UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TECNOLOGÍA ESPECÍFICA DE
TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

Diseño y Desarrollo de un Sistema de Realidad Aumentada para el Enriquecimiento Visual de Información Textual

Alberto González Sánchez

Junio, 2018

Diseño y Desarrollo de un Sistema de Realidad Aumentada para el Enriquecimiento Visual de Información Textual

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

Tecnologías y Sistemas de Información

TECNOLOGÍA ESPECÍFICA DE
TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

Diseño y Desarrollo de un Sistema de Realidad Aumentada para el Enriquecimiento Visual de Información Textual

Autor: Alberto González Sánchez

Director: Javier Alonso Albusac Jiménez

Junio, 2018

Alberto González Sánchez

Ciudad Real – Spain

E-mail: Alberto.Gonzalez7@alu.uclm.es
Teléfono: 660 239 441

TRIBUNAL:

Presidente:

Vocal:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE	VOCAL	SECRETARIO
Fdo.:	Fdo.:	Fdo.:

Resumen

Actualmente la información impresa en papel aún sigue siendo uno de los medios más utilizados en la comunicación pero tiene limitaciones que las nuevas tecnologías no tienen, como por ejemplo visualizando un objeto en papel carece de movimiento, sonido, etc. La combinación de las nuevas tecnologías con los libros de texto permitiría disfrutar de una experiencia enriquecida.

Una de las nuevas tecnologías que está en pleno crecimiento es la Realidad Aumentada, que gracias a los avances en *smartphones* permite que tenga infinidad de aplicaciones por si sola o complementando esta tecnología con otros medios.

Este proyecto pretende diseñar y desarrollar un sistema que permita al usuario tomar una fotografía del texto en papel y obtener una representación de lo que está tratando de entender en Realidad Aumentada, pudiendo así percibir desde diferentes ángulos y en movimiento el objeto 3D. Esta fusión del papel con el contenido digital en Realidad Aumentada supone una innovadora mejora en la experiencia visual sobre el texto en papel.

Con el fin de mejorar las posibilidades de encontrar un objeto 3D acorde al contexto de la información transmitida en papel, se ha diseñado también un sitio web colaborativo que permite a los usuarios compartir modelos 3D. Esta opción permite tener un amplio número de objetos 3D en un repositorio.

Abstract

Nowadays, written communication keeps being one of the most popular ways of transmitting information. However there are some limitations to this that only newer technologies can bring, like adding motion or sound. A combination of new technologies and traditional ones like text books, would offer a much richer experience for the user.

One of the most developed of these new technologies is Augmented Reality, which in combination with the updates in smartphones generates a wide range of applications, both on its own and in combination with other media.

The main goal of this project is to design and develop a system that allows the user to turn a portion of any given text into a set of 3D imaginery directly related to what that text is representing, just by taking a picture of it. This mix between both methods is an innovative improvement over the visual experience of plain paper.

In order to increase the chances of finding a suitable 3D object for what is represented in that portion of text, a collaborative website has also been designed that allows users to share 3D models. This option allows having a large number of 3D objects in a repository.

Agradecimientos

Me gustaría empezar agradeciendo a mis padres y a mi hermana por haber aguantado y apoyado siempre, sois una garantía de familia.

A mis amigos con los que empecé la carrera y a los nuevos que he conocido en ella, por los buenos momentos que hemos pasado.

A mis amigos de toda la vida, los del barrio y en especial Juan, que aunque ahora estemos más lejos no va a cambiar nada.

A mi loco amigo Acebal de Gijón, que he perdido la cuenta de las veces que me ha ayudado.

Gracias también a mi hermano italiano Giovanni, qué decir que no sepas.

A mi director de proyecto Javi, que ha demostrado que es un excelente profesor, implicándose y motivándome cuando más lo necesitaba y por supuesto compartiendo conmigo todo lo que sabía.

Y, sobre todo, a Marta por ayudarme, apoyarme, entenderme y subirme los ánimos cuando más lo he necesitado. No hay suficientes agradecimientos para tí ;).

Alberto

A mi familia y a mi pareja

Índice general

Resumen III

Abstract IV

Agradecimientos V

Índice general VII

Índice de cuadros X

Índice de figuras XII

Índice de listados XV

Listado de acrónimos XVII

1. Introducción 1

1.1. Qué es y cómo nos afecta la Realidad Aumentada 2

1.2. Foco principal del Proyecto 3

1.3. Estructura del documento 6

2. Objetivos 8

2.1. Objetivo general 8

2.2. Objetivos parciales 8

2.2.1. Aplicación Móvil 8

2.2.2. Aplicación Web 8

2.2.3. Servidor Gestor de Objetos 3D 9

2.3. Objetivos docentes 9

3. Antecedentes o Estado del Arte 11

3.1. Desarrollo Web 11

3.1.1. Análisis de los lenguajes de programación 12

3.1.2. Análisis de los *frameworks* 16
3.1.3. Gestor de Base de Datos 22
3.2. Reconocimiento óptico de caracteres (OCR) 32
3.2.1. Tesseract Android Tools 33
3.2.2. ABBYY Mobile OCR Engine 34
3.2.3. Asprise OCR 34
3.2.4. Google Cloud - Vision API 35
3.2.5. IBM Cloud - Visual Recognition API 35
3.3. Procesamiento del lenguaje natural 35
3.3.1. Google Cloud Natural Language API 36
3.3.2. IBM - Watson Natural Language Understanding 37
3.3.3. Apache OpenNLP 37
3.3.4. Stanford CoreNLP 38
3.3.5. ParallelsDots - Text Analysis API 38
3.3.6. MAchine Learning for LanguagE Toolkit (MALLET) 39
3.3.7. Natural Language Toolkit (NLTK) 39
3.4. Comunicación Cliente-Servidor(*Web Services*) 39
3.4.1. Tipos de *Web Services* 40
3.4.2. Comparación REST vs SOAP 42
3.4.3. RESTful WS - JAX-RS 43
3.4.4. Herramienta de Testing - Advanced REST Client 47
3.5. Realidad Aumentada 48
3.5.1. Criterios 49
3.5.2. Listado de SDKs 49
3.6. Entorno de Desarrollo Integrado 55
3.7. Herramientas de Gestión del proyectos 56
3.8. Elaboración de documentos 57
3.9. Metodología ágil - Scrum 58
3.9.1. Beneficios de utilizar Scrum 58
3.9.2. Manifiesto Ágil 59
3.9.3. Roles y responsabilidades 61
3.9.4. Artefactos de Scrum 62
3.9.5. Qué es un Sprint 64

4. Método de trabajo **68**
4.1. Arquitectura del sistema 68

4.1.1. Cliente Android 70
4.1.2. Cliente Web 89
4.1.3. Servidor - Algoritmo de Selección de Obj 3D y SGBD 98
4.1.4. Comunicación 113
4.1.5. Patrones de diseño 117
4.2. Desarrollo de la Metodología Ágil 122
4.2.1. Roles en el proyecto 122
4.2.2. Product Backlog 122
4.2.3. Planificación temporal del proyecto 123
4.2.4. Marco Tecnológico 125
4.2.5. Coste estimado del proyecto 126
4.2.6. Sprint 0 126
4.2.7. Sprint 1 128
4.2.8. Sprint 2 130
4.2.9. Sprint 3 132
4.2.10. Sprint 4 133
4.2.11. Sprint 5 134
4.2.12. Sprint 6 135

5. Resultados **137**
5.1. Cámara y OCR 137
5.2. Procesamiento del Lenguaje Natural 139
5.3. Algoritmo de Selección de Objeto 3D 142
5.4. Realidad Aumentada 150

6. Conclusiones y trabajo futuro **158**
6.1. Objetivos alcanzados 158
6.2. Aportación personal 160
6.3. Trabajo futuro 161

A. Otros resultados **163**
A.1. SGBD 163
A.2. Comunicación 165

Referencias **168**

Índice de cuadros

3.1. Listado de Frameworks para *back-end*. 15
3.2. Ventajas y desventajas de Microsoft Access. 27
3.3. Comparación REST vs SOAP 43

4.1. Product Backlog 123
4.2. Tabla de costes estimados del proyecto. 126
4.3. Historia de Usuario 1. 127
4.4. Historia de Usuario 2. 127
4.5. Historia de Usuario 3. 127
4.6. Historia de Usuario 4. 128
4.7. Historia de Usuario 5. 129
4.8. Historia de Usuario 6. 130
4.9. Historia de Usuario 7. 130
4.10. Historia de Usuario 8. 131
4.11. Historia de Usuario 9. 131
4.12. Historia de Usuario 10. 132
4.13. Historia de Usuario 11. 133
4.14. Historia de Usuario 12. 133
4.15. Historia de Usuario 13. 134
4.16. Historia de Usuario 14. 135
4.17. Historia de Usuario 15. 135
4.18. Historia de Usuario 16. 136

5.1. Resultados OCR. 138
5.2. Resultados OCR - Calidad de la imagen. 138
5.3. Resultados Caso 1. 140
5.4. Resultados Caso 2. 141
5.5. Resultados Caso 3. 142
5.6. Resultados Algoritmo Selección Objeto 3D - Caso 1. 144

5.7. Resultados Algoritmo Selección Objeto 3D - Caso 2. 147
5.8. Resultados Algoritmo Selección Objeto 3D - Caso 3. 149

Índice de figuras

1.1. Ejemplos de Realidad Aumentada. 2
1.2. Niño usando una aplicación de Realidad Aumentada. 3
1.3. Ejemplo simplificado del sistema - Inicio. 4
1.4. Esquema general simplificado del sistema. 5
1.5. Aplicaciones de la Realidad Aumentada 6

3.1. Top 10 Lenguajes de programación del índice de TIOBE para mayo de 2018. 14
3.2. Lenguajes de programación que utilizan los sitios web más populares. . . . 15
3.3. Arquitectura MVC. 18
3.4. Evolución tecnológica de ASP.NET Web Forms. 18
3.5. Ejemplo de aplicación genérica de Ruby on Rails. 21
3.6. Ranking de SGBD acorde a su popularidad. 23
3.7. Modelo de red (izquierda) y modelo jerárquico (derecha). 24
3.8. Aplicación genérica de base de datos usando la libreria de SQLite. 31
3.9. Diagrama de un sistema de Reconocimiento Óptico de Caracteres. 33
3.10. Arquitectura típica de un servicio web basado en SOAP. 40
3.11. Estructura de un mensaje SOAP. 41
3.12. Diagrama de diversidad de sistemas que puede ofrecer una empresa con información comercial. 41
3.13. Diagrama ilustrativo de la vista arquitectural de alto nivel de un sistema RESTful. 44
3.14. Interfaz de usuario de ARC. 48
3.15. Continuidad Realidad-Virtualidad 48
3.16. Estimación de dispositivos de RA comprados e ingresos (en dólares) para el año 2021. 49
3.17. Esquema general de la arquitectura Wikitude Cloud 51
3.18. Comparativa de la compatibilidad en dispositivos de Wikitude con otros SDKs 51
3.19. Eclipse. 56
3.20. Eclipse. 56

3.21. Trello. 56
3.22. Ejemplo de un tablero en Trello. 57
3.23. Bitbucket. 57
3.24. TeXstudio. 57
3.25. Lucidchart. 58
3.26. Ejemplo de un diagrama en Lucidchart 58
3.27. Comparación de un Método tradicional en cascada vs Método Ágil. 60
3.28. Descripción general y cómo se relacionan los roles centrales del Equipo Scrum. 61
3.29. Flujo de eventos de un Sprint de Scrum. 65

4.1. Arquitectura general detallada del sistema. 69
4.2. Módulo OCR del Cliente Android. 71
4.3. Interfaz cámara - Rectángulo ajustable de selección. 72
4.4. Entrada y Salida del Módulo OCR. 73
4.5. Ejemplo de un resultado obtenido mediante OCR. 76
4.6. Módulo PLN del Cliente Android. 77
4.7. E/S del Módulo PLN en el Cliente Android. 77
4.8. Ejemplo listado de palabras clave y su factor de importancia. 79
4.9. Ejemplo listado de palabras clave y su factor de importancia. 80
4.10. Gestión de peticiones y respuestas del Cliente Android. 81
4.11. Módulo de Realidad Aumentada del Cliente Android. 82
4.12. Ejemplo HUD. Mensaje de ayuda y botón de la cámara. 88
4.13. Esquema Cliente Web. 90
4.14. Cabecera con usuario logeado. 93
4.15. Cabecera login - Usuario no logeado. 93
4.16. Vista para insertar modelos 3D en el repositorio. 95
4.17. Visualización de perfil del objeto 3D. 96
4.18. Eliminación de perfil del objeto 3D. 97
4.19. Visualización del repositorio. 97
4.20. Funcionalidad de puntuaje y *likes*. 98
4.21. Diagrama de flujo - Algoritmo de Selección de Objeto 3D. 100
4.22. Ejemplo de obtención del valor de $\boldsymbol{n}$. 103
4.23. Ejemplo de valores de la tabla *Item* en la base de datos. 108
4.24. Diagrama de tablas de la base de datos. 110
4.25. Listado de clases del SGBD. 112
4.26. Comunicación Cliente a Servidor - Petición. 114

4.27. Comunicación Servidor a Cliente - Respuesta. 116
4.28. Manejo de *request* del cliente. 118
4.29. Modelo de sincronización Rendezvous. 119
4.30. Sincronización Cliente-Servidor. 120
4.31. Diagrama de interacción para realizar una operación *delete*. 121
4.32. Modelo-Vista-Controlador. 122
4.33. Diagrama de Gantt del TFG. 124

5.1. Tabla *items* para las pruebas del Caso 1. 143
5.2. Tabla *keywords* para las pruebas del Caso 1. 143
5.3. Tabla *items* para las pruebas del Caso 2. 145
5.4. Tabla *keywords* para las pruebas del Caso 2. 146
5.5. Tabla *items* para las pruebas del Caso 3. 147
5.6. Tabla *keywords* para las pruebas del Caso 3. 148
5.7. Superficies diferentes. 151
5.8. Cambio de posición por rotación. 151
5.9. Cambio de posición por desplazamiento. 152
5.10. Cambio de tamaño. 153
5.11. Cambio de ángulo. 154
5.12. Superficies diferentes. 155
5.13. Cambio de posición por rotación. 156
5.14. Cambio de tamaño. 156
5.15. Cambio de ángulo. 157

A.1. Ejemplo tabla *items*. 164
A.2. Inserción de palabra clave en la tabla *keywords*. 165
A.3. Contraseñas encriptadas en el SGBD. 165
A.4. Contenido del cuerpo de la *request*. 166
A.5. Resultado de la petición. 167

Índice de listados

4.1. Ejemplo de código para dibujar la parte exterior del rectángulo 72
4.2. Dependencias de la libreria de OCR 74
4.3. Permisos de Internet en el *manifest* 74
4.4. Método encargado del reconocimiento óptico de caracteres 75
4.5. Definiendo la petición al servidor 78
4.6. Invocación del servicio 78
4.7. Extracción de datos de la respuesta 79
4.8. Ejemplo de pares obtenidos en el módulo de PLN 80
4.9. Uso de la librería Gson para pasar *HashMap* a *String* 81
4.10. Descomprimiendo respuesta y preparando directorio 81
4.11. Recibiendo el directorio en la activity de RA 82
4.12. Parámetros de configuración de *ViroViewARCore* 83
4.13. Método `displayScene` para la inicialización de la escena 83
4.14. Método `initARCrosshair` para la inicialización del *crosshair* 84
4.15. Método `init3DModelProduct` para cargar el objeto 3D 85
4.16. Estableciendo la iluminación de entorno 85
4.17. Captura foto y menú para compartir en otras aplicaciones 86
4.18. Conjunto de constantes definidas para el estado. 86
4.19. Método `setTrackingStatus`. 87
4.20. Función `update3DModelProduct`. 88
4.21. Método para utilizar las animaciones de un objeto 3D. 89
4.22. Clase *SessionUtils*. 91
4.23. Inserción de componente *validator* de JSF. 91
4.24. Componente *validator* de JSF. 92
4.25. Inserción de cabecera en las vistas JSF. 93
4.26. Inserción de cabecera en las vistas JSF. 93
4.27. Método `logout` - Invalidar sesión. 94
4.28. Ejemplo del listado de entrada que recibe el Algoritmo. 99

4.29. Transformación del listado. 101
4.30. Ordenación descendente. 101
4.31. Método `busquedaArrayCoincidencias`. 102
4.32. Array de coincidencias vacio. 102
4.33. Máximo número de coincidencias. 104
4.34. Procedimiento almacenado `insertarUsuario`. 111
4.35. Método constructor de la clase *DataSourceService*. 113
4.36. Ejemplo de conexión con la base de datos. 113
4.37. URL del servicio en web.xml. 115
4.38. Clase *MessageApplication* 118
4.39. Definición del *servlet* en el descriptor 118
A.1. Método `selectSumaPuntuaciones` de la clase *DAOItems* 163
A.2. Método `insert` de la clase *DAOKeywords* 164
A.3. Instrucción del *script* que lista los usuarios del SGBD 165

Listado de acrónimos

RPC	Remote Procedure Call
PLN	Procesamiento del Lenguaje Natural
JSF	JavaServer Faces
JSP	JavaServer Pages
API	Application Programming Interface
SGBD	Sistema Gestor de Bases de Datos
OCR	Optical Character Recognition
HTTP	Hypertext Transfer Protocol
TFG	Trabajo de Fin de Grado
REST	REpresentational State Transfer
XML	Extensible Markup Language
MVC	Modelo-Vista-Controlador
URL	Uniform Resource Locator
POO	Programación Orientada a Objetos
SOAP	Simple Object Access Protocol
SMTP	Simple Mail Transfer Protocol
JAX-RS	Java API for RESTful Web Services
POJO	Plain Old Java Object
SDK	Software Development Kit
IDE	Integrated Ddevelopment Enviroment
DAO	Data Access Object
FI	Factor de Importancia

Capítulo 1

Introducción

Los inicios de la comunicación fueron probablemente la era de los signos y las señales, que se desarrolló al comienzo de la prehistoria, anterior al lenguaje. Los antropólogos opinan que el hombre prehistórico entró en la era del habla y del lenguaje hace alrededor de 40.000 años. El comienzo de la era de la escritura data de hace 5.000 años. Esta herramienta de comunicación supuso un gran avance para el ser humano, ya que al principio se basaba en representaciones pictográficas que reflejaban las ideas y evolucionaron hasta llegar a ser letras que representaban sonidos específicos.

Hasta el siglo XV, los libros se difundían en copias manuscritas, que eran realizadas por amanuenses, de los cuales muchos eran monjes y frailes que se dedicaban a replicar ejemplares por encargo. En 1450 hay un punto de inflexión en la evolución de la comunicación, este logro fue la creación de la prensa de imprenta moderna por Johannes Gutemberg. Gracias a este descubrimiento han surgido los libros que ofrecen la posibilidad de transmitir la información hasta la actualidad, donde han aparecido otros medios de transmisión como la radio.

En la actualidad, la información impresa en papel aún sigue siendo uno de los medios más utilizados en la comunicación pero tiene limitaciones que otros medios modernos no tienen, como por ejemplo visualizando un objeto impreso en papel tienes la limitación de incorporar movimiento, sonido, hacer búsquedas textuales sobre el papel, facilidad para editar los contenidos, etc. Combinando los avances tecnológicos con los libros de texto se podría disfrutar de una experiencia de lectura enriquecedora.

Durante los últimos años, una de las nuevas tecnologías que está en auge es la Realidad Aumentada, esto es debido a los avances en los teléfonos inteligentes (*smartphones*) y tabletas que, cada vez mejor, facilitan una experiencia de la Realidad Aumentada en entornos dinámicos. Además, a partir del 2010, la tecnología *Wearable* es cuando ha evolucionado lo bastante para poder atraer un extenso abanico de consumidores. Con la aparición de las *Smart Glasses* se permite el acceso a información de Realidad Aumentada afectando mínimamente en las tareas que el usuario realiza. Estos avances en la Realidad Aumentada hace que tenga infinidad de aplicaciones por si sola o complementando esta tecnología con otros medios.

Teniendo en cuenta las carencias que tiene la comunicación en papel y la capacidad de complementarse de las nuevas tecnologías, la fusión del papel con los contenidos digitales sobre pantalla y la Realidad Aumentada pueden ser una solución muy útil de cara a la evolución de la comunicación impresa. La Realidad Aumentada permitiría, sobre todo, percibir visualmente lo que estamos tratando de entender, desde diferentes ángulos y en movimiento.

Por lo que, haciendo uso de un libro de texto clásico, el usuario podría ver representado, mediante Realidad Aumentada, un objeto 3D sobre la superficie que el usuario decida. Por ejemplo, en ingeniería mecánica las partes de un motor, al tener una gran libertad de movimiento, se puede percibir el objeto desde cualquier ángulo y así facilitar y mejorar el aprendizaje.

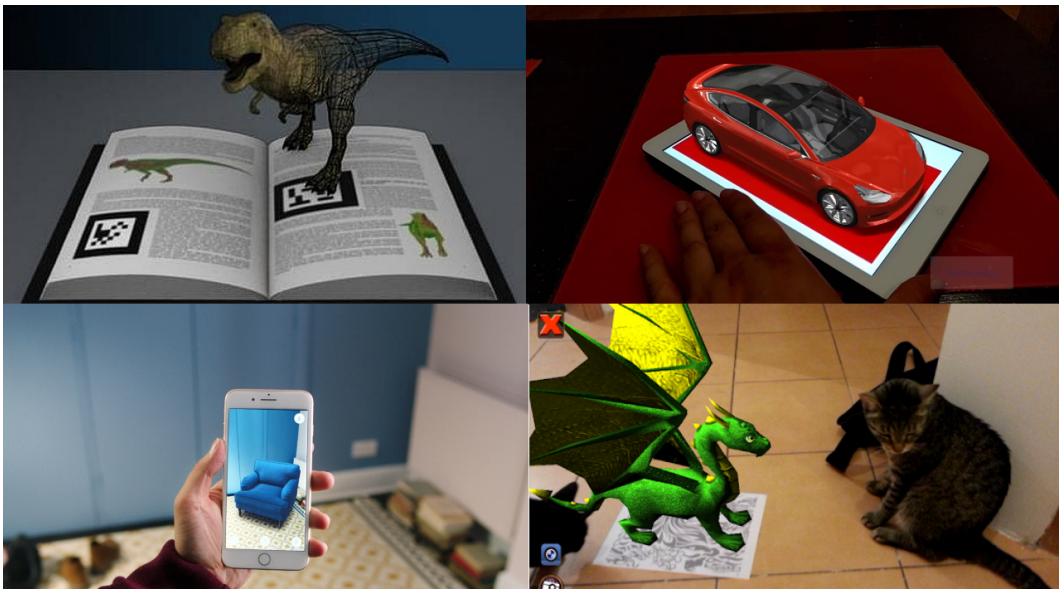


Figura 1.1: Ejemplos de Realidad Aumentada.

1.1 Qué es y cómo nos afecta la Realidad Aumentada

En primer lugar, para hacer más fácil la comprensión del documento, se va a definir lo que es la Realidad Aumentada. Una definición proporcionada por Azuma, es que «*la Realidad Aumentada se presenta como un conjunto de tecnologías que combinan lo real con lo virtual, de forma interactiva y en tiempo real, es decir, que da la opción de incluir e integrar a la perfección información virtual a la información física que el usuario está percibiendo de la realidad*». La realidad o mundo real es el principal elemento para diferenciar la Realidad Aumentada de la Realidad Virtual, ya que la Realidad Virtual sustituye la realidad física y la Realidad Aumentada utiliza esta realidad física para sobreponer esta información, por ejemplo, añadiendo e integrando perfectamente un objeto 3D con forma de lámpara sobre la superficie de una mesa. La Realidad Aumentada, a diferencia de la Realidad Virtual, no «lleva» al usuario a un mundo virtual creado por ordenador [Azu97].

Un sistema de Realidad Aumentada debe tener las siguientes características:

1. Percibe el mundo real de forma que lo puede combinar con el virtual, para ello el dispositivo dispone de algún elemento que capture las imágenes de la realidad que están viendo los usuarios. Como por ejemplo, un dispositivo móvil, tablet o *Smart Glasses* de Realidad Aumentada, en las que esté presente una cámara.
2. Se necesita un elemento sobre el que proyectar la combinación de imágenes reales con la virtuales, como puede ser la pantalla de un ordenador o *smartphone*. Además debe ser interactivo en tiempo real, es decir, debe generar la información virtual y mezclarla con la real en el

momento concreto que el servicio necesite. Esto determina el punto de madurez en el que se encuentra el dispositivo.

3. La información virtual debe estar correctamente alineada y posicionada con el mundo real. Por esto, los elementos de localización como los GPS que van integrados en la gran mayoría de los *smartphone*, además de las brújulas y acelerómetros, permiten determinar la posición y orientación del dispositivo con respecto al mundo real, y así se puede conocer cual es la posición correcta de la información virtual.

Haciendo un análisis de la evolución de la Realidad Aumentada con respecto a las expectativas creadas a principio de esta década, llegamos a la conclusión de que las expectativas han sido más elevadas que los desarrollos y aplicaciones actuales. Sin embargo, tener la posibilidad de poder contextualizar cierta información de manera visual, en movimiento y con una mínima intrusión en las tareas del usuario, garantiza el éxito de la Realidad Aumentada en sectores donde pueda ser empleada, como por ejemplo en medicina, aportando beneficios con su uso en quirófanos y entrenamiento de doctores; en fabricación, para mantenimiento y reparación de maquinaria, los pasos son mucho más intuitivos y fáciles de seguir si aparecen sobre la imagen real; en entretenimiento, esta industria ya está sacando partido del enorme potencial de la Realidad Aumentada gracias a su capacidad de interacción; en publicidad, por parte de las empresas hay gran tendencia a utilizar la Realidad Aumentada como herramienta de reclamo en su publicidad, por ejemplo, en la última década, empresas como Adidas, Ford, BMW, Lego, FOX, Paramount, Ikea, etc. han hecho uso de técnicas de este tipo [CGC12].

Figura 1.2: Niño usando una aplicación de Realidad Aumentada.

1.2 Foco principal del Proyecto

Debido a las limitaciones que supone transmitir información en papel, este medio tradicional podría combinarse con tecnologías actuales de Realidad Aumentada para enriquecer visualmente la información. Este proyecto ha surgido con la intención de aprovechar nuevas tecnologías, como son

los *smartphones* y la Realidad Aumentada, con el fin de mejorar la información que obtenemos de cualquier texto en papel.

Propuesta

El objetivo principal es enriquecer visualmente cualquier tipo texto, independientemente del tema tratado. Para ello, como podemos ver en la Figura 1.4, se desarrollará un sistema compuesto de tres partes claramente diferenciadas: aplicación para dispositivos móviles, aplicación web y sistema servidor.

Primero, mediante una aplicación para dispositivos móviles (ver Figura 1.3), el usuario realiza una fotografía del texto que está leyendo. Un procedimiento encargado de extraer texto permite digitalizar el texto a través de esta fotografía. Una vez extraído el texto del papel, el siguiente paso es obtener un listado con las palabras más importantes. Este paso se lleva a cabo con un procedimiento encargado del análisis del texto, que deduce mediante el entendimiento de lenguaje natural qué palabras son las más importantes. En ejemplo de la Figura 1.3, teniendo el texto «This is a top secret document» sobre el papel, se realiza la fotografía y tras realizar un análisis del texto se obtiene el listado de las palabras más importantes que son «Document» y «Secret».

