digo_QR
[30] *Jsqrcode* (09/01/2014). Recuperado el 28/04/2018
https://github.com/LazarSoft/jsqrcode
[31] *Mapas de calor (Heatmaps): Qué son, para qué sirven y por qué usarlos* (08/09/2015). Recuperado el 01/04/2018
https://wanaleads.com/mapas-de-calor-heatmaps-para-optimizar-web/
[32] *Microsoft Cognitive Services API* (s.f.). Recuperado el 28/04/2018
https://azure.microsoft.com/es-es/pricing/details/cognitive-services/speech-api/
[33] *Microsoft Bing Speech API* (s.f.). Recuperado el 28/04/2018
https://azure.microsoft.com/en-us/services/cognitive-services/speech/
[34] *Xamarin.Cognitive.BingSpeech* (s.f). Recuperada el 28/04/2018
https://github.com/NateRickard/Xamarin.Cognitive.BingSpeech
[35] *Microsoft Visual Studio* (s.f.). Recuperado el 01/04/2018
https://en.wikipedia.org/wiki/Microsoft_Visual_Studio
[36] *Wikitude 3D Encoder* (s.f.). Recuperado el 01/04/2018
https://www.wikitude.com/products/wikitude-sdk-features/wikitude-3d-encoder/
[37] *Documentation Wikitude SDK Web* (s.f.). Recuperado el 01/04/2018
http://www.wikitude.com/external/doc/documentation/latest/htmlcss/encoder.html
[38] *Chrome DevTools* (s.f.). Recuperada el 27/04/2018

https://developers.google.com/web/tools/chrome-devtools/?hl=es
[39] *Web Development Tools* (s.f.). Recuperado el 17/05/2018
https://developer.apple.com/safari/tools/
[40] *Jira (Software)* (s.f.). Recuperado el 01/04/2018
https://en.wikipedia.org/wiki/Jira_(software)
[41] *TortoiseSVN* (s.f.). Recuperado el 01/04/2018
https://tortoisesvn.net/
[42] *IDbox – Data integration, processing and analysis. Industrial Monitoring SCADA* (s.f.). Recuperada el 01/04/2018
https://www.cic.es/en/idbox-monitorizacion-industrial-scada/
[43] *Best Voice Recognition Software* (s.f.). Recuperado el 18/04/2018
https://www.g2crowd.com/categories/voice-recognition
[44] *How to use 'speechrecognition' plugin?* (01/07/2018). Recuperado el 18/04/208
https://forums.xamarin.com/discussion/99975/how-to-use-speechrecognition-plugin
[45] *Top 20 NuGet speech Packages* (s.f.). Recuperado el 18/04/2018
http://nugetmusthaves.com/Tag/speech
[46] *Xamarin.Cognitive.BingSpeech NuGet* (s.f.). Recuperado el 18/04/2018
https://github.com/NateRickard/Xamarin.Cognitive.BingSpeech
[47] *Speech Recognition* (17/03/2017). Recuperado el 18/04/2018
https://docs.microsoft.com/en-us/xamarin/ios/platform/speech?tabs=vsmac
[48] *Voice Recognition system for Android and iOS* (s.f.). Recuperado el 18/04/2018
https://github.com/ihassantariq/VoiceRecognitionSystem
[49] *PocketSphinx* (s.f.). Recuperado el 18/04/2018
https://github.com/cmusphinx/pocketsphinx
[50] *XLabs* (s.f.). Recuperado el 18/04/2018
https://github.com/XLabs/Xamarin-Forms-Labs