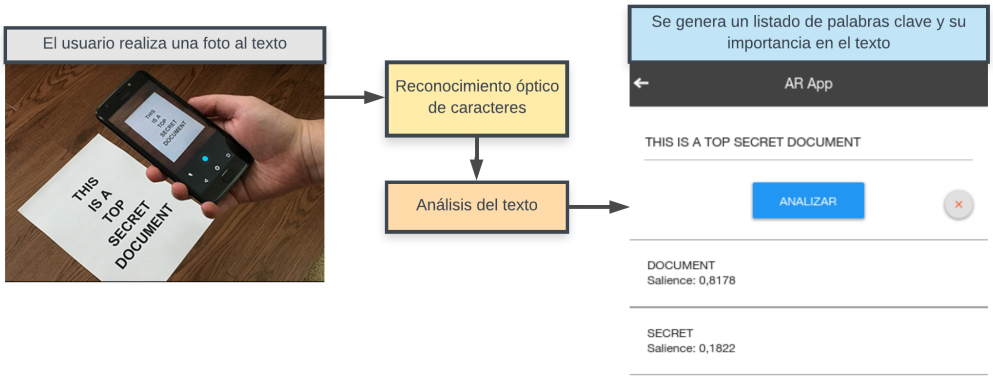


Figura 1.3: Ejemplo simplificado del sistema - Inicio.

En este punto entra en juego la parte del Servidor. La aplicación móvil realiza una comunicación con el Servidor, enviando la lista de palabras clave generadas con el análisis del texto. A continuación, la lógica del servidor se encargaría de procesar este listado, de manera que comprueba si existen objetos 3D acordes al tema del texto analizado, y en el caso de que exista más de un resultado, determina cual es más adecuado en base al sistema de votaciones de usuarios de la Aplicación Web, que a continuación se explicará. El resultado final, de la búsqueda en la base de datos del servidor y la selección del más adecuado es un único objeto 3D. El algoritmo de selección de objeto 3D está explicado de manera mucho más detallada en los siguientes capítulos. Basándonos en el ejemplo de la Figura 1.3, el servidor recibiría el listado de palabras más importantes, es decir, «Document» y «Secret», y en base a estas palabras buscaría objetos 3D que tengan relación.

Una vez generada la respuesta del servidor, haya encontrado o no resultado, se envía al dispositivo

móvil. En el caso de haber encontrado un resultado adecuado, entonces representa el objeto 3D mediante una librería integrada de Realidad Aumentada en la aplicación móvil, permitiendo al usuario tener información visual complementaria al texto y los beneficios que conlleva, como tener la libertad de visualizar el objeto 3D desde diferentes perspectivas y tamaños.

En la Figura 1.4 están representadas las tres partes del sistema.

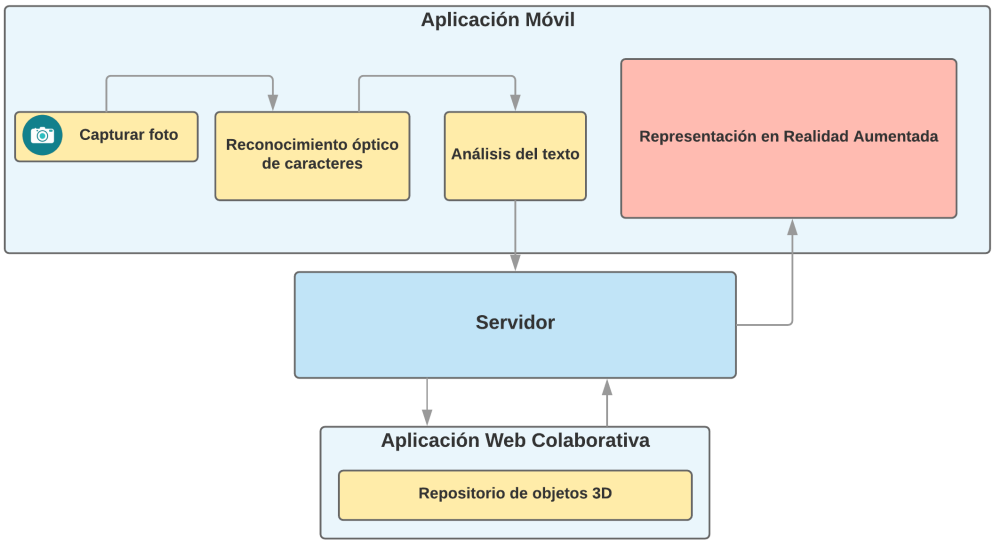


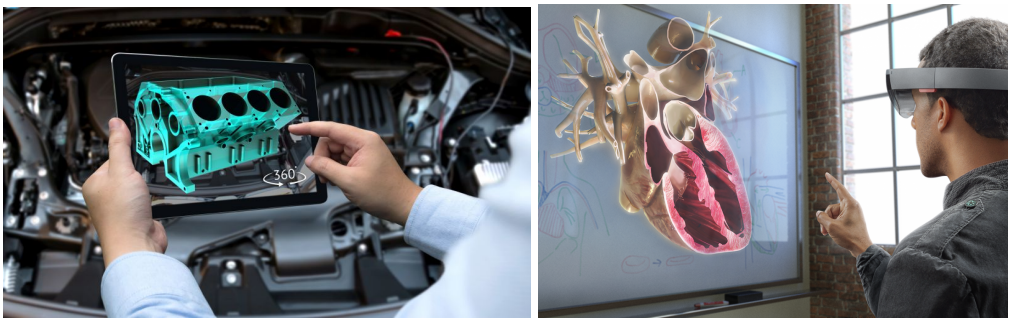
Figura 1.4: Esquema general simplificado del sistema.

Para satisfacer la necesidad de tener una amplia base de datos de contenidos, se ha desarrollado también una solución web colaborativa. Una simple aplicación para dispositivos móviles con algunos objetos estaría limitada, pero con esta solución se amplía la base de conocimientos, es un repositorio donde hay objetos de múltiples temáticas y contenidos. Mediante esta aplicación web el usuario puede compartir objetos 3D junto con características descriptivas obligatorias, como las palabras clave, y opcionales, como la descripción o fotos. En esta aplicación web, además de poder colaborar compartiendo objetos 3D, el usuario puede tener su propio perfil, guardar como favoritos los objetos que desee, puntuar otros objetos 3D y buscar y visualizar el perfil de objetos 3D subidos por los otros usuarios.

Beneficios y Aportaciones

El usuario, mediante una aplicación móvil, tendría la opción de realizar una foto al texto que está leyendo y obtener una representación en Realidad Aumentada, de un objeto 3D acorde al contexto, de una manera rápida y sencilla. Lo que podría ayudar al usuario a tener un cierto enriquecimiento visual del texto en papel, gracias a la Realidad Aumentada. Por ejemplo, si el usuario estuviera leyendo en un libro de texto la definición de «Sistema Solar», podría realizar una foto y si en la base de datos existe un objeto 3D acorde a este tema, podría verlo en Realidad Aumentada encima de la superficie que elija e incluso, si este objeto 3D lo permite, verlo en movimiento pudiendo así apreciar la traslación los planetas en su órbita, la rotación, su posición, etc. Para un estudiante de Ingeniería Mecánica

(ver Figura 1.5a), le permitiría visualizar una pieza de un motor pudiendo moverla, rotarla para verla desde un ángulo diferente o incluso aumentar su tamaño tal como si tuviese un motor real frente a sus ojos. En el campo de la medicina por ejemplo un alumno que estuviera estudiando alguna parte del cuerpo como el corazón (ver Figura 1.5b), podría ver en Realidad Aumentada un modelo 3D con un parecido muy realista a uno de verdad con las ventajas de poder apreciar todas sus partes. Además, para no depender de una base de datos de objetos 3D definida por un número fijo de objetos 3D, con la página web colaborativa se ofrece la posibilidad de compartir un número ilimitado de objetos 3D, por lo que si un cierto número de usuarios colaboran compartiendo, existiría una base de datos amplia y variada.



(a) Parte de un motor. (b) Corazón

Figura 1.5: Aplicaciones de la Realidad Aumentada

En la actualidad no existe ningún sistema que aporte este conjunto de funcionalidades complementadas, por lo que esta aportación es algo novedoso.

1.3 Estructura del documento

La documentación de este trabajo de fin de grado está compuesta por 6 capítulos. En los capítulos incluidos en este documento se intenta orientar al lector, de manera que comprenda las fases y puntos importantes por los que ha pasado el proyecto hasta llegar a cumplir los objetivos finales.

Capítulo 1: Introducción

Expone al lector una visión global y contextualizada del TFG.

Capítulo 2: Objetivos

Define la finalidad y se justifica en detalle los puntos principales y subprincipales del documento, incluyendo, además, objetivos docentes.

Capítulo 3: Antecedentes o Estado del Arte

Explica, de manera organizada, los aspectos básicos de las tecnologías y herramientas que se han analizado para el desarrollo de todas las partes del proyecto.

Capítulo 4: Método de trabajo

En este capítulo se explica la arquitectura de los sistemas creados, así como los algoritmos diseñados, y el método de trabajo seguido.

Capítulo 5: Resultados

Presenta y describe las iteraciones y los resultados conseguidos en cada fase del proyecto.

Capítulo 6: Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones que se han alcanzado, así como las posibles mejoras y futuras aplicaciones.

Bibliografía:

Muestra el conjunto de referencias a artículos, libros y páginas web utilizados como material bibliográfico durante el desarrollo del proyecto.

Capítulo 2

Objetivos

En este capítulo se pretende concretar y exponer los objetivos a alcanzar. Para ello, se presenta el objetivo principal, que está enfocado en explicar, de manera sencilla, la finalidad global del trabajo; los objetivos parciales, que se muestran de forma estructurada hasta llegar a un despliegue completo; y los docentes, en los que se va a exponer los objetivos académicos.

2.1 Objetivo general

El objetivo principal de este trabajo es enriquecer la información textual impresa en papel con información virtual integrada, de tal forma que se pueda animar y haya libertad de movimiento para examinar cualquier parte tal como si existiera un objeto real.

2.2 Objetivos parciales

En esta sección se analizan los objetivos parciales que permiten la realización del objetivo principal expuesto.

2.2.1 Aplicación Móvil

Consiste en el desarrollo de un sistema para dispositivos móviles que permita enriquecer visualmente la información a la que el usuario tiene acceso sobre papel. Los subobjetivos son:

- Realización de toma de fotos mediante la cámara integrada del dispositivo móvil, con opción a seleccionar, mediante la pantalla táctil, la parte donde se encuentra el texto.
- Conversión del texto fotografiado en texto digital, mediante técnicas de reconocimiento óptico de caracteres.
- Contextualizar el texto obtenido mediante el análisis basado en técnicas de procesamiento del lenguaje natural, con el fin de obtener un listado de palabras clave y su importancia en el texto.
- Realizar una petición a un servicio del servidor y procesar la respuesta.
- Representar el objeto 3D obtenido como respuesta, permitiendo al usuario seleccionar la superficie donde quiere representar este objeto 3D, rotar el objeto y desplazarlo sobre la superficie.

2.2.2 Aplicación Web

Desarrollo de un sitio web que permite la compartición de contenidos 3D, con el objetivo de tener un repositorio de objetos 3D amplio. Los subobjetivos son:

- Permitir al usuario tener su propio perfil.
- Sistema de subida de objetos, donde el usuario defina las características, ofrezca datos descriptivos y suba archivos y fotos.
- Sistema de votación y *likes*, donde el usuario pueda dar una puntuación y pueda guardar en su lista de favoritos los objetos que desee. Además, con esta información junto al número de descargas servirá para determinar la relevancia de cada objeto, la cual es necesaria para el algoritmo de selección de objetos 3D del servidor, que es desarrollado con detalle en los próximos capítulos.

2.2.3 Servidor Gestor de Objetos 3D

Realización de un gestor, que se encarga de las solicitudes recibidas por clientes de la aplicación móvil. Los objetivos específicos son:

- Encontrar los objetos candidatos conforme a las palabras clave recibidas.
- Hacer una selección del objeto 3D más adecuado. Basándose en una fórmula matemática que toma valores del sistema de puntuación y de *likes*.
- Envíar respuesta al cliente.

2.3 Objetivos docentes

Además de los objetivos técnicos mencionados, hay una serie de objetivos más centrados en el aprendizaje consecuente de la realización de este proyecto. Los objetivos docentes a destacar son los siguientes:

1. Aplicar técnicas de gestión de proyectos.
2. Ser capaz de crear una aplicación Web utilizando un patrón Modelo-Vista-Controlador, con el uso del *framework* JSF.
3. Dominar el Sistema Gestor de Base de Datos escogido, tanto para la realización de gestión de datos como para el funcionamiento correcto del *pool* de conexiones. También llevará a cabo unas medidas de seguridad que permiten garantizar la protección de datos.
4. Aplicar y comprender el funcionamiento de la librería de Google para Android encargada del reconocimiento de caracteres.
5. Aplicar y comprender el funcionamiento de la librería de Google para Android encargada del procesamiento del lenguaje natural para la extracción de palabras clave.
6. Aprender como funciona y ser capaz de realizar una comunicación asíncrona, entre una aplicación móvil y el servidor gestor de objetos 3D, mediante un servicio REST. A su vez, entender la manera en que esta arquitectura trabaja con el protocolo HTTP, así como el servidor realiza el envío de contenido basado en archivos, no sólo texto plano.
7. Aprender a renderizar objetos 3D en Realidad Aumentada para Android con la librería ARCore, de manera asíncrona y en diferentes formatos. Comprendiendo como funciona el sistema

de iluminación (tipos de iluminación, posición, intensidad, etc.), la generación de sombras dinámica, tamaño del objeto, rotación y posición con respecto al eje universal y utilización de animaciones en objetos 3D que lo permiten.

8. Realización de esta documentación en Latex.

Capítulo 3

Antecedentes o Estado del Arte

En este capítulo se va a analizar el estado del arte de todos los componentes de este proyecto.

3.1 Desarrollo Web

La historia del desarrollo web es considerada en la actualidad como un fenómeno que ha evolucionado a pasos agigantados, a pesar de no tener mucha antigüedad, desde la aparición de Internet y de diversas tecnologías y protocolos que se han ido creando debido a las nuevas necesidades y demandas del sector.

Los inicios de Internet se remontan a la década de 1960, aunque es durante la década de 1990 cuando aparece el interés masivo y su correspondiente demanda debido a la necesidad de comunicarse por gran parte de la sociedad. Es por tanto cuando realmente podemos hablar del actual significado de web, tal y como la conocemos ahora, así como de las tecnologías que han promovido el desarrollo web actual.

Si queremos explicar a un nivel global como ha sido el impacto en la web durante todos estos años, refiriéndonos a las nuevas tecnologías de desarrollo y las actualizaciones de las tecnologías ya existentes, entonces, hablamos de *tipos* de web [Ban14].

La **web 1.0** hace referencia al primer estado de la *World Wide Web* (WWW), lo que entendemos como el periodo comprendido desde sus inicios hasta que aproximadamente apareció el fenómeno *fiebre punto com* ocurrido en el año 2001. Durante este primer estado de la web, los diseños típicos de las páginas web eran páginas estáticas, debido a que la demanda de los usuarios de aquel entonces no requería una alta interacción con la web. La mayoría de los casos que podías encontrarte eran sitios web que su única función era mostrar información no variable separada en páginas web a las que se redirigía con una navegación tradicional. En cuanto al contenido multimedia más avanzado era la reproducción de imágenes en movimiento (GIF), en muchos casos la resolución no era variable, debido a que venia definida por el propio navegador web con el que se accedía. Las páginas eran creadas de forma fija y casi nunca se actualizaban, y la forma más usada para enviar información en formularios era mediante el correo electrónico, a pesar de todos los problemas de seguridad que esto podía provocar.

La **web 2.0** la entendemos como el estado que continúa a la web tradicional ó 1.0, cuya principal característica es que la participación de los usuarios deja de ser pasiva para convertirse en una participación activa, permitiendo contribuir en la red, creando y dando soporte a una sociedad virtualizada que facilita la comunicación y genera conocimiento y contenido. Esta segunda etapa, se caracteriza

por la aparición masiva de *websites* con páginas web dinámicas, es decir, que su contenido puede cambiar de la manera frecuente. Por ejemplo, entre los casos más conocidos de esta etapa estan los blogs, las redes sociales, foros, etc.. Aunque por otro lado aparecieron las aplicaciones web, *cloud services*, etc. Debido a la aparición de estas soluciones en la web, que su principal fin consiste en cumplir las demandas de funcionalidades cada vez más complejas y extensas exigidas por los usuarios, han surgido nuevas tecnologías y arquitecturas punteras que facilitan al desarrollador su trabajo, por ello, la aparición de esta etapa puede designarse como sinónimo de la aparición de herramientas y tecnologías que actualmente utilizamos, estas son el apoyo principal para la implementación de la aplicación web que se tratará en este proyecto [PII15].

3.1.1 Análisis de los lenguajes de programación

A pesar de la gran diferencia que puede haber en comparación de complejidad entre una solución web y otras, generalmente se puede realizar un esquema de alto nivel para explicar la interacción entre diferentes tecnologías en la creación de aplicaciones web enfocadas a todos los tipos de usuarios. El desarrollo web de la actualidad tiene una clara división en dos capas o sectores con importantes diferencias, por una parte, podemos definir *front-end* o la capa de presentación para el usuario, y por otra parte, *back-end* o capa de modelado, acceso y procesado de datos. En la actualidad, para ambos sectores existen numerosas tecnologías pero no todas tienen compatibilidad con otras a pesar de que si que existe una cierta flexibilidad. En la mayoría de casos, una determinada tecnología de *front-end* esta desarrollada para que funcione con un determinado conjunto de tecnologías *back-end* [Kal13].

Los lenguajes *front-end* más conocidos actualmente son:

- JavaScript
- HTML5
- CSS

La referencia en lenguajes *back-end* es la siguiente:

- JavaScript
- Java Enterprise
- PHP
- C#
- Python

Sin embargo, para dar un enfoque representativo de las tendencias en tecnologías web, no es suficiente con hablar de los lenguajes de desarrollo utilizados para la web ya que existen diversas tecnologías que comparten un mismo lenguaje, además de marcos de referencia, lo que hace que usen distintos lenguajes y tecnologías. Por esto, en algunas circunstancias, cuando nos referimos al uso que esta teniendo un determinado lenguaje más que otro, sería más correcto referirnos a qué tecnología hace uso de ese lenguaje con mayor repercusión. El mejor ejemplo actual para entenderlo podría ser JavaScript, que es un lenguaje que siempre se ha estado utilizando para la parte cliente del desarro-

llo, mientras que actualmente esta teniendo mucho uso como lenguaje de *back-end*, esto es por la popularidad de las tecnologías como Node.JS [Nod18].

Debido a que no hay un análisis puro de investigación, las labores de obtención de estadísticas fiables sobre uso, popularidad de lenguajes y tecnologías aplicados al desarrollo web son tareas complicadas y con incertidumbre.

A continuación, se va analizar la popularidad de los lenguajes de programación. Para ello, se van a observar los datos de TIOBE, que es una de las páginas más fiables para rankings de este sector. El índice publicado se actualiza una vez al mes. Las puntuaciones están basadas en el número de ingenieros expertos en todo el mundo, cursos y proveedores externos. Es importante aclarar, que el índice que proporciona TIOBE no trata sobre «los mejores lenguajes de programación» o «el lenguaje en el que se han escrito más lineas de código» [TIO18].

Los motores de búsqueda como Google, Bing, Yahoo, Wikipedia, Amazon, Youtube y Baidu también influyen a la hora de calculcar las puntuaciones. El seguimiento de los lenguajes de programación más populares que realiza esta web es público, y lo hace basándose en los primeros 25 motores de búsqueda que aparecen en Alexa[1] bajo las siguientes condiciones:

- La página de entrada del sitio contiene una función de búsqueda.
- El sitio contiene un contador de la cantidad de visitas a la página.
- Los resultados deberían estar disponibles en HTML con etiquetas claras.
- Los motores de búsqueda en idiomas con caracteres especiales deben codificarse correctamente.
- El motor de búsqueda debería devolver al menos 1 hit por 1 consulta.
- Están excluidos sitios para adultos.

Por otra parte, los lenguajes de programación también deben cumplir el requisito de tener una entrada en Wikipedia, donde se defina claramente su estado como un lenguaje de programación. Esta es la razón por la que (Ruby on) Rails, Excel, Android, Boost, Cocoa, ASP y AJAX no estan considerados lenguajes de programación en este índice.

El lenguaje de programación debe ser Turing completo[2]. Como consecuencia de ello, HTML y XML no están considerados lenguajes de programación para TIOBE. Lo mismo ocurre para SQL que, por ejemplo, es imposible escribir un bucle infinito. Aunque a las extensiones PL/SQL y Transact-SQL si se les considera lenguajes de programación.

También es imprescindible que el lenguaje de programación obtenga al menos 5000 aciertos en su búsqueda en Google con los parámetros: +«*<language> programming*».

Este índice (ver Figura 3.1) se puede utilizar para comprobar si tus habilidades programando están al día o para realizar una decisión estratégica sobre que lenguaje de programación se debería utilizar para empezar con la creación de un nuevo sistema *software.*

[1] `https://www.alexa.com/topsites`

[2] `https://es.wikipedia.org/wiki/Turing_completo`

Aunque los *rankings* de popularidad compartidos por TIOBE no sean especificamente para lenguajes de programación enfocados al desarrollo web, si son de utilidad para este proyecto, al ser un estudio actual, debido a que gran parte de los lenguajes más populares también son los más utilizados en desarrollo web. Por otra parte, es un estudio suficientemente fiable para su análisis en este proyecto debido a que tiene una base sólida que lo respalda.

May 2018	May 2017	Change	Programming Language	Ratings	Change
1	1		Java	16.380%	+1.74%
2	2		C	14.000%	+7.00%
3	3		C++	7.668%	+2.92%
4	4		Python	5.192%	+1.64%
5	5		C#	4.402%	+0.95%
6	6		Visual Basic .NET	4.124%	+0.73%
7	9	^	PHP	3.321%	+0.63%
8	7	v	JavaScript	2.923%	-0.15%
9	-	^^	SQL	1.987%	+1.99%
10	11	^	Ruby	1.182%	-1.25%

Figura 3.1: Top 10 Lenguajes de programación del índice de TIOBE para mayo de 2018.

En la figura anterior se muestra un listado con los diez primeros lenguajes de programación más populares según el índice de TIOBE [TIO18].

De este listado, podemos deducir que JavaScript y PHP tienen una alta popularidad, además de que son directamente relacionados casi por completo con el mundo del desarrollo web.

Como dato a recalcar, los primeros puestos los ocupan lenguajes aplicados a propósitos generales, entre los que destacan Java y C#, que aparte de ser utilizados en innumerables sectores, tienen un papel importante en el desarrollo web actualmente. Por otro lado, lenguajes como C, C++ y Python se suelen utilizar menos en el sector del desarrollo web, ya que son de bajo nivel. Esto es debido a que, en el desarrollo web, a las páginas se les requiere el aprovisionamiento de altas prestaciones. Podemos ver el ejemplo de páginas web muy populares, que se les exige que soporten una gran cantidad de conexiones simultáneas y a la vez ser lo más eficiente posibles en el uso de recursos.

Websites	Popularity (unique visitors per month)[1]	Front-end (Client-side)	Back-end (Server-side)
Google.com[2]	1,600,000,000	JavaScript	C, C++, Go,[3] Java, Python, PHP (HHVM)
Facebook.com	1,100,000,000	JavaScript	Hack, PHP (HHVM), Python, C++, Java, Erlang, D,[6] XHP,[7] Haskell[8]
YouTube.com	1,100,000,000	JavaScript	C, C++, Python, Java,[11] Go[12]
Yahoo	750,000,000	JavaScript	PHP
Amazon.com	500,000,000	JavaScript	Java, C++, Perl[16]
Wikipedia.org	475,000,000	JavaScript	PHP, Hack
Twitter.com	290,000,000	JavaScript	C++, Java, Scala, Ruby[19]
Bing	285,000,000	JavaScript	C#
eBay.com	285,000,000	JavaScript	Java,[21] JavaScript,[22] Scala[23]
MSN.com	280,000,000	JavaScript	C#
Microsoft	270,000,000	JavaScript	C#
Linkedin.com	260,000,000	JavaScript	Java, JavaScript,[24] Scala
Pinterest	250,000,000	JavaScript	Django,[26] Erlang
WordPress.com	240,000,000	JavaScript	PHP, JavaScript [28] (Node.js)

Figura 3.2: Lenguajes de programación que utilizan los sitios web más populares.[3]

En la Figura 3.2 podemos observar que los lenguajes de programación C, C++ y Python son los que más destacan en el *back-end* de los sitios web con un gran número de visitas [Wik18].

En cuanto al *front-end*, vemos que JavaScript es el único lenguaje de programación utilizado, esto es porque tecnologías como HTML y CSS son consideradas como lenguajes de marcado en lugar de lenguajes de programación. Es cierto que todas esas páginas utilizan HTML, CSS y JavaScript en su *front-end* y sus principales diferencias están en la parte del *back-end*.

A continuación, se muestra una tabla con los *frameworks* más conocidos para algunos de los lenguajes más populares de programación *back-end*:

Lenguajes	Frameworks
Java	Java Server Faces y Spring MVC
C#	ASP.NET MVC
Python	Django y Flask
PHP	Laravel y Symfony
JavaScript	Express y Koa
Ruby	Ruby on Rails

Cuadro 3.1: Listado de Frameworks para *back-end.*

Los *framework* de JavaScript mencionados en la Tabla 3.1 están asociados con Node.js.

Existen también algunos *frameworks* para el *front-end*, aunque hay menor variedad. Actualmente, lo más utilizados son AngularJS y Bootstrap.

[3] `https://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites`

3.1.2 Análisis de los *frameworks*

Para realizar el análisis vamos a conocer sus características y en base a los siguientes criterios se van a evaluar: grado de madurez, tamaño y grado de actividad de la comunidad, disponibilidad de librerías y aplicaciones de terceros, dificultad de la curva de aprendizaje, compatibilidad, rendimiento y escalabilidad.

JavaServer Faces (JSF)

JSF es una tecnología y *framework* de la especificación de Java para el desarrollo de interfaces de usuario basado en componentes para aplicaciones Java EE. JSF establece un estándar para construir interfaces de usuario en el servidor. JSF usa, como tecnología para realizar el despliegue de las páginas, JavaServer Pages (JSP) aunque puede también adaptarse a tecnologías como XUL (*XML-based User-interface Language*, que es un lenguaje enfocado a la interfaz de usuario y basado en XML) [Jun14].

Este *framework* trae un conjunto de APIs cuya función es crear componente en una interfaz de usuario y gestionar su estado, el manejo de eventos, la validación de entradas, tener un esquema de navegación de las páginas definido y dar soporte para que sea accesible e internacionalizado.

JSF permite tener un modelo de eventos *server-side*, administración de estados y *Beans* administrados. Además, tiene dos librerías con *tags* personalizados para JSP con lo que se puede interpretar una interfaz JSF dentro de una página JSP [Ora18a].

Actualmente, la última versión estable de JSF es la 2.3.0, pero a partir de la versión 2.0 fue cuando hubo grandes cambios. Una de las principales diferencias con las versiones anteriores es que Facelets es el lenguaje definido para la vista predeterminada en JSF 2.0 en lugar de JSP [Voh14]. Algunas de las nuevas características son:

- Nuevos *scopes* (definen la vida de un *bean*).
- Simplificación y configuración basada en anotaciones.
- Permite guardar los estados.
- Mejora del manejo de excepciones y de recursos.
- Nuevo sistema de eventos.
- Proporciona soporte integrado para Ajax al estandarizar el ciclo de vida de peticiones Ajax y proporcionar interfaces de desarrollo simples para eventos Ajax, permitiendo que cualquier evento desencadenado por el cliente pase por validación, conversión y finalmente invocación de método, antes de devolver el resultado al navegador vía una actualización XML DOM.
- Soporte para HTML5, vistas sin estado y *Resource library contracts*.

Los componentes o extensiones más conocidos y que permiten Ajax son: BootsFaces (basado en Bootstrap), Apache MyFaces, PrimeFaces, ICEfaces (Ajax sin JavaScript y con componente Rich) y OmniFaces. Aunque todos son muy parecidos y comparten muchos componentes, dependiendo de la solución que queramos desarrollar podemos encontrar más utilidad en alguno.

Spring MVC

Spring MVC es un *framework* MVC que emplea un *dispatcher servlet* que invoca métodos en controladores y maneja el avance a una vista. El primer beneficio de usar Spring MVC es: No necesitas escribir tu propio *dispatcher servlet* [Dec16].

Las características con las que viene equipado Spring MVC para hacer el desarrollo más rápido son [Spr18]:

- Provee un *dispatcher servlet*, ahorrándote tener que escribir uno.
- Tiene un archivo de configuración basado en XML que puedes editar sin necesidad de volver a compilar la aplicación.
- Instancia clases de controlador y genera *Beans* con entradas de usuarios.
- Automáticamente identifica la entrada del usuario con el tipo correcto. Por ejemplo, Spring MVC puede, directamente, parsear una cadena y establecer una propiedad del tipo a float o decimal.
- Valida la entrada del usuario y redirige al usuario de vuelta a esa entrada del formulario si la validación falla.
- Es parte del *framework* de Spring, por lo tanto trae todo lo que Spring ofrece.
- Soporta internacionalización y localización. Esto significa, que puedes ver mensajes en diferentes idiomas dependiendo de donde estés.
- Soporta múltiples tecnologías para las vistas.

El acrónimo de MVC viene de *Model View Controller*, y es considerado muy popular a la hora de construir interfaces de usuario desacoplando las capas de datos y de presentación. El patrón MVC, como podemos ver en la Figura 3.3, está compuesto de la capa del **Modelo**, que consiste en representaciones varias de los datos que la aplicación conoce, la **Vista**, esta hecha de algunas representaciones de datos que serán mostrados a los usuarios, y el **Controlador**, que es la parte de la aplicación que manejará las interacciones de los usuarios (como un puente entre el modelo y la vista) [War15].

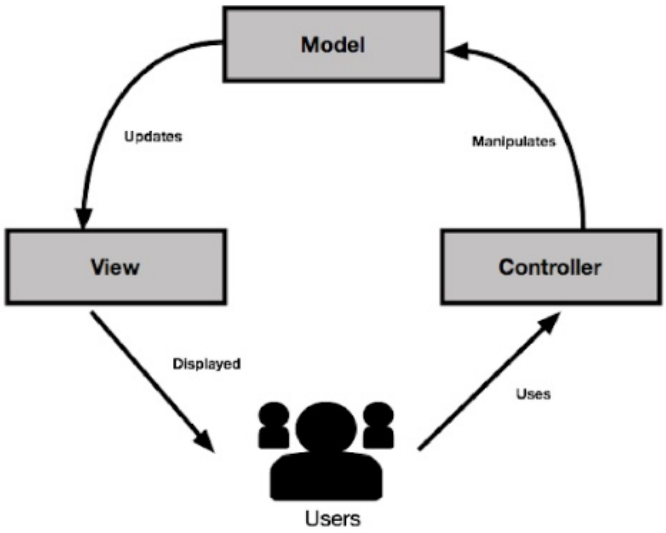


Figura 3.3: Arquitectura MVC.

ASP.NET MVC

Es un *framework* de desarrollo Web de Microsoft que combina eficacia y claridad en la arquitectura *model-view-controller* (MVC), con las ideas más actualizadas y técnicas de desarrollo ágil, y las mejores partes de la plataforma ASP.NET. Es una completa alternativa al tradicional ASP.NET *Web Forms*, presenta todo tipo de ventajas incluso para los proyectos de desarrollo Web más triviales. ASP.NET hizo un gran cambio desde que llegó en 2002 (ver Figura 3.4) hasta que la primera versión de ASP.NET MVC fue publicada por Microsoft en 2009 [Mic18a].

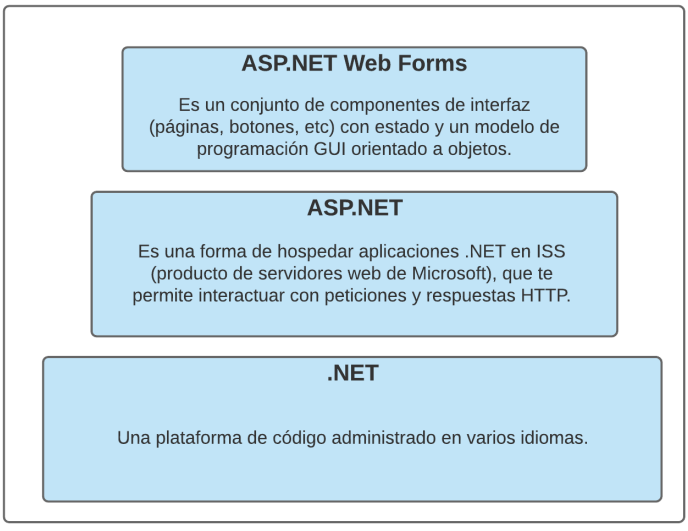


Figura 3.4: Evolución tecnológica de ASP.NET Web Forms.

El enrutamiento (*routing*) es uno de los pilares básicos de ASP.NET MVC. Este concepto permi-

te que las aplicaciones puedan aceptar *requests* a URL que no están definidas como ficheros físicos en el servidor. Por ejemplo, en ASP.NET Web Forms la dirección URL tendría el siguiente formato «*http://sitioweb/productos.aspx?categoria=video*», por lo que existiría un fichero físico llamado *productos.aspx* en el directorio del sitio web. En la versión ASP.NET MVC este mismo ejemplo tendría el siguiente formato: «*http://sitioweb/productos/video*» y en el servidor no existiría ninguna carpeta llamada «productos» ni «video». De manera predefinida, ASP.NET MVC genera las peticiones al controlador y vista correspondiente dependiendo de la URL [Fre15].

Django

Es un *framework* gratuito y de código abierto, escrito en Python, que sigue el patrón arquitectural *model-view-template* (MVT). Lo mantiene Django Software Foundation (DSF), una organización independiente sin ánimo de lucro.

El objetivo principal de Django es facilitar la creación de sitios web complejos basados en bases de datos. Django enfatiza la reutilización y la «capacidad de conexión» de los componentes, menos código, bajo acoplamiento, desarrollo rápido y el principio de No te repitas (DRY, del inglés *don't repeat yourself*). Utiliza Python en todas partes, incluso en archivos de configuración y modelos de datos. Django también proporciona una interfaz administrativa opcional parar crear, leer, actualizar y eliminar que se genera dinámicamente a través de la introspección y se configura a través de modelos de administración [Hol10].

Algunos sitios muy famosos que utilizan Django son: Public Broadcasting Service, Instagram, Mozilla, The Washington Times, Disqus, Bitbucket y Nextdoor. También se usaba en Pinterest, pero decidieron moverse a otro *framework* llamado Flask.

Django fue creado a finales de 2003, cuando los programadores web del periódico Lawrence Journal-World, Adrian Holovaty y Simon Willison, comenzaron a utilizar Python para crear aplicaciones.

A pesar de tener su propia nomenclatura, como nombrar los objetos invocables que generan las «vistas» de respuestas HTTP, el núcleo del *framework* Django puede verse como una arquitectura MVC. Consiste en un *object-relational mapper* (ORM) que media entre los modelos de datos (definidos como clases de Python) y una base de datos relacional (*Model*), un sistema para procesar peticiones HTTP con un sistema de plantillas web (*View*), y un *dispatcher* de URL basado en expresiones regulares (*Controller*) [Fou18a].

Flask

Flask es un framework minimalista que esta desarrollado en Python, lo que facilita la creación de aplicaciones web y a su vez se necesitan muy pocas lineas de código.

Es un *framework* relativamente joven, apareció en 2010. La razón por la que Flask es considerado más Python que Django es simple, porque el código de Flask para aplicaciones web es, en la mayoría de casos, más explicito. Flask es la opción para la mayoría de los principiantes debido a la falta de obstáculos para poner en marcha una aplicación simple [Ron18].

Las ventajas de Flask son:

- Es extremadamente flexible.
- Es minimalista sin sacrificar potencial.
- Es simple para aprender y usar.
- El enrutamiento URL es fácil.
- Tiene un núcleo pequeño y se pueden añadir extensiones fácilmente.

La parte negativa de este *framework* es: no es amigable con tareas asíncronas, el soporte y la documentación es muy limitado, carece de base de datos/ORM/formularios y tiene características muy limitadas [Mai15].

Laravel

Laravel es un *framework open-source* y gratuito creado por Taylor Otwell. Permite desarrollar aplicaciones web y servicios web con PHP 5 y PHP 7 [Otw18].

La primera versión beta de Laravel 1 fue publicada en junio de 2011, y fue escrita completa de la nada. Presentaba un ORM personalizado; enrutamiento basado en el cierre (inspirado por Ruby Sinatra); un sistema de módulos para extensiones; y asistentes para formularios, validaciones, autentificación, etc. Poco después, en noviembre de 2011 y febrero de 2012 se publicaron Laravel 2 y 3, respectivamente. Se introdujeron controladores, unidades de *testing*, una herramienta de linea de comandos, un contenedor de control de inversiones (IoC) y migraciones.

Con Laravel 4, Taylor reescribió el *framework* entero. Desarrolló un conjunto de componentes bajo el código llamado *Illuminate* y, durante mayo de 2013, publicó Laravel 4 con una estructura completamente nueva. En la versión 4, se introdujeron colas, un componente para correos, fachadas y *seeding* de bases de datos.

En febrero de 2015 se publicó Laravel 5, se caracterizaba por mejorar la estructura de directorios, la eliminación de los asistentes de formularios y HTML, la introducción de un contrato de interfaces, una serie de nuevas vistas, Socialite para realizar la autentificación de medios de comunicación, Elixir para la compilación de activos, Scheduler para simplificar cron, dotenv para la gestión de entornos simplificados, solicitudes de formularios y nuevos REPL (bucles *read-evaluate-print*) [Sta17].

Las principales características de Laravel son [Bea15]:

- **Modularidad:** Está construido sobre más de 20 librerías diferentes y en si mismo también está dividido en módulos individuales.
- **Capacidad para pruebas:** Construido desde cero para facilitar las pruebas (*testing*).
- **Enrutamiento:** Da gran flexibilidad cuando defines las rutas de la aplicación.
- **Gestión de configuración:** Diferentes configuraciones pueden ser aplicadas en diferentes entornos.
- **Creador de esquemas, migraciones y *seeding*:** Inspirado en Rails, estas características permiten definir el esquema de la base de datos en código PHP y localizar cualquier cambio con

ayuda de las migraciones de bases de datos.

- **Motor de plantillas:** Inspirado parcialmente por el lenguaje de plantilla Razor en ASP.
- **E-mailing:** Con la clase *Mail*, la cual se basa en una librería muy popular llamada *SwiftMailer*, Laravel hace que sea fácil enviar un e-mail.
- **Autentificación:** Es una característica común en las aplicaciones web, por eso Laravel trae implementaciones como registrarse, autentificarse e incluso la opción de recordar la contraseña.
- **Redis:** Es un almacén de valores-clave en memoria que tiene la reputación de ser extremadamente rápido.

Ruby on Rails

Es también conocido como RoR o Rails, es un *framework open-source* que esta escrito en Ruby, aunque el lenguaje de programación Ruby no está diseñado para usarse en el desarrollo Web, por eso existe Rails. RoR sigue el patrón arquitectural *Model-View-Controller* (MVC), provee estructuras predefinidas para la base de datos, un servicio web y páginas web. Facilita el uso de estándares web para la transferencia de datos como JSON o XML, para la visualización como HTML, CSS y JavaScript, y para la interfaz de usuario. Rails, aparte de MVC, también utiliza otros patrones y paradigmas de la ingeniería del *software*, como CoC y DRY [Han18].

Una aplicación de Ruby on Rails tiene varios componentes, como podemos ver en la Figura 3.5.

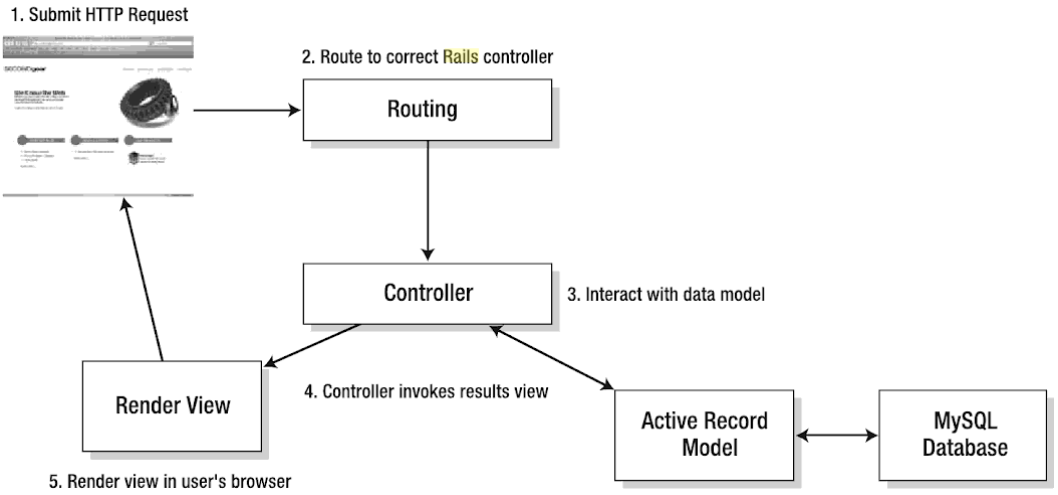


Figura 3.5: Ejemplo de aplicación genérica de Ruby on Rails.

Una simple solicitud web puede tener un camino largo en Ruby on Rails. Cuando el usuario realiza la primera petición de una página via navegador, el controlador de Rails (*Action Controller*) recibe la petición y la redirecciona al método correcto del enrutado URL. Una vez que el método correcto esta invocado, es ejecutado y toma la información que necesita de la base de datos SQL usando el modelo *Active Record*. Cuando tiene toda la información que necesita, renderiza la vista final (HTML, CSS e imágenes) en el navegador del usuario [Wil07].

3.1.3 Gestor de Base de Datos

Como ya se ha comentado en los apartados anteriores, las capas clave en el desarrollo web son *back-end* y *front-end*, pero por si solas no completan una solución web funcional. Para completar este esquema global de desarrollo web necesitamos tecnologías que sirvan para el intercambio de información entre el cliente y el servidor, un sistema gestor de la base de datos, servidores capaces de interpretar la información recibida y la lógica correspondiente a estos elementos.

La base de datos es una colección de datos. Podría, por ejemplo, contener información sobre una empresa particular. Un Sistema Gestor de Base de Datos (SGBD) consiste en una colección de datos interrelacionados y un conjuntos de programas para acceder a estos datos. El objetivo principal de un SGBD es proveer un entorno que sea conveniente y eficiente para usar recibiendo y almacenando información en una base de datos [Ked09].

Los sistemas de base de datos están diseñados para manipular grandes cantidades de información. El sistema de base de datos debe asegurar la seguridad de la información almacenada, a pesar de que el sistema pueda tener alguna problema o un acceso no autorizado. Si los datos están compartidos entre algunos usuarios, el sistema debe mantener la consistencia [Ked09]. Las operaciones que se ejecutan sobre la base de datos para así poder proporcionar al usuario un resultado en función de sus peticiones se realiza de un modo eficiente y seguro. Estas características permiten el cumplimiento de una serie de funciones, que se pueden agrupar de la siguiente manera [Pow15]:

1. **Definición de los datos:** El SGBD ha de poder definir objetos de la base de datos a partir de definiciones en código fuente para transformarlas en una versión objeto.
2. **Manipulación de los datos:** Debe dar respuesta a las peticiones de los usuarios, como eliminar, actualizar, solicitar datos, etc. El manejo de estos datos debe ser lo más eficiente posible y poder modificar la estructura de la base de datos sin afectar externamente.
3. **Seguridad e integridad de los datos:** Un SGBD debe registrar todo tipo de uso de la base datos, con cualquier tipo de petición debe aplicar la normativa establecida para cumplir con los requisitos de seguridad e integridad previamente definidos. A su vez, ha de garantizar su seguridad ante cualquier posible ataque y no permitir el acceso a usuarios no deseados.
4. **Recuperación y restauración de los datos:** Ante un posible fallo el sistema debe garantizar que sea posible recuperar y restaurar los datos.

Los criterios para elegir el SGBD adecuado deben tener en cuenta que el sistema este o pueda estar integrado con otro *software*, que sea escalable y adaptarse a un posible crecimiento del sistema, que sea sostenible y rentable. Por otra parte, tiene gran importancia la seguridad de los datos almacenados, la funcionalidad que nos permite cumplir con los requisitos y necesidades y la facilidad de uso para el usuario.

Actualmente existen multitud de SGBD, la mayoría son relacionales como podemos apreciar en la Figura 3.6 donde se muestran los diez SGBD más populares:

342 systems in ranking, May 2018

Rank May 2018	Rank Apr 2018	Rank May 2017	DBMS	Database Model	Score May 2018	Score Apr 2018	Score May 2017
1.	1.	1.	Oracle	Relational DBMS	1290.42	+0.63	-63.90
2.	2.	2.	MySQL	Relational DBMS	1223.34	-3.06	-116.69
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1085.84	-9.67	-127.96
4.	4.	4.	PostgreSQL	Relational DBMS	400.90	+5.43	+34.99
5.	5.	5.	MongoDB	Document store	342.11	+0.70	+10.53
6.	6.	6.	DB2	Relational DBMS	185.61	-3.34	-3.23
7.	↑ 9.	↑ 9.	Redis	Key-value store	135.35	+5.24	+17.90
8.	↓ 7.	↓ 7.	Microsoft Access	Relational DBMS	133.11	+0.89	+3.24
9.	↓ 8.	↑ 11.	Elasticsearch	Search engine	130.44	-0.92	+21.62
10.	10.	↓ 8.	Cassandra	Wide column store	117.83	-1.26	-5.28

Figura 3.6: Ranking de SGBD acorde a su popularidad [DE18].

Las bases de datos se pueden clasificar, además de por la función que desempeñan, por su modelo de la administración de datos. Un modelo de datos se puede definir como una descripción de información que sabemos como *contenedor de datos*, además de las funciones para guardar y devolver los datos de estos contenedores. Los modelos de datos son abstracciones que otorgan eficiencia en la base de datos de un sistema, normalmente se les conoce como algoritmos y conceptos matemáticos. El modelo de datos relacional es el más usado para soluciones reales y administración de datos dinámica.

El concepto de base de datos relacional fue definido por primera vez por el Doctor Edgar F. Codd en un artículo de investigación de IBM llamado «*System R4 Relational*» que fue publicado en 1970. Al principio, no estaba claro si algún sistema basado en este concepto podría lograr el éxito comercial. Sin embargo, una compañía llamada *Software Development Laboratories Relational Software* entró en escena en 1977 y publicó un producto, llamado Oracle V.2, como la primera base de datos relacional del mercado a los dos años. En 1985, Oracle ya tenía más de 1000 cliente con sitios que utilizaban base de datos relacionales. Curiosamente, IBM no acogió la tecnología relacional en productos comerciales hasta 1983.

Otros modelos de datos también muy conocidos son los jerárquicos y los de red (ver Figura 3.7) que además coincidieron en su momento de popularidad. El modelo jerárquico organiza los datos en una estructura de árbol. Hay una jerarquía de segmentos de datos primarios y secundarios. Esta estructura implica que un registro puede tener información repetitiva, generalmente en los segmentos de datos secundarios. Con el modelo de red, algunos datos se modelaron de forma más natural con más de un padre por cada hijo. Por lo que, el modelo de red permitió el modelado de relaciones de datos de muchos a muchos. Por último, pero menos populares, tenemos los modelos de datos transaccionales, multidimensionales, orientados a objetos, documentales y deductivos [RGS13].

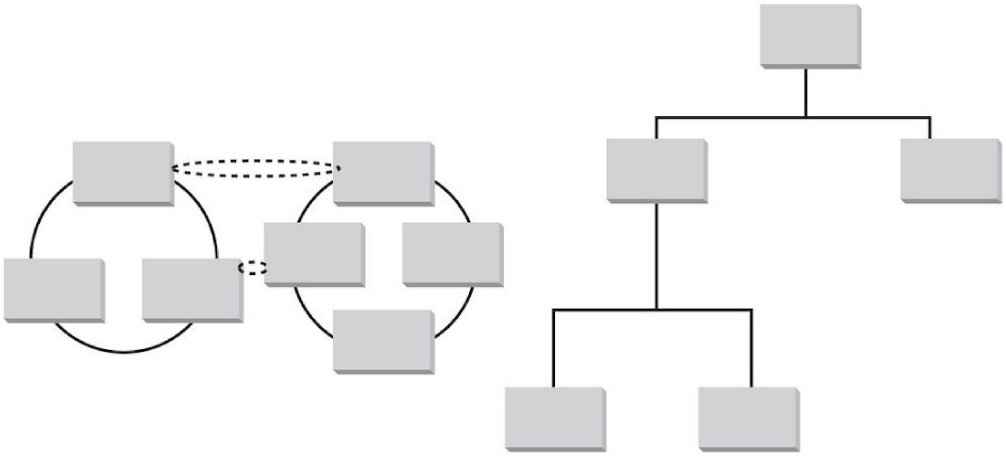


Figura 3.7: Modelo de red (izquierda) y modelo jerárquico (derecha).

A continuación se van a analizar los SGBD más conocidos y que mejor se adaptan a las necesidades de este proyecto.

MySQL

MySQL es un sistema encargado de la administración de bases de datos relacionales. Las bases de datos relacionales archivan los datos en diferentes tablas, en lugar de almacenar todos los datos en un único archivo. Permitiendo una mejora de la eficiente y flexibilidad. Entre las tablas se definen relaciones que permiten combinaciones de datos de diferentes tablas.

Está desarrollado por Oracle Corporation y se considera la base de datos de código abierto con mayor popularidad y la segunda más popular en general junto a Oracle y Microsoft SQL Server, en el sector del desarrollo web.

Como MySQL es un *software* de código abierto, significa que cualquier persona puede usarlo y modificarlo. El código fuente de MySQL lo puede descargar cualquier persona y usarlo de forma gratuita. El código fuente puede ser estudiado y modificado para que se ajuste a las necesidades del usuario [Ora18b].

A diferencia de proyectos como Apache, donde el software lo desarrolla una comunidad pública y el *copyright* del código le pertenece al autor individual, MySQL está bajo el patrocinio de una empresa privada, a la que le pertenece los derechos de autor de gran parte del código. Esto es lo que da lugar a un esquema de doble licenciamiento. El SGBD está disponible de dos formas, una versión llamada *Community* que es una Licencia pública y varias versiones *Enterprise*, que va enfocado a empresas que lo quieran incorporar en sus servicios privados.

Gran parte del desarrollo de MYSQL es en ANSI C y C++. MySQL es una herramienta utilizada por grandes y populares empresas en sus sitios web como Wikipedia, Google, Facebook, Twitter, Flickr y Youtube. Según cifras publicadas por el fabricante, en la actualidad hay más de seis millones de licencias de MySQL funcionando, esto superaría al resto de herramientas para base de datos [DuB13].

MySQL funciona sobre un gran número de plataformas, entre las más conocidas: GNU/Linux (dónde alcanza su máxima eficiencia), todas las versiones de Windows (incluido Windows Server), Mac OS X, etc.

Microsoft SQL Server

Es un sistema gestor de base de datos relacional desarrollado por Microsoft. Como servidor de base de datos, es un producto *software* cuya principal función es almacenar y recibir datos según las peticiones hechas por otras aplicaciones que se estén ejecutando en el mismo ordenador o en otro ordenador conectado a la red.

Microsoft ha sacado al mercado decenas de ediciones de Microsoft SQL Server, cada una con el objetivo de llegar a diferentes audiencias para cargas de trabajo que van desde pequeñas aplicaciones de una sola máquina hasta grandes aplicaciones orientadas a Internet con muchos usuarios simultáneos [Aru16].

Para el desarrollo se ha usado el lenguaje Transact-SQL (TSQL), que es una implementación del estándar ANSI del lenguaje SQL, que se utiliza para gestionar y retomar datos (DML), crear nuevas tablas y asignar relaciones entre ellas (DDL).

Las características de este SGBD son [Mic18b]:

- Permite las transacciones.
- Se pueden crear procedimientos almacenados.
- Tiene una interfaz gráfica para la administración, en la que se pueden utilizar comandos DDL y DML de manera gráfica.
- Da la posibilidad de trabajar en el modo *client-server*, almacenando la información y todos los datos en el servidor y el cliente tiene acceso a la información.
- Se puede administrar datos e información de otros servidores.
- A través de los ADP (*Access Data Project*) permite desarrollar proyectos haciendo uso de Microsoft SQL Server y Microsoft Access.

Oracle Database

Oracle Database es un sistema de gestión de base de datos relacional de tipo objeto-relacional (ORDBMS, es el acrónimo en inglés de *Object-Relational Data Base Management System*), este tipo es una extensión de la base de datos relacional tradicional el cual tiene las características de la programación orientada a objetos (POO) [Ora18c].

Se considera Oracle como uno de los sistemas de bases de datos más completos. Su dominio en el mercado de servidores dedicados a empresas era casi total hasta que apareció su principal competencia, Microsoft SQL Server.

Sus características a destacar son:

- Soporte de transacciones.
- Garantiza una gran estabilidad.

- Permite crear sistemas escalables.
- Es multiplataforma.

Su mayor defecto es su enorme precio, que según la versión y licencias puede ascender hasta varios miles de euros. Otro aspecto criticado por expertos, es la seguridad de la plataforma y las políticas de actualización de seguridad, que fue cambiada a principios del 2005 y que permiten una mayor exposición de los usuarios. En los parches de actualización de seguridad de 2005 fueron arreglados 22 fallos de seguridad que eran públicamente conocidos, algunos de ellos tenían más de dos años de antigüedad [KK14].

Las ventajas de este sistema es que está muy extendido a nivel mundial y además puede ejecutarse en todas las plataformas. Es, quizás, la base de datos relacional más probada para grandes instalaciones. Puede manejar una gran cantidad de datos y funcionar bien bajo grandes cargas. Además, hay buen soporte y documentación para clientes.

En cuanto a sus desventajas, es caro. Por lo tanto, si no necesitas dar soporte a una gran cantidad de datos y transacciones por segundo, debes considerar seriamente una opción más económica como MySQL o una base de datos noSQL como MongoDB.

Microsoft Access

Microsoft Access o Microsoft Office Access es un sistema de base de datos creado para individuos y pequeñas y medianas empresas que desean capturar, administrar e informar datos de manera profesional. Combina el motor relacional de bases de datos de Microsoft Jet con una interfaz gráfica de usuario y herramientas de desarrollo *software*. Forma parte de la *suite* de aplicaciones de Microsoft Office. Es conocido por ser una opción líder para los administradores de datos profesionales que requieren formas avanzadas de informar sus datos de proyectos. Dado que la programación en Microsoft Access no es complicada, la gente sin conocimiento de programación puede crear bases de datos potentes de manera avanzada con esta herramienta [Ben12].

En la Tabla 3.2, que esta a continuación, se analizan las ventajas, limitaciones y características de Microsoft Access:

Ventajas	Desventajas
Fácil de usar e instalar. La interfaz de usuario es muy intuitiva.	Microsoft Access es útil para departamentos individuales o sectores de negocios pequeños y medianos. Cualquier sector cuyo uso supere los 2 GB se encontrará limitaciones.
Fácil de integrar. Funciona bien con todos los programas de desarrollo *software* en Windows y también se compenetra bien con los de otras plataformas.	SQL para MS Access no es tan robusto como para MS SQL Server u Oracle entre otros.
.NET-Friendly. Access es una buena opción para los usuarios que planean desarrollar software utilizando .NET	Toda la información de la base de datos se guarda en un archivo. Esto limita las opciones y la forma de utilizar datos, ralentiza informes, consultas y formularios. Su rendimiento se vuelve lento a medida que el usuario escala el tamaño de los datos. Los datos multimedia pueden agotar rápidamente el espacio limitado de MS Access.
Es uno de los SGBD más populares en el mundo.	Es difícil publicar archivos a partir de archivos estáticos.
Buena capacidad de almacenamiento	
Soporte multiusuario: aproximadamente diez usuarios en una red pueden usar una aplicación Access.	Límite multiusuario: el límite técnico es de 255 usuarios simultáneos, pero el límite del mundo real es de 10 a 80 (según el tipo de aplicación).
Microsoft Access facilita la importación de datos.	

Cuadro 3.2: Ventajas y desventajas de Microsoft Access.

PostgreSQL

PostgreSQL, también conocido como Postgres, es un sistema gestor de bases de datos relacional orientado a objetos (ORDBMS) que le da un gran énfasis al cumplimiento de los estándares. Como servidor de bases de datos, su principal función es almacenar datos de forma segura y devolver estos datos cuando existan peticiones de otras aplicaciones *software* que los requieran. Puede manejar cargas de trabajo desde aplicaciones pequeñas hasta grandes aplicaciones orientadas a Internet con muchos usuarios conectados de manera concurrente. Para los servidores MacOS, PostgreSQL es el SGBD predeterminado aunque también esta disponible para Microsoft Windows y Linux [Gro18].

Como otros muchos proyectos, PostgreSQL es de código abierto y su desarrollo no se lleva a

cabo por una empresa o persona, sino que es realizado por una comunidad de desarrolladores que trabajan de forma desinteresada y libre. Dicha comunidad se llama PostgreSQL Global Development Group.

Los *Pros* de este sistema de bases de datos son su motor de administración de bases de datos es escalable y puede manejar terabytes de datos, soporta JSON, hay una gran variedad de funciones predefinidas y varias interfaces están disponibles.

Los puntos negativos son que PostgreSQL no tiene un rastreador de fallos (*bug tracker*), lo que dificulta el conocimiento del estado de los fallos, la documentación puede que este incompleta y necesites buscar por Internet, a la hora de realizar la configuración es confuso y la velocidad puede verse afectada durante operaciones masivas o consultas de lectura a gran escala.

El uso se considera ideal para organizaciones con un presupuesto limitado que desean tener la capacidad de seleccionar su interfaz y usar JSON [OH12].

DB2

IBM DB2 ha sido desarrollado por IBM y contiene varios productos para servidores de base de datos. Estos productos dan soporte a modelos de datos relacionales, aunque en los últimos años, algunos de estos productos han extendido su soporte a características relacionales orientadas a objetos y estructuras no-relacionales (NoSQL) como JSON y XML [MM02].

Este sistema esta hecho específicamente para servidores iSeries de IBM aunque la versión *workstation* funciona en Windows, Linux y Unix.

La versión actual de DB2 para LUW (Linux/Unix/Windows) es la 11.1, por lo que ofrece ciertas mejoras. Una de ellas, en particular, ha sido la mejora de la función *BLU Acceleration*, por la cual hace que este motor de base de datos trabaje más rápido mediante la tecnología de omisión de datos. La omisión de datos tiene el objetivo de mejorar la velocidad de sistemas con más datos de los que caben en memoria. Esta última versión también mejora las funciones de recuperación de desastres, compatibilidad y análisis [IBM18b].

Las ventajas de usar DB2 son:

- BLU Acceleration puede aprovechar al máximo los recursos disponibles para bases de datos enormes.
- Se puede hospedar desde la nube, un servidor físico o ambos al mismo tiempo.
- Se pueden ejecutar múltiples procesos a la vez usando el Programador de tareas.
- Los códigos de error y los códigos de salida pueden determinar qué procesos se ejecutan a través del Programador de tareas.

Se recomienda el uso de esta herramienta para organizaciones grandes, que necesitan aprovechar al máximo los recursos disponibles y manejar grandes bases de datos, debido a que los costes están fuera del alcance de organizaciones pequeñas.

DB2 Express-C

DB2 Express-C es una versión gratuita para descargar, usar y edición redistribuida del servidor de datos IBM DB2. Esta versión tiene las características de una base de datos XML y de un sistema gestor de base datos relacional. Esta limitada a dos núcleos de la CPU, 16 GB de RAM, y el tamaño de la base de datos a 15 TB. No tiene soporte para empresas ni parches para arreglar fallos. Además, DB2 Express-C no tiene límite en el número de usuarios [IBM18a].

El 30 de enero de 2006 IBM anunciaba una versión especial y gratuita de la edición DB2 Express llamada DB2 Express-C. Esta edición fue creada a partir de la versión 8.2 de IBM DB2. Después, DB2 Express-C ha sido creado para nueva versión que ha tenido DB2.

DB2 Express-C, antes de la versión 10.5, no tenía límite en la cantidad de datos que podía almacenar en la base de datos y a partir de la versión 10.5 y posteriores la limitación es de 15 TB. El motor de la base de datos no limita el número de usuarios conectados de manera concurrente. IBM ofrece versiones nativas de DB2 Express-C de 64-bit para todas las plataformas compatibles, aunque sigue disponible la versión de 32-bit para algunos entornos. Las aplicaciones remotas se pueden conectar a DB2 Express-C sobre TCP/IP v4, TCP/IP v6 o tuberías nombradas. El motor de base de datos de DB2 Express-C esta limitado para utilizar como máximo 16 GB de RAM (incrementado desde la versión 10.5, que era 4 GB) y dos núcleos de la CPU [BZ07].

El conjunto de características de DB2 Express-C es muy parecido al de la edición DB2 Express (versión de pago), pero con la principal diferencia de que Express-C tiene un límite de CPU y de memoria menores.

DB2 Express-C no está permitido para su uso en entornos de alta disponibilidad como los que implican replicación, activo-pasivo o clúster de disco compartido. Las actualizaciones que el *software* de DB2 vía parches no están disponibles para Express-C, por lo que para actualizar debemos eliminar primero el existente DB2 Express-C y reemplazarlo con una nueva versión de si mismo.

SQLite

SQLite es una librería pequeña, no necesita configuración, con un mantenimiento fácil, hecha a medida, adaptable, segura, orientada a transacciones y basada en SQL para la gestión de sistemas de bases de datos relacionales. SQLite no es un motor de base de datos cliente-servidor, está adaptado e incluido en el programa final. Es elegido popularmente para sistemas con almacenamiento local/-cliente de aplicaciones como navegadores web. Podría decirse que es el motor de base de datos más usado, ya que es utilizado actualmente por varios de los navegadores web más utilizados, sistemas operativos y sistemas integrados (como dispositivos móviles), entre otros [Con18].

SQLite cumple con las garantías de ACID (*Atomicity, Consistency, Isolation, Durability*) e implementa la mayoría de los estándares de SQL, utilizando una sintaxis de SQL de tipado dinámico y débil que no garantiza la integridad del dominio.

SQLite almacena una base de datos entera en un único archivo nativo que contiene todas las tablas e índices. Organiza todas las tablas en árboles-B^+ separados y los índices en árboles-B separados.

Está desarrollado completamente usando el lenguaje de programación ANSI C. Tiene un fácil

mantenimiento, es razonablemente rápido y tiene un sistema relacional para la base de datos. Las características que destacan para que sea recomendable y de calidad son [Hal16]:

- **Cero configuración:** No necesitas tener ninguna instalación adicional o realizar otros pasos antes de inicializar SQLite. No hay archivos de configuración para diferentes comportamientos. La base de datos no necesita ninguna administración. Puedes descargar el código fuente de la página, compilarlo y empezar a usarlo.
- ***Embeddable*:** No necesitas separar el proceso servidor dedicado a SQLite. La librería permite incluirla en tus aplicaciones.
- **Interfaz sencilla:** Provee un entorno SQL para manipular bases de datos. Tiene un conjunto de llamadas a funciones de la API para usar SQL dinámico.
- **Soporte transaccional:** SQLite admite las propiedades ACID transaccionales principales, conocidas como atomicidad, consistencia, aislamiento y durabilidad. No requiere acciones de los usuarios de la base de datos o administradores cuando un proceso falla, el sistema se cae o hay un fallo de potencia para recuperar la base de datos. Cuando SQLite lee la base de datos automáticamente realiza acciones de recuperación.
- **Seguro ante amenazas:** SQLite es una librería *thread-safe*. Muchos hilos pueden acceder a la misma o diferentes bases de datos de manera concurrente.
- **Ligera:** es una librería con un tamaño pequeño, sobre 350 KB. Incluso se podría reducir desactivando algunas características.
- **Personalizabilidad**: Suministra un buen *framework* en el cual puedes definir y usar funciones SQL a medida, añadir funciones, etc.
- **Unicode:** Soporta los estándares de codificación de texto basados en UTF-8 y UTF-16. UTF-16 soporta tanto formatos *little* como *big-endian*.
- **Multiplataforma:** SQLite funciona en Linux, Windows, iOS y Mac OS X, entre otros. También funciona en sistemas como Android, Symbian, Palm y VxWroks.

La Figura 3.8 representa un esquema genérico de una aplicación SQLite. La librería está incrustada en el espacio asignado para procesos de la aplicación, y una parte del proceso del espacio del montículo (*heap*) es usado para almacenar datos de la ejecución de SQLite. Además, SQLite utiliza el proceso de la pila cuando se llama a las funciones de la API.

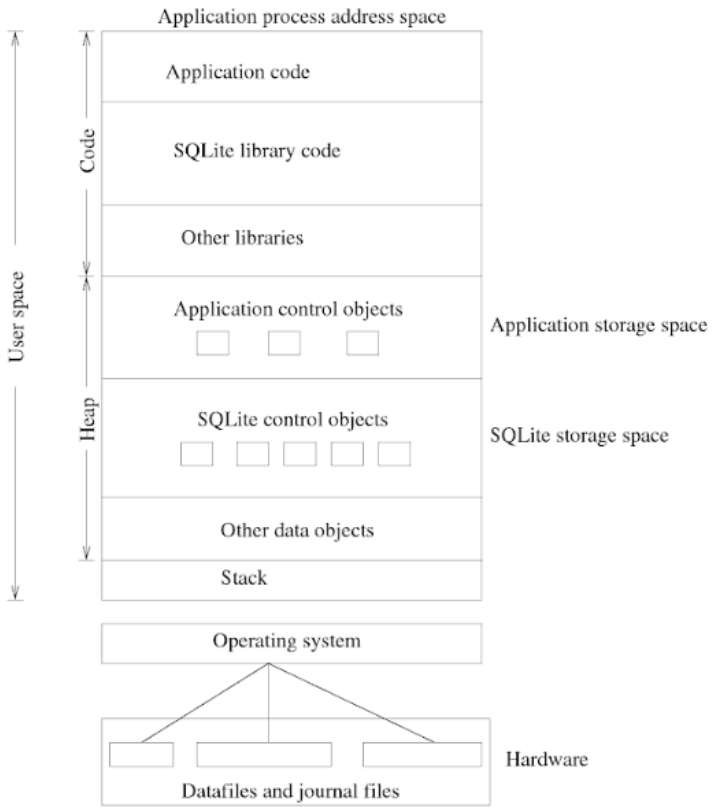


Figura 3.8: Aplicación genérica de base de datos usando la libreria de SQLite.

Apache Derby

Es un sistema gestor de base de datos relacional, desarrollado por *Apache Software Foundation*. Apache Derby puede ser integrado en programas Java y usado para procesamiento de transacciones en linea [Pro17].

Apache Derby se originó en Cloudscape Inc, Oakland, California, en una *start-up* fundada en 1996 por Nat Wyatt y Howard Torf con el fin de desarrollar tecnología de base de datos en Java. La primera *release* fue llamada JBMS, en 1997. Posteriormente, el producto pasó a llamarse Cloudscape y las publicaciones pasaron a ser cada seis meses.

En 1999 Informix Software Inc, adquirió la empresa. En 2001 IBM se hizo con los activos de la base de datos de Informix Software además de Cloudscape. Pasó a llamarse IBM Cloudscape y las versiones continuaron, sobre todo centrándose en el uso integrado con los productos Java de IBM y los *middleware*.

En el año 2004 IBM cedió su código a Apache Software Foundation. Con la publicación de la versión 6 de Java en el 2006, Sun juntó a Derby en el mismo paquete como Java DB. La base de datos Java DB es la distribución compatible de Apache Derby de Oracle [PCZS10].

Sus características más destacadas son: Derby soporta SQL92 y la mayoría de SQL99, usa la sintaxis SQL de DB2, el peso de la librería es de 3,5 mb aproximadamente, su instalación, despliegue y uso son muy sencillos, soporta procedimientos, cifrado y compresión.

MongoDB

MongoDB es una base de datos NoSQL ágil y escalable. El nombre Mongo viene de la palabra *humongous*. MongoDB está basado en el modelo de almacenamiento de documentos NoSQL, en el cual los objetos de datos están almacenados como documentos separados dentro de una colección en lugar del modo tradicional de columnas y filas de una base de datos relacional. Estos documentos estan almacenados como objetos binarios JSON o BSON.

La motivación del lenguaje MongoDB es implementar un almacenamiento de datos que proporciona un alto rendimiento, alta disponibilidad y un escalado automático. MongoDB es extremadamente simple de instalar e implementar. MongoDB ofrece un excelente almacenamiento de *back-end* del sitio web para páginas con un tráfico alto que necesiten almacenar datos tales como comentarios de los usuarios, blogs y otros artículos porque es rápido, escalable y fácil de implementar [Mon18].

A continuación se exponen algunas razones más por las que MongoDB ha llegado a ser la base de datos NoSQL más popular [Day15]:

- **Orientado a documentos:** Los datos estan almacenados en la base de datos en formato muy parecido al que usará tanto el lado del servidor como en los scripts del lado del cliente. Esto elimina la necesidad de transferir datos de filas a objectos y devolverlos.
- **Alto rendimiento:** Es uno de los servicios de bases de datos con mejor rendimiento disponible. Especialmente en el mundo actual, donde mucha gente interacciona con sitios web, en los que se necesita un *back-end* que pueda soportar mucho tráfico.
- **Alta disponibilidad:** El modelo de replicación de MongoDB hace que sea fácil mantener la escalabilidad mientras se mantiene el alto rendimiento.
- **Alta escalabilidad:** La estructura de MongoDB hace que sea fácil de escalar horizontalmente compartiendo los datos entre múltiples servidores.
- ***No SQL injection*:** No es susceptible a SQL *injection* (poner declaraciones SQL en formularios web u otras entradas del navegador que comprometen la seguridad de la base de datos) porque los objetos están almacenados como objetos, sin utilizar cadenas SQL.

3.2 Reconocimiento óptico de caracteres (OCR)

Es un proceso de clasificación de patrones ópticos, contenidos en una imagen digital, correspondientes a caracteres alfanuméricos o de otro tipo. El reconocimiento de caracteres se consigue mediante importantes pasos de segmentación, extracción de características y clasificación (ver Figura 3.9). OCR está teniendo un gran incremento en la atención tanto por parte de investigaciones académicas como por la industria. Desde los tiempos más antiguos, el sueño del hombre es que las máquinas repliquen las funciones de los humanos, una función como puede ser leer documentos con diferentes formas de texto. En las últimas décadas, la posibilidad de que las máquinas puedan leer ha crecido, de ser un sueño a ser realidad a través del desarrollo de sofisticados y robustos sistemas de OCR.

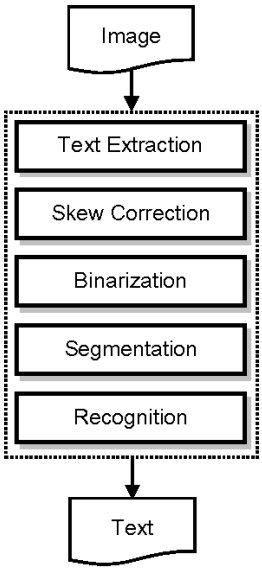


Figura 3.9: Diagrama de un sistema de Reconocimiento Óptico de Caracteres.

La tecnología OCR nos permite convertir diferentes tipos de documentos como documentos a papel escaneados, archivos PDF o imágenes capturadas por una cámara digital en datos editables y buscables. Los sistemas OCR han llegado a ser unas de las aplicaciones tecnológicas con mayor éxito en campos de reconocimiento e inteligencia artificial. Aunque existen muchos sistemas comerciales para realizar OCR para una amplia variedad de aplicaciones, las máquinas disponibles todavía no están preparadas para competir con las habilidades humanas debido a que no cumplen con el nivel deseado de precisión [ACG17]

OCR pertenece a la familia de técnicas que realizan la identificación automática. La identificación automática es el proceso donde un sistema de reconocimiento identifica automáticamente objetos, recolecta información sobre ellos y registra la información directamente en los sistemas, es decir, sin participación humana.

A continuación, se van a analizar en detalle las librerías OCR con características más adecuadas para el desarrollo e integración en el proyecto.

3.2.1 Tesseract Android Tools

Es una de las opciones más conocidas para el desarrollo de aplicaciones con reconocimiento de caracteres debido a que es considerada como uno de los motores OCR de código abierto con mayor precisión disponibles actualmente. Originalmente fue desarrollado por Hewlett Packard entre 1985 y 1994, con algunos cambios para la portabilidad a Windows en 1996 y modificaciones a C++ en 1998. En 2005, Tesseract es un proyecto de código abierto por HP y desde 2006 esta desarrollado por Google hasta la actualidad.

La principal característica, que podría ser una ventaja en cierto tipo de sistemas, es que no necesita de conexión a Internet aunque para que pueda funcionar sin conexión es necesario que estén descargados los paquetes de idiomas que necesita, lo que penaliza en el almacenamiento.

La versión 3.0.1 de Tesseract OCR puede ser completamente entrenada para soportar idiomas no estándar: juegos de caracteres y glifos. El proceso de entrenamiento esta descrito en el manual y se puede preparar para que el proceso sea realizado de forma automática. La información entrenada necesita ser recolectada y almacenada de una forma específica. Tesseract puede reconocer más de 100 idiomas [Tes18].

3.2.2 ABBYY Mobile OCR Engine

Es una librería muy potente, eficiente y compacta. Tiene soporte tanto para Android como para iOS, lo que permite a los desarrolladores tener un amplio mercado en sus aplicaciones. Puede reconocer hasta 62 idiomas, incluyendo el latín, chino, japonés, coreano y griego [ABB18].

Las funcionalidades y características que aporta esta librería son:

- El pre-procesamiento de la imagen detecta la orientación del documento.
- Soporta texto con guiones.
- Muestra el nivel de confianza del texto reconocido, lo que permite a desarrolladores establecer criterios flexibles para la implementación de funciones de corrección y verificación.
- El algoritmo de análisis de datos esta optimizado, lo que significa que al analizar la imagen descarta toda la información innecesaria, dando lugar a un aumento en la precisión de reconocimiento.
- Realiza una revisión ortográfica durante el reconocimiento del texto que mejora la calidad de la salida obtenida.
- Permite configurar manualmente qué bloque de texto quieres que reconozca.
- Los recursos utilizados son muy bajos, la librería esta optimizada y ocupa muy poca memoria.
- Las operaciones de reconocimiento se realizan en paralelo mediante *multithreading*. El número de hilos para reconocimiento es 4, aunque es posible modificar este número.
- Puede procesar tarjetas de visita, obteniendo nombre, apellidos, título, teléfono, e-mail, dirección, etc. Esta tecnología puede procesar tarjetas en 26 idiomas.
- Permite el reconocimiento de código de barra.
- El reconocimiento de texto esta disponible para 62 idiomas. 23 idiomas principales con diccionario y 39 idiomas adicionales con caracteres en latín, cirílicos o griegos.
- Tiene dos modos de reconocimiento de texto, un modo rápido, para imágenes con buena calidad, y un modo completo, para imágenes de baja calidad que necesitan mayor procesamiento.

Su principal desventaja es que es una librería de pago.

3.2.3 Asprise OCR

Es una librería desarrollada por Asprise en múltiples lenguajes como C# .NET, VB .NET, Python, C/C++, Delphi Pascal y Java, lo que permite trabajar con ella en numerosas plataformas. Muchos investigadores han utilizado Asprise OCR junto con ABBYY para las comparativas de rendimiento. En una prueba de 200 imágenes Asprise OCR obtuvo un acierto del 95 % [Asp18].

Lo más destacable de esta librería es:

- Alto nivel de acierto incluso en imágenes con baja calidad.
- El tiempo de cómputo es muy corto e incluso se puede mejorar con *multi-threading*.
- Soporta más de 20 idiomas.

3.2.4 Google Cloud - Vision API

Es una de las múltiples librerías que ofrece Google, contiene una gran cantidad de funciones como detección de etiquetas, detección de caras, detección de contenido explícito, detecta los atributos generales de la imagen, busca imágenes similares en Internet, detecta logotipos de productos muy conocidos dentro de una imagen, detecta estructuras artificiales y naturales muy famosas dentro de una imagen y la que se ha utilizado en el proyecto, detecta y extrae texto de una imagen (OCR) [Goo18c].

Los puntos fuertes de esta librería son:

- Sencilla implementación
- Procesa la imagen en un servidor, por lo tanto requiere Internet, esto favorece a la facilidad de mejora sin tener que modificar el software donde se integra.
- Rápido procesamiento y respuesta del servidor.
- Es una librería que no consume prácticamente nada de almacenamiento, lo que hace que sea ultraligera

En contra, como punto negativo, es que la versión gratuita permite 1000 usos al mes.

3.2.5 IBM Cloud - Visual Recognition API

IBM trabaja de la misma manera que Google ofreciendo numerosas librerías que funcionan en la nube. Esta librería es conocida por su eficiencia y calidad de acierto. Permite el análisis de escenas, objetos, rostros y otros contenidos. Tiene una gran similitud en sus características con la librería de Google, también funciona en la nube por lo que es muy ligera y gran parte del trabajo lo realiza el servidor que procesa la imagen y genera el texto obtenido [IBM18c].

Como negativa, permite a los usuarios hacer 7.500 llamadas de API gratis, por lo que no es gratuita a partir de cierto uso.

3.3 Procesamiento del lenguaje natural

El Procesamiento del Lenguaje Natural (PLN) es un tema amplio centrado en el uso de ordenadores para analizar lenguajes naturales. Se dirige a áreas como procesamiento del habla, extracción de relaciones, categorización de documentos y combinación de texto. Sin embargo, estos tipos de análisis están basados en conjuntos de técnicas fundamentales como la tokenización, detección de frases, clasificación y extracción de relaciones [Kur17].

¿Qué es el PLN?

Una definición formal de PLN frecuentemente incluye fraseología a pesar de que es un campo de estudio que usa la ciencia de la computación, la inteligencia artificial y conceptos lingüísticos formales para analizar el lenguaje natural. Una definición menos formal sugiere que es un conjunto de herramientas usado para obtener información significativa y útil del lenguaje natural en recursos como páginas web y documentos de texto.

Significativa y útil implica que tiene algún valor comercial, a pesar de que muchas veces es usada para problemas académicos. Esto puede ayudar fácilmente a motores de búsqueda. Una petición de un usuario es procesada usando técnicas de PLN para generar una página de resultados que el propio usuario puede utilizar. Los motores de búsqueda modernos han tenido mucho éxito en este sentido. Las técnicas de PLN también son usadas en sistemas de ayuda automática y como ayuda en sistemas de peticiones complejos como es tipificado en el proyecto Watson de IBM.

Cuando trabajamos con un idioma, frecuentemente, nos tropezamos con los términos, la sintaxis y la semántica. La sintaxis de un idioma se refiere a las reglas que controlan una estructura de oraciones válida. Por ejemplo, una estructura de oración común en inglés comienza con un sujeto seguido de un verbo y luego un objeto como *"Tim hit the ball"*. No estamos acostumbrados a encontrarnos un orden inusual como *"Hit ball Tim"*. Aunque las reglas de la sintaxis para el inglés no son tan rigurosas para los lenguajes de ordenadores, aun así esperamos una oración que cumple las reglas básicas de la sintaxis [Ree15].

Hay muchas herramientas disponibles para poder realizar tareas de PLN. A continuación, nos centraremos en las de lenguaje Java que soportan PLN y que cumplen los requisitos necesitados para realizar el proyecto.

3.3.1 Google Cloud Natural Language API

Esta API revela la estructura y el significado de textos mediante modelos de aprendizaje máquina potentes en una librería REST muy fácil de usar. Te permite extraer información sobre gente, sitios, eventos y muchas más opciones, que estén mencionadas en textos, documentos, noticias o *posts* de blogs. También es usado para entender el sentimiento sobre un producto en las redes sociales o para intentar entender conversaciones de clientes en un centro de atención al cliente o en una aplicación de mensajes. Puedes analizar el texto subido en una petición o integrarlo con el almacenamiento de documentos en Google Cloud *storage* [GCP18].

La librería Natural Language tiene varias funcionalidades para realizar análisis y anotaciones en el texto. Todos estos niveles de análisis proporcionan información útil para el entendimiento del lenguaje. Estas funcionalidades son [Goo18b]:

- **Análisis de Sentimientos:** inspecciona el texto de entrada e identifica la opinión emocional prevaleciente dentro del texto, especialmente para determinar la actitud del escritor como positiva, negativa o neutral. El análisis de sentimientos está realizado a través del método `analyzeSentiment`.
- **Análisis de Entidades:** localiza en el texto de entrada entidades conocidas (Nombres propios como figuras públicas, puntos de referencia, etc. Nombres comunes como restaurantes, esta-

dios, etc.) y devuelve información sobre esas entidades. El análisis de entidades esta realizado con el método `analyzeEntities`.

- **Análisis de Sentimiento de Entidades:** busca en el texto de entrada entidades conocidas (nombres propios y nombres comunes), devuelve información sobre esas entidades e identifica la opinión emocional prevaleciente de la entidad dentro del texto, especialmente para determinar la actitud del escritor hacia la entidad como positiva, negativa o neutral. El análisis es realizado por el método `analyzeEntitySentiment`.
- **Análisis Sintáctico:** extrae información lingüística, separando el texto de entrada en varias sentencias y *tokens* (generalmente palabras), porporcionando información adicional de esos *tokens*. El método `analyzeSyntax` es el encargado de realizar el Análisis Sintáctico.
- **Clasificación de Contenido:** analiza el contenido del texto y devuelve la categoria de este contenido. La clasificación de Contenidos se realiza mediante el método `classifyText`.

Cada llamada a la API también se encarga de detectar y devolver el idioma, si el idioma no esta especificado en la invocación de la petición inicial. Además, si también se pueden realizar varias operaciones sobre el mismo texto con una sola llamada a la API, el método `annotateText` puede ser utilizado para realizar el análisis de sentimientos y de entidades.

3.3.2 IBM - Watson Natural Language Understanding

Es capaz de analizar texto para obtener metadatos del contenido como por ejemplo conceptos, palabras clave, determinar el sentimiento, las relaciones, entidades, categorías, emoción y roles semánticos [IBM18d].

El uso es bastante sencillo debido a que realiza el análisis en sus propios servidores, lo que requiere acceso a Internet. Esto agiliza el coste operacional en el cliente y reduce el consumo de almacenamiento. *Natural Language Understanding* puede analizar textos hasta en 13 idiomas [SVS17]. Como único dato negativo, es que para un uso comercial o simplemente más que un uso personal no es gratuito.

3.3.3 Apache OpenNLP

La biblioteca de Apache es un *toolkit* basado en el aprendizaje automático para el procesamiento de texto en lenguaje natural. El proyecto OpenNLP es desarrollado por voluntarios y es de libre uso bajo la licencia de Apache.

Es una librería que recibe mejoras de manera más lenta que las más conocidas como IBM o Google. Para que funcione correctamente se necesitan archivos modelo pre-entrenados, por lo que necesita cierto almacenamiento y funciona sin conexión a Internet [Apa17].

Las principales funcionalidades son:

- Puede detectar que un carácter de puntuación marca el final de una oración y no puede identificar límites de oraciones basados en el contenido de la oración.
- Puede segmentar una secuencia de caracteres de entrada en *tokens*. Los *tokens* son generalmente palabras, signos de puntuación, números, etc.

- Detecta el idioma en el texto analizado.
- Permite extraer entidades nombradas.

3.3.4 Stanford CoreNLP

Este proyecto esta desarrollado por el grupo de procesamiento del lenguaje natural de la Universidad de Stanford en California. Todas las distribuciones de *software* compatibles están escritas en Java aunque a lo largo del tiempo algunas personas han decidido ayudar con enlaces o traducciones a otros lenguajes de programación. Como resultado, gran parte de su *software* puede ser utilizado en Python (o Jython), Ruby, Perl, Javascript, F#, .NET y JVM.

Este *software* proporciona procesamiento del lenguaje natural estadístico, de aprendizaje profundo y basado en reglas para problemas importantes de lingüística computacional, que puede ser utilizados en aplicaciones con tecnologías basados en el lenguaje humano. Actualmente este *software* es utilizado en empresas, universidades y en gobiernos. Su algoritmo procesa texto en árabe, chino, francés, alemán y español. Como la librería de Apache, funciona completamente en local, por lo tanto, requiere mayor almacenamiento que las que funciona en la nube [Gro14].

3.3.5 ParallelsDots - Text Analysis API

El análisis de texto es obtenido a partir de información de alto nivel a través de patrones y tendencias establecidos en un texto. Para ello utilizan técnicas avanzadas como *Deep Learning*, LSTM, *Transfer Learning*, etc. El uso de estas técnicas proporciona un alto rendimiento en comparación con otras más tradicionales. Se puede aplicar como solución para resolver problemas complejos, como *chatbots*, análisis de redes sociales, automatización de procesos, etc [Par18].

La librería de ParallelsDots es muy completa, las funcionalidades que aporta son:

- Análisis de sentimientos: obtiene el significado contextual del texto que identifica y extrae información subjetiva en el material fuente.
- Clasificación de texto: puede ayudar a la comprensión del comportamiento de un usuario mediante la categorización de conversaciones en redes sociales, comentarios y otras fuentes web. Muchos motores de búsqueda, periódicos o portales de comercio electrónico categorizan su contenido para facilitar la búsqueda y la navegación.
- Extrae palabras clave: es capaz de generar una lista de palabras clave para poder obtener un contexto mas acertado.
- Análisis semántico: puede ayudar a los usuarios a agrupar artículos similares al comprender la relación entre diferentes contenidos y agiliza la investigación eliminando contenidos de texto redundantes.
- Análisis de emociones: algunas veces los tres tipos de sentimientos (positivo, negativo y neutral) no son suficientes para entender los matices con respecto al tono subyacente de una oración. El análisis de emociones dice si la emoción subyacente detrás de un mensaje es: Feliz, Triste, Enojado, Emocionado, Sarcasmo o Miedo.
- Análisis sintáctico: realiza un análisis de las relaciones sintácticas que se establecen entre los

pares de palabras que la componen.

- Detección del idioma: evalúa la entrada de texto y devuelve un porcentaje con el idioma identificado en dicha entrada.
- Detección de contenido ofensivo o abusivo.
- Análisis de intenciones: este clasificador dice si la intención subyacente detrás de una oración es noticias, marketing, comentarios o consultas

3.3.6 MAchine Learning for LanguagE Toolkit (MALLET)

MALLET es un paquete basado en Java para el procesamiento estadístico del lenguaje natural, clasificación de documentos, *clustering*, *topic modeling*, extracción de información y otras aplicaciones de aprendizaje automático para texto [MAL18].

Las virtudes de MALLET más destacadas son:

- Incluye sofisticadas herramientas para la clasificación de documentos: tiene rutinas muy eficientes para convertir texto a ciertas características, una amplia variedad de algoritmos (incluidos Naïve Bayes, Maximum Entropy y Decision Trees) y es capaz de evaluar el rendimiento del clasificador.
- Es capaz de extraer entidades con nombres del texto.
- Tiene una herramienta muy eficiente para el *topic modeling*. Los *topic models* son útiles para analizar grandes colecciones de texto sin etiquetar.
- Muchos de los algoritmos que utiliza dependen de la optimización numérica.
- Es un *toolkit* de código abierto.

Como datos negativos, destacar que esta poco actualizado, no funciona en la nube y tiene una mayor complejidad que los vistos anteriormente.

3.3.7 Natural Language Toolkit (NLTK)

NLTK es una de las plataformas más avanzadas para construir programas en Python que trabajen con datos del lenguaje humano. Proporciona interfaces fáciles de usar para más de 50 recursos corporales y léxicos como WordNet, junto con un conjunto de bibliotecas de procesamiento de texto capaces de clasificar, *tokenizar*, derivar, etiquetar y analizar y realizar un razonamiento semántico [NLT09].

3.4 Comunicación Cliente-Servidor(*Web Services*)

A pesar de que el término *web service* tiene varios significados, es impreciso y evoluciona. Echando un vistazo a algunas de sus típicas características es suficiente para que un desarrollador pueda escribir código de un servicio web y un cliente, también conocido como consumidor. Como sugiere el nombre, *web service* es un tipo de aplicación para la web, hecha para ser enviada por HTTP (*Hyper Text Transport Protocol*). Un *web service* es una aplicación distribuida, cuyos componentes pueden ser desplegados y ejecutados en distintos dispositivos [Kal09].

Algunas características que distinguen los servicios web de otros sistemas *software* distribuidos son:

- **Infraestructura abierta:** los servicios web se implementan usando protocolos estándar de la industria, independientes de los proveedores, como HTTP y XML, que estan en todos lados y son muy bien conocidos. Los WS pueden utilizar recursos de red, un formato de datos y la seguridad, entre otras infraestructuras existentes, lo que permite reducir el coste y facilita la interoperabilidad entre servicios.
- **Transparencia en el lenguaje:** los servicios y sus clientes pueden comunicarse entre ellos incluso si utilizan diferentes lenguajes de programación. Lenguajes como C/C++, C#, Java, Perl, Python y Ruby, entre otros.
- **Diseño modular:** están creados con la intención de ser modulares en el diseño, por lo que nuevos servicios pueden ser generados mediante la integración y extensión de servicios existentes.

3.4.1 Tipos de *Web Services*

A nivel técnico, los servicios Web se pueden implementar de varias maneras. Refiriéndonos a esto, podemos hacer diferenciación de dos tipos de servicios Web, también conocidos como los servicios web "grandes", estos son los servicios Web SOAP y los servicios web REST. Veamos a continuación que características tiene cada uno.

Simple Object Access Protocol (SOAP)

SOAP se hizo popular a partir del año 2000, cuando XML era tendencia. Fue creado por Microsoft, IBM y otros. Define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Por ejemplo, en un escenario típico, llamado "*request/response message exchange pattern (MEP)*", la biblioteca SOAP del cliente envía un mensaje SOAP como una petición de servicio, y la biblioteca SOAP correspondiente al servicio web envía otro mensaje SOAP como respuesta del servicio (ver Figura 3.10). El formato de datos XML es usado por varios sistemas distribuidos para comunicarse con otros. SOAP tiene una implementación muy compleja, además las críticas a SOAP se centran en lo voluminosas que son sus peticiones HTTP como dato negativo [Yel17].

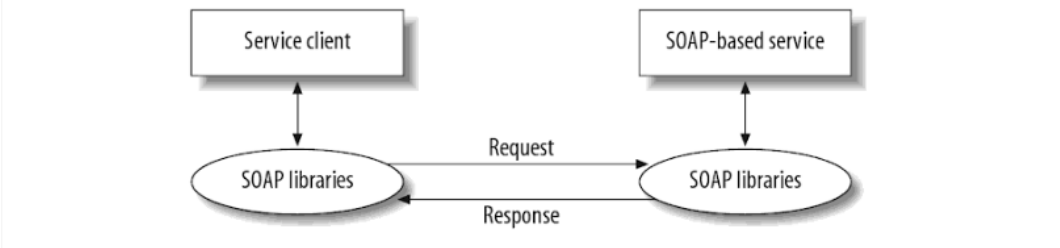


Figura 3.10: Arquitectura típica de un servicio web basado en SOAP.

Una petición SOAP normalmente consiste en estos tres componentes básicos: *Envelope, Header y Body* (Ver Figura 3.11).

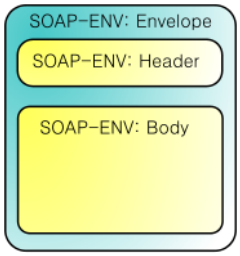


Figura 3.11: Estructura de un mensaje SOAP.

Las principales ventajas son:

- SOAP puede utilizarse mediante usuario anónimo o con autentificación.
- Normalmente se utiliza los protocolos de transporte HTTP o SMTP pero es posible hacerlo con cualquiera.
- Se puede implementar en cualquier lenguaje y no tiene restricción de plataforma.
- Con HTTP es óptimo para sistemas que necesitan escalabilidad.
- Gracias al uso de XML, tiene una gran interoperabilidad.

Como puntos negativos, es que es más lento que otros *middlewares* y que depende del WSDL.

REpresentational State Transfer (REST)

REST no es una arquitectura, más bien es un conjunto de restricciones que crean un estilo arquitectural de *software*, el cual puede ser usado para aplicaciones de sistemas distribuidos. Un gran desafío para las aplicaciones distribuidas se atribuye a la diversidad de sistemas en una empresa que ofrece silos de información comercial (ver Figura 3.12) [Bal17].

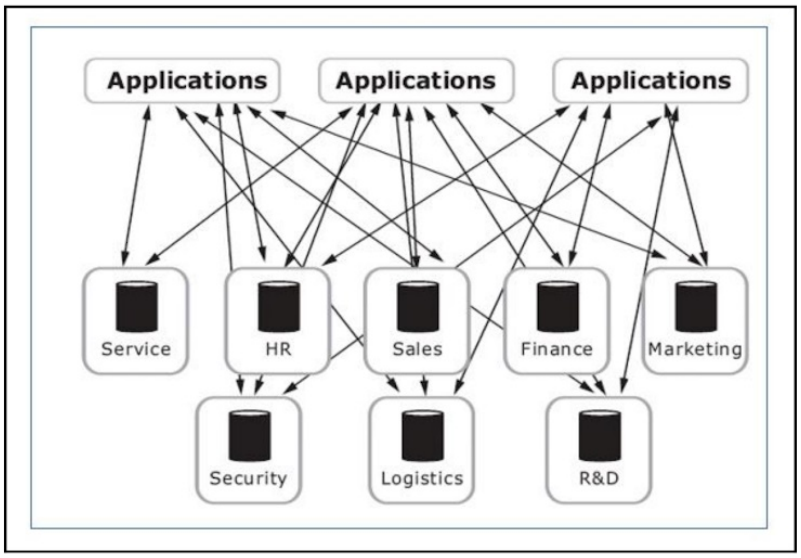


Figura 3.12: Diagrama de diversidad de sistemas que puede ofrecer una empresa con información comercial.

A menudo, una empresa demanda acceso simplificado y actualizaciones de la información que hay en diferentes sistemas. Roy Thomas Fielding, que es uno de los principales autores de la especificación HTTP y ha hecho grandes aportaciones en Arquitectura de Redes, llegó a REST evaluando todos los recursos de red y las tecnologías disponibles para crear aplicaciones distribuidas. Observó, que sin ninguna restricción, se puede terminar desarrollando aplicaciones sin reglas o limites que, al final, son difíciles de mantener o extender.

3.4.2 Comparación REST vs SOAP

La principal ventaja de SOAP no es otra que la buena definición de las interfaces de los servicios (mediante WSDL), esto da la facilidad de poder realizar test y depurar antes ejecutar la aplicación. Por otro lado, con REST optimizaríamos la escalabilidad, lo que para el sistema a desarrollar nos interesa, reduciendo el consumo de recursos en sus operaciones debido al reducido número de operaciones y a la unificación.

En el Cuadro 3.3, se puede ver una comparación detallada de las principales características de SOAP y REST. Como se comenta en el apartado 3.4.1, REST trabaja con el protocolo de comunicación HTTP y, la mayoría de veces, con formatos XML o JSON para la transmisión de datos. Cada dirección (URL) representa un servicio con el que se pueden ejecutar llamadas a métodos como GET, POST, PUT o DELETE. Por otro lado, SOAP únicamente soporta el uso de XML y cada objeto tiene unos métodos que son definidos por el desarrollador. En conclusión, REST es mucho más fácil de configurar, óptimo para sistemas escalables y funciona mediante direcciones URL. SOAP tiene un tipado más fuerte, esto permite verificar que las operaciones son correctas más a menudo [Nor15].

	REST	**SOAP**
Orígenes	Fue creado en el 2000 por Roy Fielding. Desarrollado en un entorno académico, este protocolo abarca la filosofía de web abierta.	Fue creado en 1998 por Dave Winer en colaboración con Microsoft. Aborda el objetivo de cumplir las necesidades del mercado empresarial.
Ventajas	Sigue la filosofía de la web abierta. Fácil implementación y mantenimiento. Separación clara de la implementación del cliente y servidor La comunicación no está controlada por una sola entidad. Puede almacenar información el cliente, lo que evita múltiples llamadas. Puede devolver datos en múltiples formatos (JSON, XML, etc).	Sigue un enfoque empresarial formal. Información sobre los objetos se comunica a los clientes. La seguridad y la autorización son parte del protocolo. Funciona sobre cualquier protocolo de comunicación, incluso de forma asíncrona.
Desventajas	Solo funciona sobre el protocolo HTTP. Es difícil hacer cumplir la autorización y seguridad.	Consume una gran cantidad de ancho de banda para la transmisión de metadatos. Es difícil de implementar y no es popular entre los desarrolladores web y móviles. Usa solamente XML.
Cuándo usarlo	Cuando los clientes y servidores operan en un entorno web. Cuando la información sobre los objetos no necesita ser comunicada al cliente.	Cuando los clientes necesitan tener acceso a los objetos disponibles en los servidores. Cuando se desea aplicar un contrato formal entre el cliente y el servidor.

Cuadro 3.3: Comparación REST vs SOAP

3.4.3 RESTful WS - JAX-RS

JAX-RS es una especificación para la implementación de Servicios Web REST en Java. Actualmente, JAX-RS es parte del paquete de Java EE a partir de la versión 6, lo que significa que no requiere configuración en el descriptor de despliegue (web.xml) como en versiones anteriores.

Mientras que REST es una arquitectura que se ejecuta sobre HTTP, RESTful describe el uso de ese paradigma, es decir, se suele utilizar para referirse a los servicios web que ejecutan la arquitectura REST [San09].

Roy Thomas Fielding realizó importantes investigaciones para construir mejores arquitecturas para aplicaciones distribuidas, terminó definiendo las siguientes restricciones que dan significado a un sistema RESTful (ver Figura 3.13) [Yel17]:

- **Cliente-Servidor:** Esta restricción mantiene el cliente y el servidor ligeramente unidos. En este caso, el cliente no necesita saber detalles de la implementación del servidor, y el servidor no debe preocuparse sobre como el cliente usa la información que recibe.
- **Sin estado:** No hay necesidad de mantener la sesión del usuario para el servicio. Dicho de otra manera, cada petición debe ser independiente de otras. Esto mejora la escalabilidad, ya que el servidor no necesita gestionar el estado entre diferentes peticiones que además perjudicaría al tráfico de la red.
- **Uso de Cache:** Da soporte a un sistema de cache. La infraestructura de red debe soportar cache en diferentes niveles. El uso de cache puede evitar el tráfico repetido entre el cliente y servidor.
- **Interfaz uniforme:** Indica que una interfaz genérica puede gestionar todas las interacciones entre el cliente y el servidor de una manera unificada, lo que simplifica y libera la arquitectura. Esta restricción indica que cada recurso usado por el cliente debe tener una dirección única y debe ser accesible a través de una interfaz genérica. El cliente puede llamar a estos recursos usando un conjunto de métodos genéricos.
- **Sistema de capas:** El servidor puede tener múltiples capas para su implementación. Esta arquitectura por capas ayuda a mejorar la escalabilidad mediante un equilibrio de cargas. También mejora el rendimiento mediante las caches compartidas que hay en cada nivel.
- **Código bajo demanda:** Esta restricción es optativa. Indica que la funcionalidad de las aplicaciones del cliente puede extenderse durante la ejecución permitiendo la descarga del código del servidor y ejecutando el código. Por ejemplo, las applets y su código en JavaScript puede ser transferido y ejecutado en el cliente durante la ejecución.

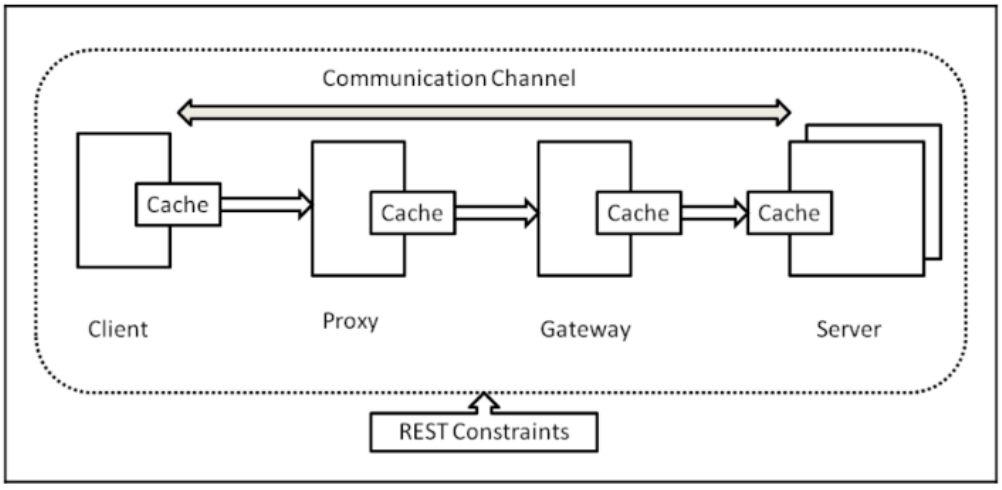


Figura 3.13: Diagrama ilustrativo de la vista arquitectural de alto nivel de un sistema RESTful.

Las implementaciones siguientes de JAX-RS cubren las necesidades para este proyecto, a continuación se va a realizar un análisis de ellas.

RESTEasy

Es un proyecto de JBoss que proporciona varios *frameworks* para ayudar a crear servicios web RESTful y aplicaciones Java RESTful. Es una implementación para la especificación JAX-RS 2.1, una especificación JCP que proporciona una API Java para los servicios web RESTful a través del protocolo HTTP [Mar18].

Los servicios web con RESTeasy son POJO (*Plain Old Java Object*) con la diferencia de que se pueden añadir anotaciones para usar sus diferentes funcionalidades [JBo18].

Los pros de la implementación RESTEasy son:

- Es un producto maduro y estable.
- Es relativamente rapido y confiable.
- Creado sobre la especificación JAX-RS 2.0.
- Tiene una amplia comunidad.
- Esta capacitado para ser utilizado por empresas.
- Tiene el entorno familiar de JBoss.
- Integración de Spring MVC.
- Soporte para Enterprise JavaBeans.
- Excelente soporte de seguridad (OAuth 2.0).

Y los contras son:

- Documentación regular.
- No proporciona muchas herramientas para el *testing*.
- Se basa mucho en Spring.

Restlet

Este proyecto tiene un *framework* ligero a la par que intuible para mapear conceptos REST en clases Java. Se puede utilizar para implementar cualquier tipo de sistema RESTful (no solo servicios web RESTful), y esta demostrado que es muy fiable desde su aparición en 2005.

Restlet directamente modela los conceptos (Recurso, Representación, Conector, Componente, etc.) de la disertación sobre la arquitectura de la web de Roy T. Fielding [RT00].

El proyecto Restlet ha sido influenciado por otras grandes tecnologías Java para el desarrollo de aplicaciones web como los Servlets, Java Server Pages, HttpURLConnection y Struts. Uno de sus objetivos es presentar una vista unificada de la web, adecuado para el uso de aplicaciones en el cliente y en el servidor.

Su filosofía es que la distinción entre cliente HTTP y servidor HTTP no tiene importancia arquitectónica. Una simple pieza de software debe ser capaz de actuar como cliente web y como servidor web, sin usar dos APIs completamente diferentes [Res18].

Los puntos a favor de Restlet son:

- Esta disponible para Java SE, Java EE, Google Web Toolkit, Google AppEngine, Android y entornos OSGi.
- Soporta las versiones antiguas de JAX-RS (como Jersey).
- Ofrece el soporte más avanzado para RESTful.
- La modularidad.
- Actualmente, su desarrollo sigue activo.
- Permite el enlace de URL inteligente y enrutamiento URI completo.

Como únicos datos negativos, es que la curva de aprendizaje no ayuda a nuevos desarrolladores y que su única comunidad abierta y activa es StackOverflow.

Jersey

Jersey es un *framework* para el desarrollo de servicios Web en RESTful con el lenguaje de programación Java. Proporciona soporte para la API de JAX-RS y es una implementación de referencia de JAX-RS (JSR 311 & JSR 339) [Gul13].

Los siguientes componentes forman parte de Jersey [Cor18]:

- *Core Server:* Para construir servicios RESTful basados en anotaciones (jersey-core, jersey-server, jsr311-api).
- *Core Client:* Ayuda en la comunicación con servicios REST (jersey-client).
- Soporta Java Architecture for XML Binding (JAXB).
- Soporta JSON.
- Módulo de integración para Spring y Guice.

Apache Wink

Es un *framework* simple pero sólido que permite crear servicio web RESTful. Esta compuesto por un módulo de servidor y un módulo de cliente para desarrollar y consumir servicios web. El módulo Servidor es una implementación de la especificación JAX-RS (JSR-311) junto con características adicionales para facilitar el desarrollo de servicios web RESTful. El módulo Cliente es un *framework* basado en Java que proporciona funcionalidad para comunicarse con los servicios web RESTful [Apa13]

Apache CXF

Es un *framework* completo y de código abierto para crear servicios usando APIs de programación *frontend*, como JAX-WS y JAX-RS [Apa18].

Las características del diseño de CXF son:

- Existe una separación clara entre los *front-ends*, como por ejemplo JAX-WS y el código del núcleo.
- Se pueden crear clientes y *endpoints* sin hacer uso de las anotaciones, por lo que resulta muy simple.
- El componente de servicios web es incrustable con, por ejemplo, Spring Framework y Geronimo.

Alternativa a JAX-RS: HTTP-RPC

Es un *framework* de código abierto para simplificar el desarrollo de las aplicaciones REST.

Permite a los desarrolladores crear y acceder a servicios web basados en HTTP utilizando una metáfora conveniente, similar a RPC, al tiempo que conserva los principios fundamentales de REST, como no tener estado y el acceso uniforme a los recursos. Este proyecto actualmente tiene soporte para implementar servicios REST en Java y usar servicios en Java, Objetive-C/Swift o JavaScript [GB18].

Proporciona una alternativa muy ligera a los grandes *frameworks* REST Java como JAX-RS, por lo que es una opción ideal para aplicaciones de con pequeñas comunicaciones, como microservicios. Toda la plataforma se distribuye en dos archivos JAR que suman aproximadamente 30 kb de tamaño.

El acceso a los servicios HTTP-RPC es mediante el uso de verbos como GET o POST en un recurso que se tenga como objetivo.

3.4.4 Herramienta de Testing - Advanced REST Client

ARC (*Advanced REST Client*) es un programa de ayuda a los desarrolladores que permite crear y configurar peticiones HTTP. Es una herramienta de testing que permite ahorrar tiempo, no es complicada de usar y permite realizar scripts. Permite realizar conexiones directamente incluyendo la cabecera, el *payload*, etc. Además tiene un historial de peticiones realizadas que facilita mucho el trabajo.

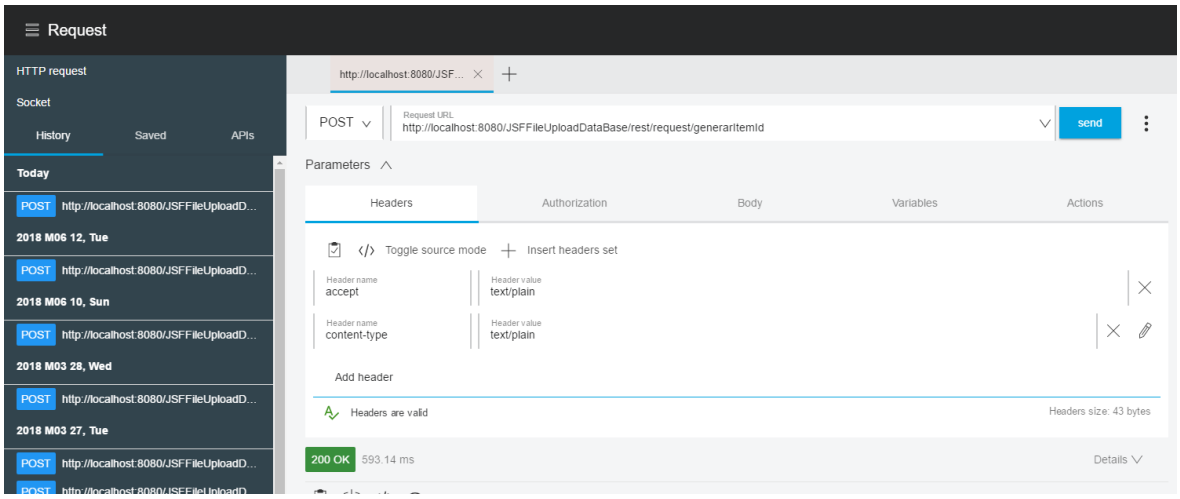


Figura 3.14: Interfaz de usuario de ARC.

3.5 Realidad Aumentada

La realidad aumentada, tal y como la conocemos, se trata de una disciplina bastante nueva, aunque durante los años 60 fue desarrollada la primera interfaz con funcionalidades de RA y la expresión en sí de RA se le asigna a un investigador de Boeing, llamado Tom Caudell, que fue quien acuñó la terminación de RA durante 1990. En 1994 es cuando la idea de RA fue perfectamente establecida debido a la continuidad de Milgram (Ver Fig. 3.15). Definió una continuidad de lo real en un entorno virtual, en el cual la RA es parte del área general llamado realidad mixta [KA17].

Figura 3.15: Continuidad Realidad-Virtualidad

Durante la última década, la RA ha cambiado las entradas de aplicaciones comunes para dispositivos móviles, reemplazándolas por nuevas: registro y toma de imágenes, localización virtual o física, datos de la brújula y del acelerómetro o pantallas táctiles. Constantemente, la forma de trabajar de los desarrolladores ha ido cambiando, porque para desarrollar la aplicación de RA hay que tener en cuenta todos estos cambios que afectan directamente a la experiencia del usuario [AH11].

La manera en que los usuarios interactúan con los sistemas de RA ha cambiado radicalmente en comparación con las aplicaciones tradicionales, porque la forma de interactuar es directamente con el mundo a través de la aplicación.

De cara al futuro, hay una gran cantidad de inversores tecnológicos que han arriesgado miles de millones, lo que ya esta cambiando la forma en que compramos, diseñamos, percibimos y pensamos. La RA se estima que ayudará en todos los campos de la industria de marketing, tecnología, arte, diseño y medicina entre otros, esto argumenta las predicciones que indican que los ingresos y los dispositivos haciendo uso de la RA van a crecer enormemente durante los próximos años (ver Figura 3.16).

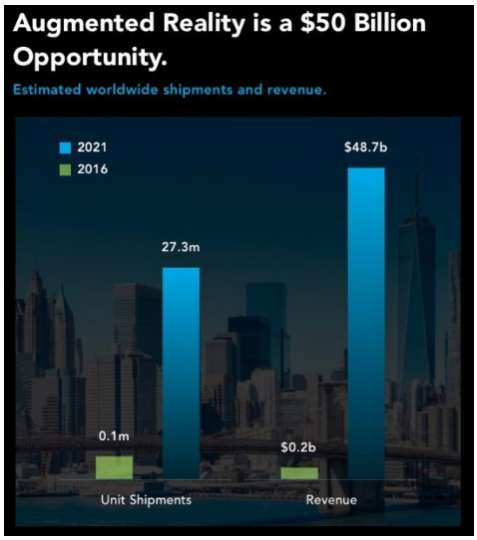


Figura 3.16: Estimación de dispositivos de RA comprados e ingresos (en dólares) para el año 2021.

3.5.1 Criterios

En la actualidad, hay un gran número de SDKs disponibles para desarrolladores. Ha sido tan grande la evolución que los SDKs actuales han alcanzado que los desarrolladores puede utilizar librerías y centrarse en la lógica de la aplicación que quieren crear y su contenido.

Los SDKs que se van a nombrar a continuación cumplen los siguientes criterios:

- Es posible de utilizar la librería mediante acceso gratuito o con una licencia comercial.
- Son librerías que se han publicado oficialmente y ya no están en desarrollo.
- Tiene una comunidad para soporte al desarrollo, como por ejemplo, un foro.
- Debe tener compatibilidad con al menos un sistema operativo móvil.

3.5.2 Listado de SDKs

Los SDKs de este listado cumplen con los criterios del punto anterior. Estos criterios son estrictamente necesarios para el desarrollo de este proyecto.

Google ARCore

Google ha lanzado ARCore en el año 2017. Su objetivo es ofrecer herramientas de realidad aumentada a los desarrolladores de Android. ARCore se basa en la tecnología Tango, lo que permite que la

RA esté disponible en los dispositivos Android sin necesitar *hardware* adicional. El SDK funciona con Unity, Unreal y Java/OpenGL [Goo18a].

ARCore es una herramienta fundamental que proporciona capacidades similares a ARKit, pero funciona en el ecosistema Android. ARCore proporciona a los desarrolladores la habilidad te crear seguimiento de movimiento, comprensión del entorno, estimación de la luz, interacción del usuario y puntos orientados(permiten colocar objetos virtuales en superficies no horizontales). Con estas capacidades se puede crear una experiencia en RA o mejorar aplicaciones existentes [Lan18].

Esta diseñado para funcionar en un gran número de dispositivos Android con versión 7.0 (Nougat) o superior. La comparación con ARKit de Apple no es buena para ARCore. La parte positiva es que actualizan constantemente su software, lo que hace que poco a poco iguale sus capacidades con la API equivalente de Apple.

Apple ARKit

En iOS 11 se ha introducido ARKit. Este nuevo *framework* permite crear fácilmente experiencias de realidad aumentada tanto para iPhone como para iPad. ARKit funciona en los procesadores A9, A10 y A11. Utiliza *Visual Inertial Odometry* (VIO) para realizar un seguimiento preciso del mundo que lo rodea. Puede detectar planos horizontales como tablas y suelos, y también puede rastrear y colocar objetos en puntos más pequeños [Wol18]. ARKit tiene como características a destacar [App18]:

- El seguimiento de movimiento es rápido y estable.
- Estimación de planos con limites básicos.
- Estimación de luz ambiente.
- Estimación de escalas.
- Localización simultanea y mapeo (SLAM)
- Tiene un alto rendimiento de *hardware* y un óptimo renderizado.

Wikitude

Es un SDK multiplataforma que trabaja con Realidad Aumentada. Es muy útil tanto para desarrolladores con pocos conocimientos como para los más avanzados. Soporta plataformas como Android, iOS, tablets, *Smart Glasses* (Epson Moverio, Vuzix M100, ODG R-7). Y los *frameworks* de desarrollo que soporta son: Native API, JavaScript API, Unity3D, Xamarin, Titanium, Cordova.

Wikitude maneja toda la complejidad de la carga de red y el envió de imágenes a un servidor en la nube. Una vez que la imagen ha sido conocida en la nube, el SDK de Wikitude realiza el *tracking* local y realiza las funciones de realidad aumentada (ver Figura 3.17).

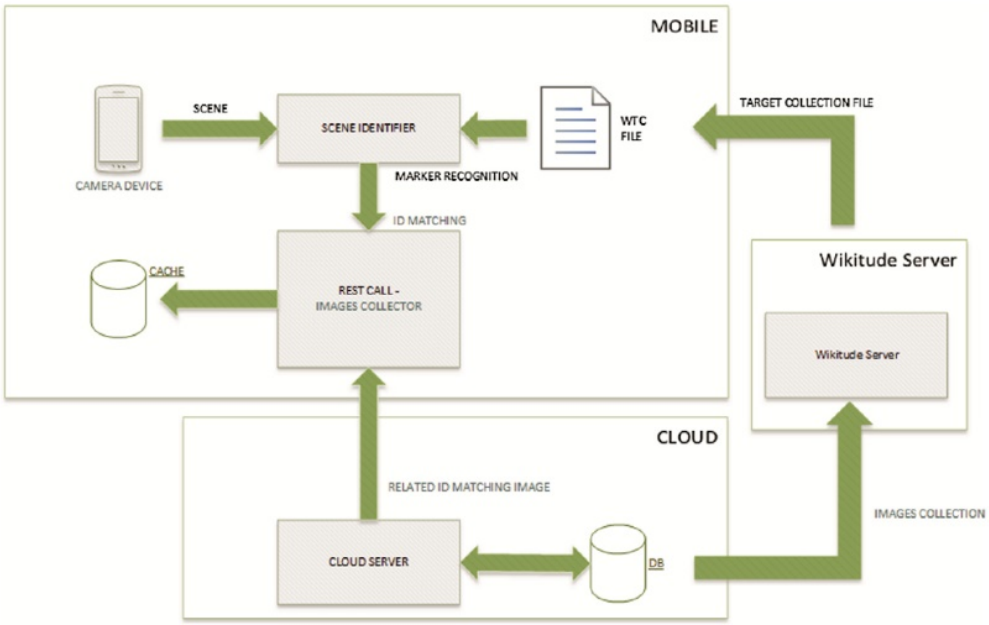
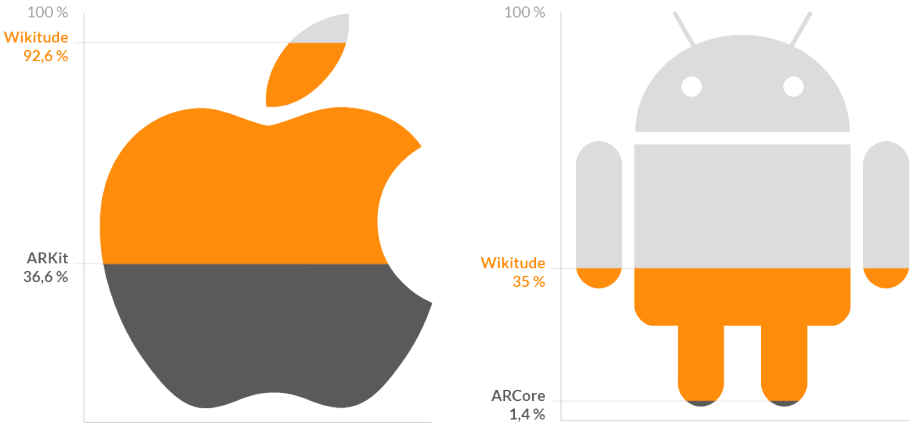


Figura 3.17: Esquema general de la arquitectura Wikitude Cloud

ARKit y ARCore han contribuido a hacer la RA más accesible a los desarrolladores pero ambos SDKs tienen todavía importantes restricciones de compatibilidad con muchos dispositivos. Wikitude, con su última versión, garantiza un mayor número de dispositivos compatibles que los SDKs de Apple (ver Figura 3.18a) y Google (ver Figura 3.18b).



(a) Wikitude vs ARKit (b) Wikitude vs ARCore

Figura 3.18: Comparativa de la compatibilidad en dispositivos de Wikitude con otros SDKs

Con la última versión Wikitude ha ampliado notablemente su listado de características [Gmb18]:

- Seguimiento 3D sin marcas.
- Seguimiento y reconocimiento de objetos.

- Seguimiento y reconocimiento de imágenes.
- Soporte para ARKit, ARCore, Geo-localización AR para apps.
- Seguimiento instantáneo.
- Localiza múltiples imágenes - reconoce varias imágenes al mismo tiempo.

ARToolKit

ARToolKit es un proyecto *software* de código abierto. Es gratuito tanto para desarrolladores como para la distribución de aplicaciones creadas. Como proyecto de código abierto, el código fuente está disponible para cualquiera que quiera usarlo, leerlo e incluso modificarlo [LB17].

La librería esta escrita en C/C++, que era la base de muchos desarrolladores de RA. Fue originalmente desarrollada por Hirokazu Kato en 1999 en el laboratorio tecnológico *Human Interface* (HITLab) de la Universidad de Washington bajo la licencia GNU. El proyecto ARToolKit, a día de hoy, es mantenido por HITLab y HIT Lab NZ en la Universidad de Canterbury en Nueva Zelanda. Los propietarios, están creando nuevas versiones como ARToolworks que adapta la versión original de ARToolKit para ser útil en otras plataformas y ofrecer numerosas soluciones a nivel profesional. Por ejemplo, ARToolKit esta disponible para los sistemas operativos de dispositivos móviles Apple bajo la licencia de ARToolworks [Mul11].

Las características que destacan en ARToolkit son:

- Cámara única o cámara estéreo (Seguimiento de la posición/orientación de la cámara).
- *Tracking* de cuadrados negros como patrón de reconocimiento.
- *Tracking* de imágenes planas.
- Calibración de la cámara, generación de patrones para marcas y utilidades para la generación de marcas de características naturales.
- Tiene *plugins* para Unity y OpenSceneGraph.
- Soporte óptico para gafas 3D.
- Es lo suficientemente rápido para aplicaciones de RA en tiempo real.

Los principales acontecimientos que destacan en la evolución de ARToolKit desde su aparición son:

- **1999** Primera demostración pública.
- **2001** versión 1, con *tracking* de marcas.
- **2001** ARToolworks Inc., realizan la versión Pro.
- **2004** *Tracking* de características naturales.
- **2008-2010** Soporte para iPhone 3g e innovación *Open Source*.
- **2011-2014** Soporte para Android y amplicación del soporte de la plataforma.
- **2015** DAQRI adquiere el proyecto.
- **2017** ARToolKit 6 - Un gran estreno en la industria de la RA.

NyARToolkit

NyARToolkit es una librería de RA basada en ARToolKit e implementada para entornos como Java o C#. Tiene compatibilidad con dispositivos Android e iOS. Es una versión simplificada de ARToolKit, tiene las funcionalidades de identificar y rastrear una imagen. Incorpora una característica que permite usar el dispositivo móvil como cliente de una aplicación RA ejecutada en un servidor remoto [Nsg16].

EasyAR

Es una librería gratuita y fácil de usar, similar a Vuforia. Su API la podemos encontrar en múltiples lenguajes como C, C++11, tradicional C++, Java para Android, Swift para iOS, Objetive-C para iOS y Unity3D. Las plataformas soportadas son Android, iOS, UWP, Windows, Mac y el editor de Unity.

La librería es completamente gratuita. Para empezar a trabajar con EasyAR solo necesitas crearte una cuenta y generar una clave. Es una librería fácil de integrar. La documentación y los ejemplos son intuitivos [Eas18]

Mientras que la versión 1.3.1 soportaba únicamente reconocimiento de imágenes, la nueva versión 2.0 tiene las siguientes características:

- Reconocimiento de objetos 3D.
- Percepción del entorno.
- Reconocimiento en la nube.
- Solución para *smart glasses*.
- Paquete de aplicaciones en la nube.

Kudan

Esta librería se caracteriza por la optimización que tiene. Kudan ayuda a las aplicaciones de dispositivos móviles a mapear modelos multipoligonales e importarlos a objetos 3D mediante un software de modelización que trae. Merece destacar que el número de *targets* a reconocer no tiene límite y necesita muy poca memoria para almacenar archivos en el dispositivo. Las plataformas soportadas son Android, iOS, Unity Editor y *Smart Glasses.*

La documentación es bastante corta y necesita que añadan información que ayude a los desarrolladores. La librería hace uso de OpenGL pero tiene unas funcionalidades muy limitadas [Kud17].

Las características de Kudan son:

- Reconocimiento de imágenes.
- Rastreo sin *targets* (*markless*, es decir, toma como referencia las características natural como son esquinas, bordes o ciertas texturas).
- Mapeo de elementos adicionales.

Maxst

Es un SDK mucho menos popular, pero no por ello con menos calidad. Es compatible para las plataformas: Android, iOS, Windows y Mac OS. Admite la herramienta de calibración para varias *Smart Glasses* como Epson MOVERIO, ODG. Te permite aumentar modelos 3D estáticos, animaciones 3D y vídeos en las imágenes. El efecto de oclusión te permite diseñar experiencias de RA más inmersivas y realistas. Si quieres implementar funciones de RA en aplicaciones usando la cámara, se puede seguir utilizando la interfaz de su cámara sin ninguna modificación de código adicional.

En cuanto al rendimiento, el algoritmo esta optimizado para usarse en el dispositivo móvil, lo que asegura un buen rendimiento con el reconocimiento y rastreo.

Las funcionalidades que definen Maxst son [Max18]:

- *Image Tracker*: el motor de rastreo de imágenes planas permite a su aplicación rastrear objetivos de manera robusta incluso en condiciones difíciles, como el movimiento rápido de la cámara o la cobertura parcial.
- *Marker Tracker*: proporcionan más de 8000 imágenes. Es una manera fácil y eficiente de gestionar una gran cantidad de objetos aumentados.
- *Instant Tracker*: este buscador de planos descubre al instante un plano horizontal en la escena presentada por la cámara. Te permite rastrear y colocar objetos 3D con respecto al plano encontrado.
- *Object Tracker*: te permite crear efectos de AR inmersivos y realistas utilizando la estructura 3D de los objetos que quieras.
- QR/*Barcode Scanner*: es rápido en el reconocimiento, bajo en la recuperación de errores y preciso en el análisis.

Vuforia

Vuforia es la plataforma más utilizada en el mundo para el desarrollo de RA, con soporte para teléfonos actuales, tabletas y *smart glasses*. Es capaz de reconocer una imagen 2D así como diferentes tipos de objetos visuales(una caja, un cilindro, un plano), texto y entornos [CH13].

Es compatible con las siguientes plataformas: Android, iOS, UWP y Unity Editor y Smart Glasses.

Las características a destacar son [Vuf18]:

- Compatibilidad realidad mixta y con *Eyewear*, incluido Microsoft HoloLens.
- Su servicio de reconocimiento en la nube le permite usar bases de datos locales o en la nube para procesar el reconocimiento de imágenes.
- Reconoce y rastrea un conjunto amplio de objetos o imágenes.
- Localiza superficies para colocar contenido en superficies horizontales en su entorno.
- VuMark: para identificar y aumentar objetos específicos como parte de una serie, como juguetes y productos de consumo.

- Usando *Vuforia Object Scanner* puedes escanear objetos 3D físicos, producir un archivo Object Data (*.OD) y utilizarlos como *Object Targets* en el gestor de *Targets.*

ViroAR

Es una plataforma que permite crear rápidamente aplicaciones con ARKit y ARCore. Es ideal para el desarrollo de juegos. Viro permite al desarrollador la rápida creación de aplicaciones mediante herramientas que facilitan el trabajo. La principal ventaja, es que no necesitas trabajar directamente con OpenGL, lo que reduce enormemente la complejidad.

Los desarrolladores pueden trabajar con ViroAR en dos plataformas:

- **ViroReact:** permite una veloz creación de aplicaciones de RA y RV usando React Native, esto facilita el trabajo a los desarrolladores web que quieren empezar. Una vez escrito el código ya no es necesario hacer ningún cambio, es soportado en múltiples dispositivos [Vir18b].
- **ViroCore:** es un SceneKit especifico para desarrolladores Android. Combina un motor de renderizado de alto rendimiento con una API para crear aplicaciones de RA y RV usando Java [Vir18a].

Las características de ViroAR que hacen destacar a la librería son:

- **Preciso *tracking* 3D del mundo real:** puede posicionar un usuario con precisión y permite el establecimiento de objetos en el mundo.
- **Detección de planos y superficies:** encuentra superficies planas y horizontales que permite al usuario poner objetos virtuales.
- **Reconocimiento de imágenes y marcas:** detecta y reconoce imágenes y marcas en el mundo del usuario y determina la posición.
- **Potente renderizador:** soporta PBR (que permite mayor realismo en los objetos que tengamos en las escenas), iluminación basada en imágenes, HDR, físicas, partículas y otras funciones que mejoran el realismo de la escena.
- **Efectos de realismo:** añade físicas y animaciones a los objetos. Puedes simular humo, niebla, fuego y otros efectos en las escenas.

3.6 Entorno de Desarrollo Integrado

También conocidos como IDE (del inglés *Integrated Development Environment*), estan diseñados para maximizar la productividad del programador proporcionando componentes que son accesibles mediante una interfaz de usuario. Los IDE son un único programa en el cual un desarrollador lleva a cabo todos los procesos necesarios. Actualmente, estos programas proporcionan numerosas características para la creación, modificación, compilación, implementación y depuración de *software*.

Eclipse IDE for Java EE Developers

Figura 3.19: Eclipse.

Es un famoso entorno de desarrollo integrado (IDE, del inglés *Integrated Development Environment*) para Java, aunque también hay versiones para C/C++ y PHP. Está compuesto por un conjunto de herramientas de programación *open source* [Fou18b].

El IDE de Eclipse permite usar *plugins*, que son extensiones para ampliar y mejorar la funcionalidad de la plataforma. Los *plugins* que nos interesan para el proyecto son Git y Maven.

Android Studio

Figura 3.20: Eclipse.

Es el entorno de desarrollo integrado oficial para la plataforma Android. En el año 2013 fue publicado por Google y reemplazo a Eclipse, que era el IDE oficial para desarrollar aplicaciones móviles en Android.

Este IDE permite tener un dispositivo virtual Android para ejecutar y testear aplicaciones. Además tiene soporte para Gradle, que es similar a Maven. Trae gran cantidad de plantillas para crear diseños comunes de Android.

3.7 Herramientas de Gestión del proyectos

Trello

Figura 3.21: Trello.

Es una aplicación Web gratuita que permite la administración de proyecto mediante una interfaz que es un tablero con tarjetas. Permite agregar listas, adjuntar archivos, etiquetar eventos, agregar comentarios y compartir el tablero [Tre18].

Esta herramienta colaborativa se ha utilizado para este TFG, cada Sprint tiene un tablero correspondiente. El tablero se divide en base a los diferentes estados por los que pasará una historia de usuario o tarea del proyecto, siguiendo la siguiente estructura: Current sprint, To Do, In Progress, Ready to Verify, Done. Aparte, hay otro listado especifico para el *Product Backlog*, que contiene la fase inicial del proyecto con todas las historias de usuario.

La Figura 3.22 es un ejemplo del uso de Trello para la gestión de proyecto Scrum.

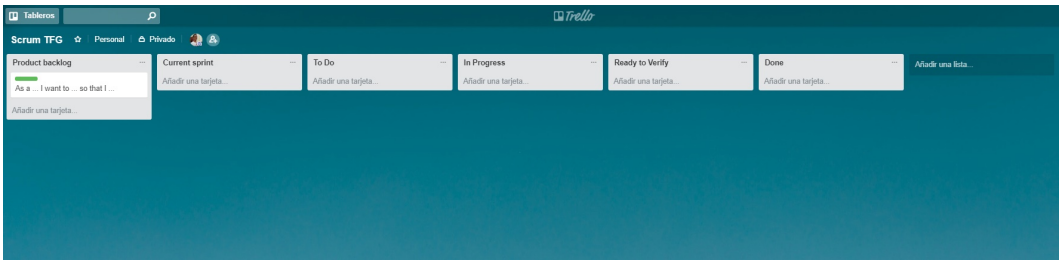


Figura 3.22: Ejemplo de un tablero en Trello.

Bitbucket

Es un repositorio web privado que permite tener un sistema de control Mercurial y Git sobre los proyectos. Trae una opción llamada *Boards*, cuya funcionalidad es casi idéntica a Trello, permitiendo crear tableros con listas para la gestión del proyecto [Bit18].

Permite usar el repositorio desde el propio Eclipse mediante un plugin.

Figura 3.23: Bitbucket.

3.8 Elaboración de documentos

TeXstudio

Es un IDE de LaTeX *open source* y multiplataforma. Antes se llamaba TexMakerX, TeXstudio empezó siendo un proyecto derivado de Texmaker cuya intención era ampliar las características sin cambiar la apariencia. Se utiliza en Windows, Unix/Linux, BSD y sistemas Mac OS X [vdZ18].

TeXstudio tiene muchas funcionalidades, las más destacadas son:

- El editor tiene autocompletado de comandos, además sugiere posibilidades a medida que escribes.
- La sintaxis está coloreada, por lo que en grandes documentos facilita y agiliza su uso.
- Permite incluir gráficos arrastrando y soltando la imagen en la posición que queremos.
- Tiene un asistente para dar formato a tablas.
- Tiene corrector ortográfico.
- Permite gestionar la bibliografía con BibTeX y BibLaTeX

Figura 3.24: TeXstudio.

Lucidchart

Lucidchart es una herramienta para realizar diagramas. Permite realizar diagramas para ingenierías de todo tipo, además trae un gran número de plantillas que permiten ayudar al usuario. Es una herramienta colaborativa, que facilita el trabajo en equipo y en tiempo real.

Figura 3.25: Lucidchart.

Su principal ventaja es que tiene una interfaz muy intuible y completa (ver Figura 3.26).

Tiene una versión de pago y una versión gratuita. A pesar de las limitaciones de la versión gratuita, se han realizado algunos esquemas y diagramas para este proyecto sin problema [Inc18].

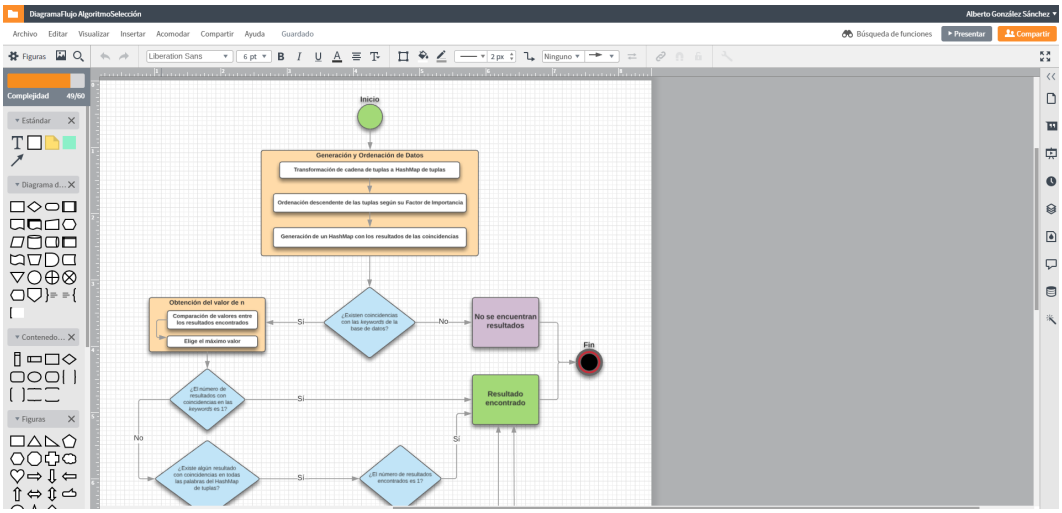


Figura 3.26: Ejemplo de un diagrama en Lucidchart

3.9 Metodología ágil - Scrum

La metodología ágil para la gestión de proyectos es Scrum, que lidera el sector de *frameworks* ágiles de desarrollo de productos. Utiliza una forma de trabajo colaborativa y organizada. Scrum fue creado básicamente para la industria del desarrollo *software* pero no está limitado solamente al sector tecnológico, podemos utilizar fácilmente este método en diferentes industrias.

En pocas palabras, Scrum es una manera de trabajar evolucionada y mejorada. Los equipos de trabajo de Scrum pueden ser pequeños o grandes. Generalmente, consiste en equipos de 5 a 9 miembros. El desarrollo se produce en pequeños pasos, después de un periodo de tiempo, cuando estos pasos están completados y combinados se obtiene el entregable final (Producto). Por lo tanto, podemos construir productos complejos usando Scrum porque te permite centrarte en pequeños entregables cada vez, lo que facilita y mejora la forma de trabajar [Ked15].

3.9.1 Beneficios de utilizar Scrum

Con esta metodología ágil el cliente se entusiasma y participa, a su manera con el proyecto dado que va viendo el progreso de cada iteración. También permite en cualquier momento que pueda modificar los requisitos para que concuerden los objetivos de negocio de su empresa, ya que al principio de cada

iteración puede introducir modificaciones en cambios funcionales o de prioridad. Además aporta una gran cantidad de beneficios:

- Cumplimiento de expectativas. El cliente determina las expectativas y el valor que tiene cada requisito del proyecto, el equipo hace una estimación y el *Product Owner*, basándose en estos datos, define su prioridad.
- Flexibilidad a cambios. Hay una buena capacidad de reacción a los cambios de requisitos generados por petición del cliente o evoluciones del mercado.
- Reducción del *Time To Market*, como al final de cada sprint se genera un software funcional, el cliente puede utilizar y probar las funcionalidades más importantes antes de que el proyecto esté terminado por completo.
- Mayor calidad del *software*. La forma de trabajar y la generación de una versión funcional después de cada sprint, permite tener como resultado un *software* de calidad superior.
- Mayor productividad, debido a que el equipo se organiza por si mismo y que no existe burocracia.
- Predicciones de tiempos. Es fácil estimar el tiempo que consumirá una determinada funcionalidad que todavía está en el Backlog.
- Reducción de riesgos. Las funcionalidades de mayor valor se realizan al principio y la velocidad con la que el equipo progresa en el proyecto se conoce, lo que permite descartar riesgos de manera anticipada.

3.9.2 Manifiesto Ágil

En 2001, diecisiete críticos de los modelos de mejora del desarrollo *software* basado en procesos se reunieron, acuñando el término «Métodos Ágiles» para definir las alternativas que están surgiendo a las metodologías tradicionales, que eran «pesadas» y demasiado rígidas (ver Figura 3.27). Durante esta reunión se definieron unos principios o valores de los métodos alternativos en cuatro postulados, denominados Manifiesto Ágil.

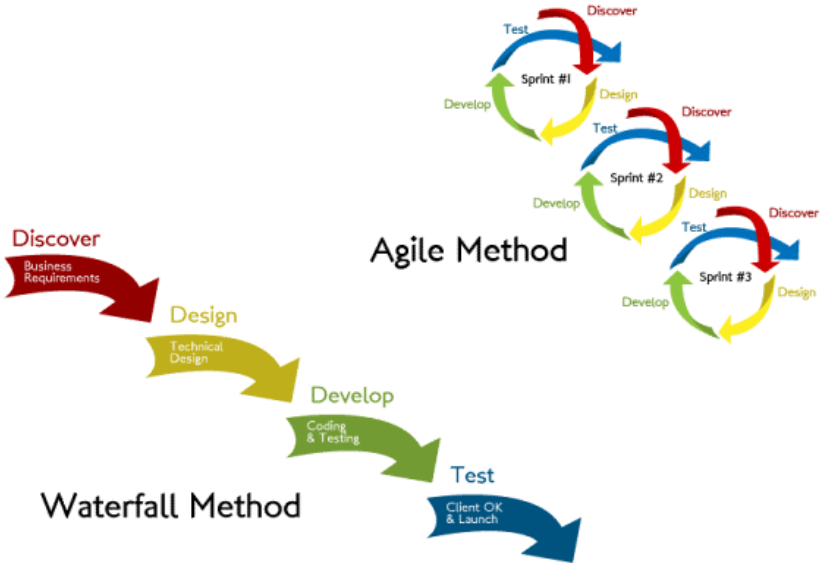


Figura 3.27: Comparación de un Método tradicional en cascada vs Método Ágil.[4]

Scrum es un *framework* ágil, por lo tanto, comprendemos e implementamos mejor scrum si entendemos los principios que derivan de los cuatro valores definidos en el manifiesto ágil [Cun01]:

- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El software funcionando es la medida principal de progreso.
- Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

[4] `https://www.business-software.com/blog/waterfall-vs-agile-development-differ-matters/`

- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

3.9.3 Roles y responsabilidades

Esta metodología tiene como objetivo principal trabajar de forma colaborativa, por lo tanto, se necesitan roles organizados previamente para tener una buena comunicación y colaboración. La suma total de los roles es conocido como Equipo Scrum. Un equipo Scrum, normalmente, esta formado por grupos de trabajo de entre 3 a 9 miembros del equipo de desarrollo, más el Scrum Master y el Product Owner (ver Figura 3.28). Cada uno de estos roles tiene asignadas unas responsabilidades concretas, por lo que cada miembro debe justificar su acciones entre ellos y para el resto de la organización [Max15].

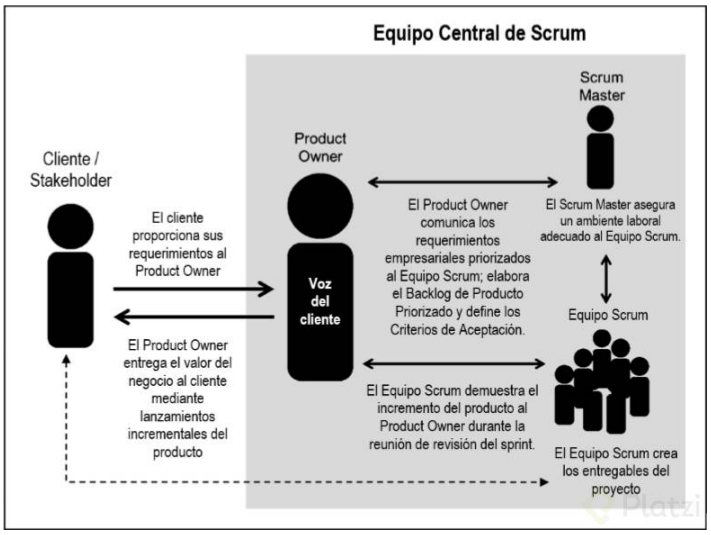


Figura 3.28: Descripción general y cómo se relacionan los roles centrales del Equipo Scrum.[5]

Product Owner

Es el responsable de conseguir un valor del producto óptimo y maximizado, además de ser el encargado de la gestión del flujo de valor del producto a través del Product Backlog. Su función como intermediario entre los *stakeholders* y sponsors del proyecto es fundamental. También es encargado de transmitir las peticiones y requerimientos de los clientes. En el caso de que el Product Owner realice también el papel de representante de negocio, añadiría valor al producto [Blo14].

La labor del Product Owner no es únicamente la de mantener el Product Backlog bien estructurado, detallado y con la prioridades definidas, sino que, además, debe comprender a la perfección cuál es el objetivo que se pretende conseguir en el producto en todo momento, siendo capaz de explicar y compartir a los *stakeholders* información que les permita entender cuál es el valor del producto.

[5] https://platzi.com/blog/que-es-scrum-y-los-roles-en-scrum/

Cada Sprint que pasa, el Product Owner hace una inversión en desarrollo que tiene que producir valor. Por lo tanto, debe señalar el Sprint Goal de forma que quede claro y consensuado con el equipo de desarrollo, esto hace que se vaya aumentando el valor del producto.

Es fundamental que el Product Owner tenga poder suficiente como para poder tomar decisiones que afecten al producto. El Product Owner debe ser el que transmita lo que el cliente quiere, sus demandas y *feedback* de ellos mismos.

Scrum Master

El Scrum Master es el encargado de cumplir dos funciones principales dentro del marco de trabajo:

1. **Gestionar el Proceso Scrum:** se encarga de gestionar y asegurar que el proceso Scrum se completa con éxito, además de hacer que sea más fácil la ejecución del proceso y sus mecánicas. Proporcina *coaching*, *mentoring* y formación a la organización, en los casos que sea necesarios, y de velar porque Scrum favorezca la entrega de valor en lugar de ser una herramienta de microgestión.
2. **Ayudar a eliminar impedimentos:** esta segunda función, indica la necesidad del Scrum Master de ayudar a eliminar progresiva y constantemente impedimentos que van apareciendo en la organización y que perjudican a la integridad de Scrum y pueden afectar a su capacidad para entregar valor. El Scrum Master es el principal responsable que debe velar para que Scrum se complete éxito, permitiendo que se pueda facilitar la implementación.

Equipo de desarrollo

El *Development Team* o Equipo de Desarrollo suele estar formado en grupos de 3 a 9 personas. Son profesionales encargados de llevar a cabo el desarrollo del producto, auto-organizarse y autogestionarse para conseguir entregar un incremento de *software* al terminar un ciclo de desarrollo.

Este equipo será el encargado de crear un incremento terminado, partiendo de los elementos del Product Backlog seleccionados o Sprint Backlog durante la fase de planificación o Sprint Planning.

En esta metodología ágil, cada miembro del equipo de desarrollo debe conocer cual es su rol, que es el mismo para todos, por lo que deberán rendir cuentas como uno solo. La responsabilidad interna decide como gestionarla el propio equipo, por lo que hay que evitar intervenir en sus dinámicas de trabajo [Mor15].

3.9.4 Artefactos de Scrum

En el *framework* de Scrum, podemos diferenciar tres artefactos. En este contexto, artefacto hace referencia a elementos físicos que se generan como resultado de la ejecución de Scrum. Los artefactos de Scrum son: Product Backlog, Sprint Backlog e Incremento.

Product Backlog

El Product Backlog es una lista priorizada y con un orden, que está ordenada por valor y riesgo comercial. Contiene el trabajo necesario para completar el proyecto. El Product Backlog muchas veces tiene historias de usuario, pero también puede contener requisitos funcionales, requisitos no funcionales, fallos y otro temas. Puede ser una lista en cualquier formato que contenga todos los requisitos necesarios para la implementación en el producto. El propietario y gestor es el Product Owner [Mor15].

El plan de publicación se puede obtener a través de algunos cálculos en el Product Backlog. En cada Sprint, el equipo tendrá un número de historias que puede generar. La velocidad depende de la cantidad de esfuerzo acumulado del producto que un equipo puede manejar en un Sprint. Esto se puede estimar viendo los Sprints previos, suponiendo que la composición del equipo y la duración del Sprint se mantienen constantes. También se puede establecer en base de sprint-a-sprint, usando planificación basada en compromiso. Una vez establecida, esta velocidad se puede usar para planificar proyectos y pronosticar fechas de lanzamiento y finalización del producto.

El Product Owner puede estimar qué historias se completarán y cuándo, basándose en el tamaño de la historia y ajustar las historias en Sprints, según la velocidad que el equipo pueda mantener. El Product Backlog debe seguir el modelo INVEST de Bill Wake donde las historias deben ser [Sad13]:

- I - *Indepedent*: cada historia de usuario debe ser completamente independiente con otras historias de usuario.
- N - *Negotiable*: una buena historia debe ser negociable en torno a la prioridad, valor, y requisitos específicos necesarios para completarla.
- V - *Valuable*: cada historia de usuario debe tener un valor explícito en el negocio.
- E - *Estimable*: No necesitamos una estimación exacta, pero sí lo suficiente para ayudar al cliente a clasificar y programar la implementación de la historia.
- S - *Small*: las historias deben ser lo más pequeñas posible para que el equipo pueda completar las historias de usuario rápidamente.
- T - *Testable*: para cada historia de usuario debe ser posible llevar a cabo todos los requisitos y procedimientos de pruebas.

Las **historias de usuario** son una forma de expresar elementos de un Product Backlog, en ellas se describe una funcionalidad que el sistema *software* debe tener, y cuya implementación es importante para el cliente [Mor15].

Sprint Backlog

Denominamos Sprint Backlog como la lista de tareas que el Scrum Team necesita completar durante el Sprint para convertir un conjunto de items del Product Backlog en un incremento de software terminado. A diferencia de los items del Product Backlog, el Sprint Backlog las tareas tienen una estimación de horas establecida. Los responsables de mantener el Sprint Backlog actualizado son los miembros del Scrum team, ya que son ellos los que están realizando el trabajo.

Durante un Sprint, se considera un comportamiento normal que se puedan descubrir nuevas tareas y que tareas ya definidas puedan necesitar modificaciones. El Scrum Team simplemente añade las nuevas tareas (junto con su estimación de trabajo correspondiente) al Sprint Backlog o reajusta la tareas en progreso (y si es necesario el trabajo estimado restante). Cuando el Scrum Team completa las tareas, deben marcarse en el sprint backlog. El Sprint Backlog muestra a los miembros del Scrum Team lo que esta completo y lo que queda pendiente. Estos datos ayudarán, a los miembros del equipo, a llevar a cabo una reunión diaria eficiente (*daily scrum meeting*) [Blo14].

Un Sprint Backlog facilita la visualización, durante cada Sprint, de las tareas que todavía no se están desarrollando, las que sí y quién está encargado de trabajar en ellas, las que están esperando desplegarse o las que están completadas.

Por lo tanto, este artefacto permite comprender como está evolucionando el trabajo durante el Sprint y la realización de un análisis de riesgos. Puesto que los objetivos de cada Sprint son diferentes y cada elemento del Product Backlog tiene un valor, el Sprint Backlog permite el análisis hasta donde se ha cumplido el objetivo y que se podría eliminar. De esta manera, obtendríamos mejores resultado de la inversión en desarrollo.

Incremento

Un Incremento es el resultado del Sprint, es decir, es el conjunto de todas las tareas realizadas durante el Sprint. Este artefacto es la pieza, en forma de *software*, que se pone en manos del usuario final, de manera que aporta un valor de negocio al producto en el que se está trabajando [RT13].

Construir *software* de manera ágil se basa en hacerlo de manera iterativa e incremental. Con el uso de iteraciones, podemos asegurarnos de que todo el ciclo ocurre en cuatro semanas o menos.

Otros artefactos

Los tres artefactos expuestos son los más importantes e imprescindibles, sin embargo, existen otros, que aunque no sean tan importantes, son útiles para asegurar la calidad de esta metodología.

- **Definition of Done (DoD):** Este documento define lo que se considera hecho en un Scrum Team.
- **Definition of Ready (DoR):** Sirve para documentar cuándo un requisito (historia de usuario u otro) está listo para que el equipo de desarrolladores pueda empezar a trabajar con él e incluirlo en un Sprint Planning.
- **Burndown Chart:** es una herramienta de medición visual que muestra el trabajo completado por día en comparación con la tasa de finalización proyectada para el lanzamiento del proyecto actual. Su propósito es permitir que el proyecto esté encaminado para entregar la solución esperada dentro del calendario deseado.

3.9.5 Qué es un Sprint

Sprint es un contenedor para el resto de los eventos de Scrum. Es continuo, es decir, su duración no varía y se puede definir como una medida de ritmo constante a lo largo del tiempo, lo que nos permite reducir complejidad y poder comprar los resultados obtenidos a lo largo de diferentes Sprints. El

Sprint es sinónimo de transparencia, permitiendo inspeccionar y adaptar todos los otros eventos de Scrum.

Scrum determina que la duración adecuada de un Sprint debe ser de aproximadamente 4 semanas. Aunque, habitualmente, los Scrum Team deciden la duración de los Sprints dependiendo de los objetivos que deseen cumplir. Según los objetivos y el tipo de organización la duración real de los Sprints es la siguiente:

- **2 semanas:** es la menos frecuente, se suele utilizar en proyectos que necesiten generar resultados a corto plazo. Por ejemplo, I+D o PoCs.
- **3 semanas:** es algo más común, es parecida a la anterior, pero con proyectos de mayor envergadura.
- **4 semanas:** es la más utilizada, ya que tiene un tamaño de ciclos que se adapta bien a la mayoría de los productos.
- **5 semanas:** es también muy utilizada, aunque escapa de los estándares establecidos por Scrum. Es bastante útil en las fases iniciales de algunos proyectos.

El Sprint es una iteración definida (*time boxed*) que es muy útil para llevar a cabo un desarrollo iterativo e incremental. La duración de un Srint viene definida por el periodo de tiempo mínimo en el que un equipo de desarrollo es capaz de generar valor mediante un incremento determinado [Rub12].

Los eventos o ceremonias que un Sprint normal tiene son: *Sprint Planning*, *Daily Scrums*, *Sprint Review*, *Sprint Retrospective* y *Sprint Grooming* o *Refinement*. En la Figura 3.29, se puede observar el flujo de eventos normal de un Sprint [Roc17].

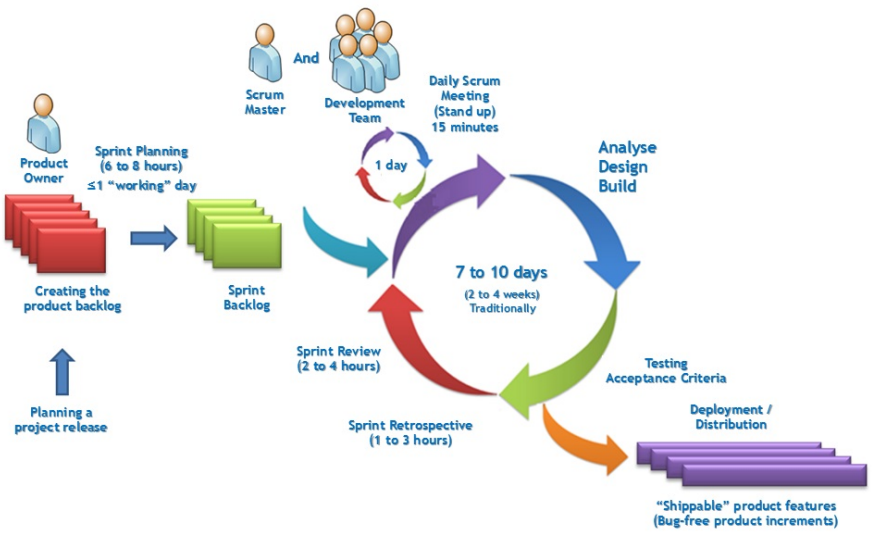


Figura 3.29: Flujo de eventos de un Sprint de Scrum.

Sprint Planning

Es una reunión que se realiza al principio de cada Sprint, en la que debe participar el Scrum Team completo. En esta reunión se analiza el *Product Backlog* para que el equipo de desarrollo pueda elegir en que items se va a trabajar en el próximo Sprint. Estos items elegidos son los que formarán el *Sprint Backlog* [Rub12].

En esta reunión, el *product owner* es el encargado de presentar una versión actualizada del *Product Backlog* para que el equipo de desarrolladores se encargue de realizar una estimación y, si fuera necesario, aclarar los items.

Las dos partes principales de un *Sprint Planning* son, **Qué** se va a hacer en el próximo Sprint y, después, se aborda el **Cómo**. El encargado de liderar la primera parte es el *product owner* y en la segunda es el propio equipo de desarrolladores. A su vez, la única función del *Scrum Master* es comprobar que la reunión se realiza como parte de Scrum y cumple con la duración estimada.

La duración del *Sprint Planning* puede ser de hasta 8 horas para Sprints de 30 días. La razón de esta reunión, es obtener una sincronización entre negocio y el avance del producto en desarrollo en base a las prioridades.

Daily Scrum

Es una reunión diaria de planificación de 15 minutos en la que participa el equipo de desarrolladores y el *Scrum Master*. El propósito del *Daily Scrum* es para coordinar las actividades diarias del Sprint e identificar los impedimentos que evitan que el equipo de desarrolladores consiga los objetivos definidos. Todos los miembros del equipo de desarrolladores participa, por lo que todos son conscientes del trabajo de todo el equipo. Si surge un impedimento durante el *Daily Scrum*, se resuelve durante la reunión. El evento es una reunión de coordinación, no una reunión para resolver problemas ni quejas [LM18].

El *Scrum Master* se asegura de que la reunión tiene lugar y que el equipo de desarrolladores la dirige. El *product owner* debe estar al tanto de lo que ocurre y también debe aclarar las prioridades si es necesario. El resto de gente involucrada e interesada puede escuchar pero no pueden hablar.

El *Daily Scrum* es la manera en la que desarrolladores se auto-organizan y auto-gestionan. Cada día, el equipo decide lo que hará cada y quién ayudará a quién. La forma de participar de cada una de las personas del equipo de desarrolladores es respondiendo a las preguntas:

1. ¿Qué hice ayer para contribuir al *Sprint Goal*?
2. ¿Qué es lo que voy a hacer hoy para contribuir al *Sprint Goal*?
3. ¿Hay algún problema que me impida hacer la entrega?

Sprint Review

Es la reunión que ocurre al final del Sprint, en donde el incremento terminado es presentado a los *stakeholders* por el *product owner* y el equipo de desarrolladores. La reunión es organizada por el *product owner*, se analiza como está la situación actual y el *Product Backlog* recibe los cambios oportunos para adaptarse a las nuevas condiciones.

El *Sprint Review* significa que el Sprint ha terminado, por lo tanto, el equipo ha estado hasta cuatro semanas trabajando en un incremento terminado de *software* para mostrárselo a los *stakeholders*. Este evento es una reunión de trabajo, no es una demostración del *software* [MJ18].

Sprint Retrospective

La principal razón para una reunión *Sprint Retrospective* es para crear un diálogo, haciendo que el *Scrum Team* tenga la oportunidad de analizar y revisar el ciclo de vida de Sprint entero, con la intención de mejorar el proceso en general. La reunión esta basada en conceptos de mejora continuados. Al final de cada Sprint, se lleva a cabo una reunión retrospectiva para analizar y revisar todas las cosas que han ido bien, y también las que no.

En la reunión, el *Scrum Team* puede analizar procesos existentes, tecnologías, colaboraciones, técnicas de comunicación, etc. Una vez que el *Sprint Retrospective* concluye, el *Scrum Team* seleccionará cuidadosamente las posibles mejoras y el plan para ellas en el siguiente Sprint.

La duración típica de una reunión *Sprint Retrospective* es de una hora y media, pero en muchos casos puede requerir más tiempo dependiendo de la duración del Sprint, complejidad del proyecto, tamaño del equipo o la experiencia general del equipo utilizando metodologías Scrum [MM16].

Sprint Grooming o Refinement

El *Sprint Grooming* o refinamiento del *Product Backlog* es una práctica recomendada que asegura su preparación. Este evento es parecido a los otros. Su duración estimada es de dos horas por cada semana del Sprint. En este *meeting* participan el *Scrum Team* y cualquier otro recurso que el *product owner* crea necesario. Es importante que antes de llevarse a cabo la reunión todos los participantes tengan conocimiento de los requisitos o historias de usuario que se van a tratar [Sch12].

Capítulo 4

Método de trabajo

Este capítulo explica de forma detallada la arquitectura general del proyecto, el desarrollo progresivo del trabajo en base a la metodología seguida y las herramientas y tecnologías utilizadas en cada una de las partes desarrolladas en este TFG.

La metodología de gestión de proyectos escogida es Scrum, de la que se habla en detalle en el capítulo anterior. En este capítulo se explica el por qué de su elección y se expondrá los detalles de los *Sprints*.

4.1 Arquitectura del sistema

En esta sección se va a hacer un desarrollo progresivo de la estructura y arquitectura de este proyecto, desde una visión general hasta los detalles de cada módulo.

Recordamos que el objetivo principal de este proyecto es enriquecer la información textual con información virtual integrada. Debido a que la información en papel tiene limitaciones, como por ejemplo, la falta de libertad para visualizar un objeto desde diferentes ángulos, surge la idea de complementar la Realidad Aumentada con la información en papel, de tal forma que el usuario tenga libertad para examinar cualquier parte e incluso se pueda animar. En muchas circunstancias, animar un objeto o poder visualizarlo desde diferentes ángulos puede ayudar a la comprensión.

En la Figura 4.1 se reconocen cuatro partes claramente diferenciadas, Servidor, Cliente Android, Cliente Web y Comunicación.

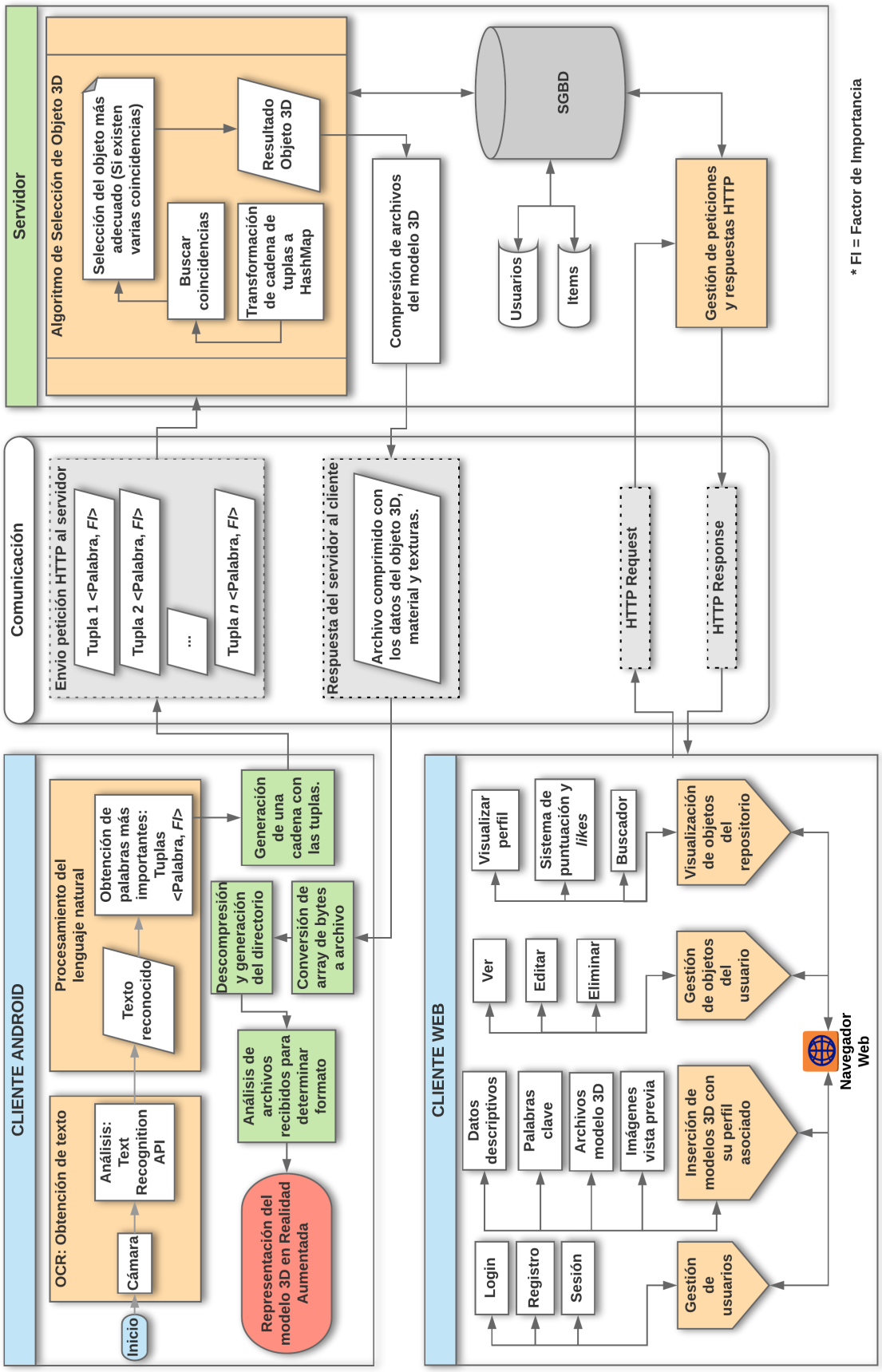


Figura 4.1: Arquitectura general detallada del sistema.

El **Cliente Android**, es la aplicación para dispositivos móviles, encargada de obtener el texto mediante reconocimiento óptico de caracteres y un procesamiento del lenguaje natural para obtener un listado de palabras clave y su importancia en el texto. Este listado es enviado al servidor con el fin de recibir un objeto 3D que sea acorde al significado del texto y así reproducirlo sobre una superficie mediante una librería integrada de Realidad Aumentada.

La parte del **Cliente Web** consiste en una aplicación web colaborativa que funciona como un repositorio para los objetos 3D y permite el registro y *login* de usuarios, esto hace posible que los usuarios tengan sus objetos 3D compartidos. Estos objetos 3D compartidos son los que utiliza el Cliente Android.

El **Servidor** es la parte del sistema encargada de gestionar las peticiones de los clientes. Para el Cliente Web de gestionar las peticiones y respuestas HTTP y para el Cliente Android de recibir el listado de palabras clave, generar una respuesta mediante el algoritmo de selección de objeto 3D y enviar los archivos al cliente.

La **comunicación** Cliente Android - Servidor es la encargada de hacer que el servidor reciba el listado de tuplas que definen las palabras clave y su factor de importancia, y que el cliente reciba los archivos de forma correcta.

La comunicación Cliente Web - Servidor se encarga de transmitir las peticiones HTTP y las respuestas HTTP.

A continuación se analizan en detalle las partes del sistema.

4.1.1 Cliente Android

La aplicación Android es junto al algoritmo de selección de objetos 3D del servidor, la parte más compleja del sistema. El desarrollo de esta aplicación se ha llevado a cabo en cuatro módulos:

- Módulo OCR (reconocimiento óptico de caracteres).
- Módulo PLN (procesamiento lenguaje natural).
- Gestor de peticiones/respuestas.
- Módulo de Realidad Aumentada.

A continuación se hará una descripción detallada de los cuatro módulos del Cliente Android.

Módulo OCR

Este módulo forma parte del Cliente Android y es el encargado de obtener la entrada inicial de la aplicación móvil (ver Figura 4.2).

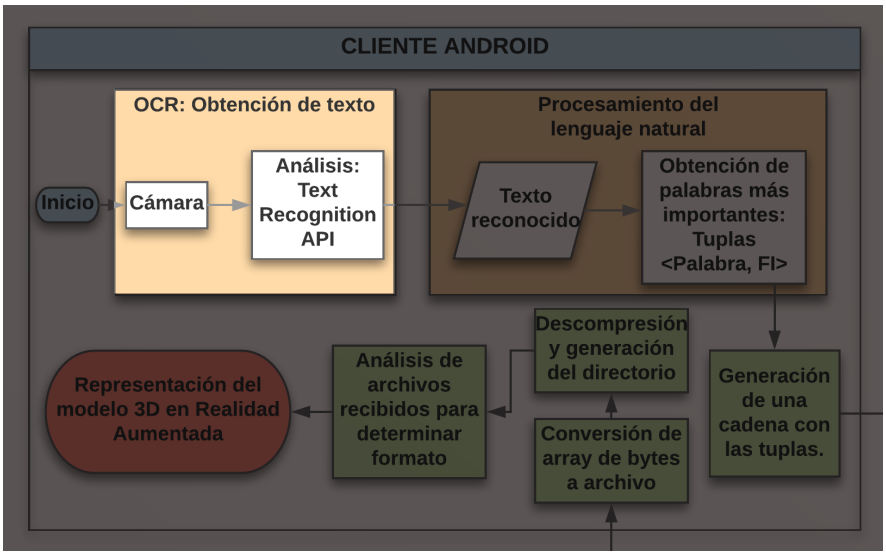


Figura 4.2: Módulo OCR del Cliente Android.

El objetivo final es obtener el texto en el dispositivo para así digitalizarlo y analizarlo. Para ello diferenciamos dos partes:

1. Cámara: Es lo primero que el usuario se encuentra cuando ejecuta la aplicación. La cámara es un paso muy importante en el proceso de analizar el texto, ya que para obtener un buen análisis es necesario haber extraído adecuadamente el texto a formato digital. El reconocimiento óptico de caracteres recibe la fotografía generada por la cámara y la calidad de este reconocimiento se ve totalmente afectada por la calidad de la imagen. La calidad de la imagen para el reconocimiento está definida por factores como la iluminación, la focalización, el ángulo usado, el tamaño del texto, el encuadre, etc. En el capítulo de resultados se analizarán estos aspectos en detalle realizando pruebas y comparando los resultados obtenidos.

La cámara tiene una interfaz *landscape* que permite al usuario seleccionar mediante un rectángulo ajustable donde quiere hacer la foto, lo que facilita la realización de una fotografía con un texto concreto y encuadrada, evitando así hacer una foto a una cantidad de texto no deseado que podría generar resultados erróneos (ver Figura 4.3). Esto, además, ayuda con la alineación y obliga al usuario a mantener el dispositivo lo suficientemente lejos para garantizar que la imagen esté enfocada.

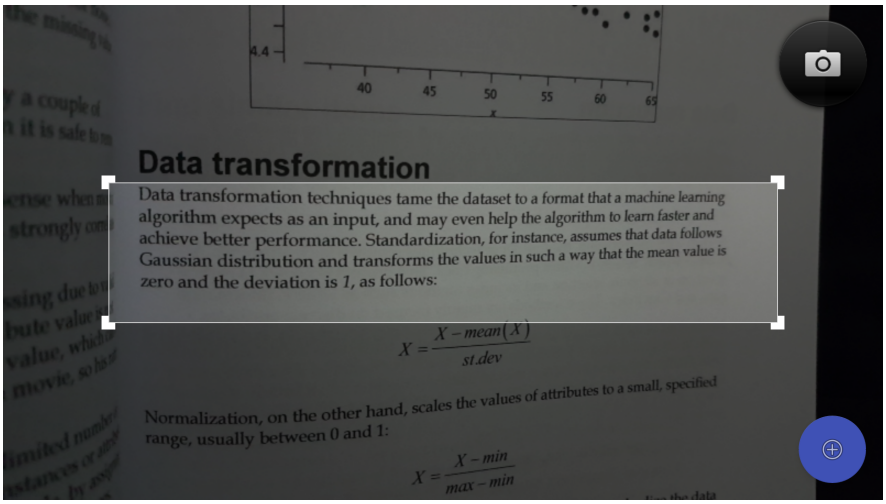


Figura 4.3: Interfaz cámara - Rectángulo ajustable de selección.

Para conseguir esta interfaz en la cámara, se superpone una vista llamada `viewfinderView` sobre la vista previa de la cámara, que es una vista donde se añaden el rectángulo ajustable para el encuadre y la transparencia parcial de la parte externa del rectángulo.

En la clase `CameraManager` se encuentra el método `adjustFramingRect` que es el método encargado de modificar el rectángulo que la interfaz de usuario debe dibujar. Este método es llamado desde la actividad mediante un `setOnTouchListener` que detecta si el usuario toca las esquinas y modifica las coordenadas del rectángulo. Con el método `getFramingRect` obtenemos un objeto de la clase `Rect` que contiene las coordenadas del rectángulo.

La librería utilizada para representar visualmente el rectángulo ajustable es `android.graphics`, esta agrega el rectángulo, la transparencia parcial fuera de él y las esquinas blancas para que sea más intuible su uso. Por ejemplo, en el fragmento de código del Listado 4.1, se establece el color y después dibuja el rectángulo basándose en las coordenadas definidas en la clase `CameraManager`.

```
1 paint.setColor(maskColor);
2 canvas.drawRect(0, 0, width, frame.top, paint);
3 canvas.drawRect(0, frame.top, frame.left, frame.bottom + 1, paint);
4 canvas.drawRect(frame.right + 1, frame.top, width, frame.bottom + 1,
      paint);
5 canvas.drawRect(0, frame.bottom + 1, width, height, paint);
```

Listado 4.1: Ejemplo de código para dibujar la parte exterior del rectángulo

Al añadir esta vista que se superpone a la vista de la cámara original se pierde también el botón original, por lo tanto, se ha añadido un botón llamado `ShutterButton` para cumplir con la funcionalidad, además de otras adicionales, como por ejemplo que al mantenerlo pulsado realiza un *autofocus* o, también, que mientras esta pulsado tiene un color diferente.

La calidad de la foto obtenida es un factor realmente importante a la hora de conseguir un buen reconocimiento de caracteres. Para conseguir una imagen de calidad la cámara realiza un *autofocus* al presionar el botón de realizar la foto o al mantenerlo pulsado hace un *autofocus* continuado.

La fotografía es almacenada como cadena de bytes en un objeto tipo *Bitmap*, lo que hace que no se almacenen las fotos realizadas, evitando así un uso inapropiado del espacio en memoria.

Como uso alternativo se ha añadido la opción de escribir el texto manualmente. Utilizando esta función el usuario accede directamente al módulo del procesamiento del lenguaje natural, evitando usar el OCR y la vista previa del resultado obtenido. Esta función, por ejemplo, podría servir en el caso de que el usuario quisiera ver un objeto 3D y el nombre no aparece en texto.

2. OCR y visualización de resultado: En esta parte del módulo la entrada es la imagen generada por la cámara, y mediante el uso de técnicas de reconocimiento óptico de caracteres se genera la salida (ver Figura 4.4).

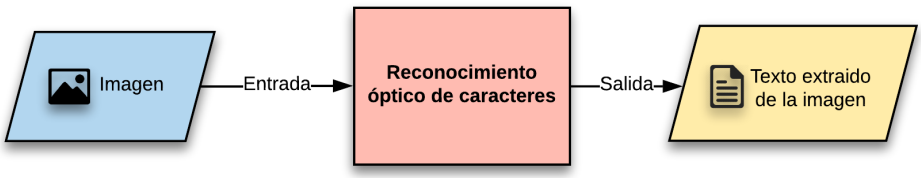


Figura 4.4: Entrada y Salida del Módulo OCR.

Para el reconocimiento óptico de caracteres se ha utilizado la librería *Vision API* de Google Cloud. Esta librería ofrece funcionalidades como:

- Detección de etiquetas.
- Detección de caras.
- Detección de contenido explícito.
- Detecta los atributos generales de la imagen y busca imágenes similares en Internet.
- Detecta logotipos de productos muy conocidos dentro de una imagen.
- Detecta estructuras artificiales y naturales famosas dentro de imágenes.
- Detecta y extrae texto de una imagen, que es la que se ha utilizado en el proyecto.

La principal ventaja de esta librería es que funciona mediante la conexión con un servidor que es el encargado de analizar la imagen. Por lo que esta librería integrada es ultraligera y sencilla de utilizar. El algoritmo de reconocimiento óptico de caracteres se basa en cuatro etapas:

1. **Binarización:** Para poder analizar la imagen debe estar en blanco y negro, por eso es importante la conversión, ya que las imágenes normalmente están en escala de grises o en color. Mediante este proceso quedan claramente definidos los contornos de los caracteres y símbolos que aparecen en la imagen.

2. **Segmentación de la imagen:** Es el proceso más costoso. Se obtienen las entidades lógicas mediante la detección de contornos o regiones de la imagen.
3. **Adelgazamiento de los componentes:** En este proceso se borran los puntos de los contornos de todos los componentes sin perder las proporciones originales.
4. **Comparación con patrones:** Se comparan los caracteres obtenidos con unos patrones almacenados en una base de datos.

Como la mayoría de APIs ofrecidas por Google, para acceder a Cloud Vision API es necesario usar la librería de Google API Client. Para usar esta librería en el proyecto de Android Studio hay que añadir la siguientes dependencias en el archivo `build.gradle`:

```
compile 'com.google.api-client:google-api-client-android:1.22.0'
compile 'com.google.apis:google-api-services-vision:v1-rev357-1.22.0'
compile 'com.google.code.findbugs:jsr305:2.0.1'
```

Listado 4.2: Dependencias de la libreria de OCR

La librería de Google sólo funciona si la aplicación tiene permisos de Internet, para ello es necesario añadir la siguiente linea en el archivo *manifest* del proyecto:

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

Listado 4.3: Permisos de Internet en el *manifest*

La forma de usar esta librería es creando un método que utiliza la imagen generada por la cámara, la envía al servidor y recibe el texto. En el Listado 1 está el fragmento de pseudocódigo simplificado con lo necesario para poder realizar el análisis de la imagen obtenida.

Proceso 1 Método llamada a Google Cloud Vision API

Entrada: Bitmap - Imagen en cadena de bytes.

Salida: Almacena la cadena con el texto obtenido.

1: Inicializar servicio TextRecognizer
2: $resultado = ""$
3: $text\_array \leftarrow analisis\_OCR$
4: Siendo i un entero inicializado a 0
5: **para** i <tamaño text_array **hacer**
6: $item \leftarrow text\_cogerValor(i)$
7: **si** $item \neq nulo$ **entonces**
8: $resultado \leftarrow resultado + valor\_item + " "$
9: **fin si**
10: **fin para**
11: Almacena y muestra el valor de resultado

Este método recibe, como parámetros, un objeto de la clase `Bitmap`. La clase `TextRecognizer` se inicializa al principio del método, es la encargada de realizar la petición y obtener el texto. Para poder hacer la petición es necesario crear un objeto del tipo `Frame` formado por el *Bitmap* de la imagen. El resultado lo forman bloques de texto almacenados en un `SparseArray`, para obtener el texto en una única cadena se recorre hasta el final y se va concatenando en cada iteración el texto obtenido 4.4.

```
1    public void detectText(Bitmap textBitmap){
2    TextRecognizer textRecognizer = new TextRecognizer.Builder(this).
     build();
3    Frame frame = new Frame.Builder().setBitmap(textBitmap).build();
4    SparseArray<TextBlock> text = textRecognizer.detect(frame);
5    String cadena="";
6    for (int i = 0; i < text.size(); ++i) {
7      TextBlock item = text.valueAt(i);
8      if (item != null && item.getValue() != null) {
9        cadena += item.getValue()+" ";
10     }
11   }
12   resultado.setText(cadena);
13 }
```

Listado 4.4: Método encargado del reconocimiento óptico de caracteres

Para mostrar el texto como resultado al usuario, la aplicación avanza cambiando de *Layout*, este está compuesto por la imagen tomada con las palabras señaladas y por el resultado, es decir, el texto correspondiente a la imagen. En la Figura 4.5 se muestra un ejemplo de un resultado, a la izquierda está la imagen tomada con las palabras detectadas correctamente señaladas y a la derecha está el texto

dentro de un `textView` que permite desplazarse en el caso de que el resultado obtenido sea extenso.

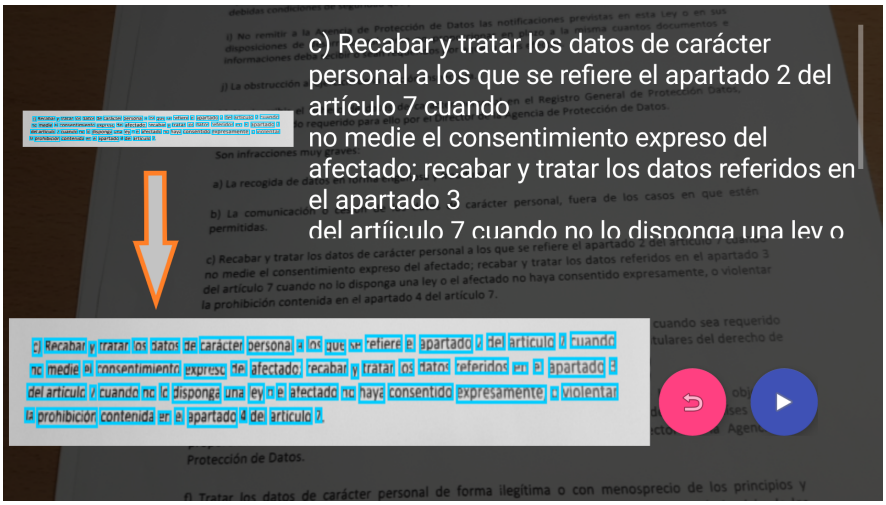


Figura 4.5: Ejemplo de un resultado obtenido mediante OCR.

El texto obtenido en formato digital es la salida final del módulo OCR. Este texto es utilizado por el módulo del Procesamiento del Lenguaje Natural para obtener un listado de las palabras clave del texto (ver Figura 4.4).

Módulo PLN (Procesamiento del Lenguaje Natural)

El módulo PLN es el intermediario entre la obtención del texto y la comunicación con el Servidor, ya que para realizar la petición al servidor es necesario ejecutar un análisis del texto (ver Figura 4.6).

Este proceso toma como entrada el texto obtenido del módulo de reconocimiento óptico de caracteres, este texto digitalizado se envía al servidor que mediante su propio algoritmo obtiene un listado de palabras clave y su factor de importancia. El **Factor de Importancia(FI)** representa un valor numérico comprendido en el intervalo unidad, es decir, es un valor que forma parte del conjunto de todos los números reales del intervalo [0,1], donde el valor mínimo es 0 y el valor máximo es 1.

La salida es el listado de tuplas formadas por las palabras clave y la relevancia en el texto analizado, es decir, su factor de importancia (ver Figura 4.7). Este listado de tuplas está almacenado en un *HashMap* que es una estructura compuesta por pares de claves y valores. Las claves son de tipo *String*, para representar las palabras clave, y los valores son *Doubles*, que permiten representar el Factor de Importancia que es un número real. El uso de esta estructura aporta cierta organización, ya que este resultado es el que se envía al servidor y el algoritmo de selección de objeto 3D necesita saber qué Factor de Importancia corresponde a cada palabra clave.

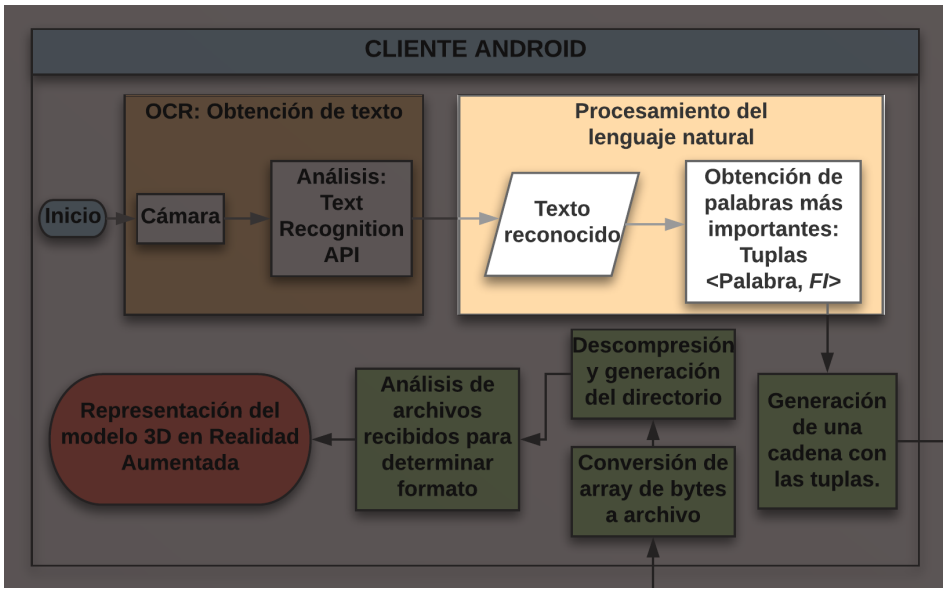


Figura 4.6: Módulo PLN del Cliente Android.

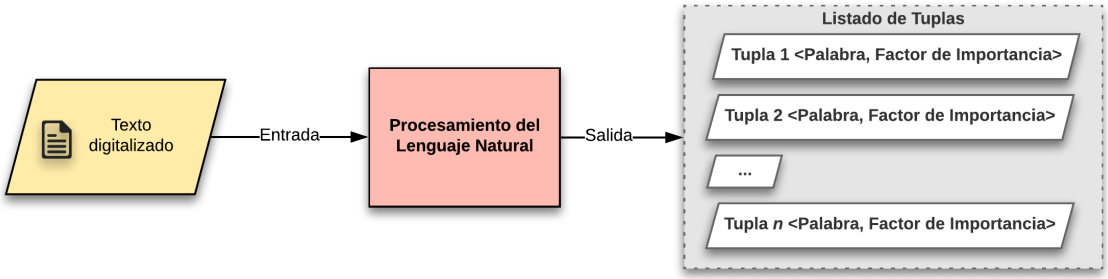


Figura 4.7: E/S del Módulo PLN en el Cliente Android.

En la ventana de análisis del texto, el texto puede ser escrito manualmente o utilizar el texto obtenido mediante el reconocimiento óptico de caracteres. Para realizar el análisis del texto y obtención del listado de palabras clave se ha utilizado la librería *Natural Language* de Google Cloud. Esta librería permite obtener información valiosa de textos mediante una API REST[1] fácil de usar, como por ejemplo, entidades, sentimientos, sintaxis, categorías, palabras clave, etc. Este algoritmo aprende representaciones de documentos a partir de características geométricas y relaciones espaciales, características de contenido multimodal, señales sintácticas, semánticas y pragmáticas.

Para establecer los parámetros necesarios para el análisis del texto y obtener lo que queremos (ver Listado 4.5), primero se debe crear un objeto de clase `Document` y establecer el tipo, en este caso es texto plano y el contenido del documento, que sería el texto que queremos analizar. A continuación, hay que definir qué características vamos a ejecutar en la llamada al servidor, esto se hace mediante un objeto de la clase `Features` y estableciendo a *true* la extracción de entidades con el método

[1]Protocolo de intercambio y manipulación de datos en los servicios de internet.

`setExtractEntities`. Por último, antes de ejecutar la llamada al servicio, se crea un objeto de la clase `AnnotateTextRequest` para crear la petición que se va a enviar al servidor, en ella se establece el documento (con el contenido y el tipo) y la característica que se requiere (extracción de entidades).

```
1   document = new Document();
2   document.setType("PLAIN_TEXT");
3   document.setContent(text);

5   features = new Features();
6   features.setExtractEntities(true);

8   final AnnotateTextRequest request = new AnnotateTextRequest();
9   request.setDocument(document);
10  request.setFeatures(features);
```

Listado 4.5: Definiendo la petición al servidor

Para realizar la petición al servicio del servidor de Google se crea una tarea asíncrona `AsyncTask`. En la tarea previa a la ejecución, únicamente realiza un `clear` al *HashMap* del resultado y a la lista con la que muestra el resultado por pantalla, de esta manera, si el usuario realizara varias ejecuciones de análisis el resultado antiguo se borraría y se mostraría sólo el actual.

Durante la ejecución del método `doInBackGround` se realiza una llamada al servicio en la linea 4 de código mostrada en el Listado 4.6, esta ejecución contiene la petición `AnnotateTextRequest` con el tipo y contenido del documento y la característica que se va a solicitar. Si la ejecución es satisfactoria, se obtiene como respuesta un objeto `AnnotateTextResponse`.

```
1 protected AnnotateTextResponse doInBackground(Object... params) {
2   AnnotateTextResponse response = null;
3   try {
4     response = naturalLanguageService.documents().annotateText(request).
      execute();
5   } catch (IOException e) {
6     e.printStackTrace();
7   }
8   return response;
9 }
```

Listado 4.6: Invocación del servicio

En el Listado 4.7 se muestra el código del método `onPostExecute`, que es el encargado de mostrar el resultado al usuario y de extraer de la respuesta las palabras clave y sus factores de importancia correspondientes. Primero, se añaden las entidades con todos sus valores a la lista que se muestra en pantalla, para que el usuario pueda ver cual es el resultado obtenido del análisis de ese texto. A continuación, para poder realizar la petición al servidor gestor de objetos 3D, se guarda en un *HashMap*

las palabras clave y su factor de importancia, de forma que sólo se tendría almacenada la información que el servidor necesita para realizar la búsqueda y selección del objeto 3D.

```
protected void onPostExecute(AnnotateTextResponse response) {
  super.onPostExecute(response);
  if (response != null) {
    entityList.addAll(response.getEntities());
    entityListAdapter.notifyDataSetChanged();

    for(int i=0; i<entityList.size(); i++){
      mapResultado.put(response.getEntities().get(i).getName().toString()
    , response.getEntities().get(i).getSalience());
    }
  }
}
```

Listado 4.7: Extracción de datos de la respuesta

En el caso de la Figura 4.8, el *HashMap* con el resultado correspondería a las palabras «Tierra» y «sistema solar», y el factor de importancia a 0,76 y 0,23, respectivamente.

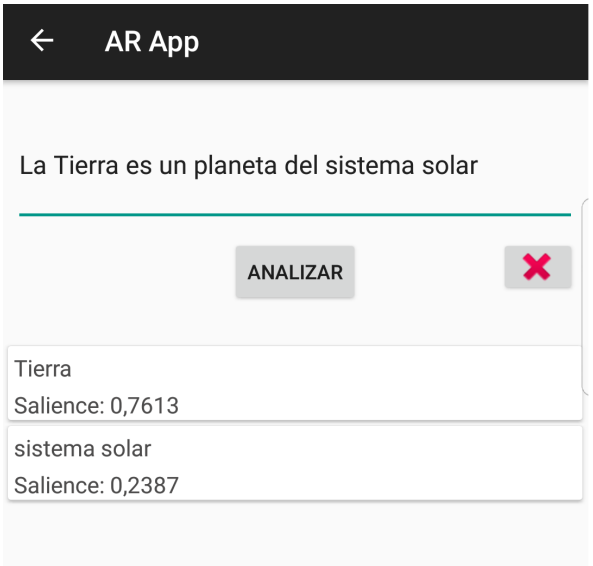


Figura 4.8: Ejemplo listado de palabras clave y su factor de importancia.

En número de palabras clave aumenta si el texto es más extenso, en el ejemplo «La Tierra es un planeta del sistema solar que gira alrededor de su estrella —el Sol— en la tercera órbita más interna.» se obtiene el resultado de la Figura 4.9.

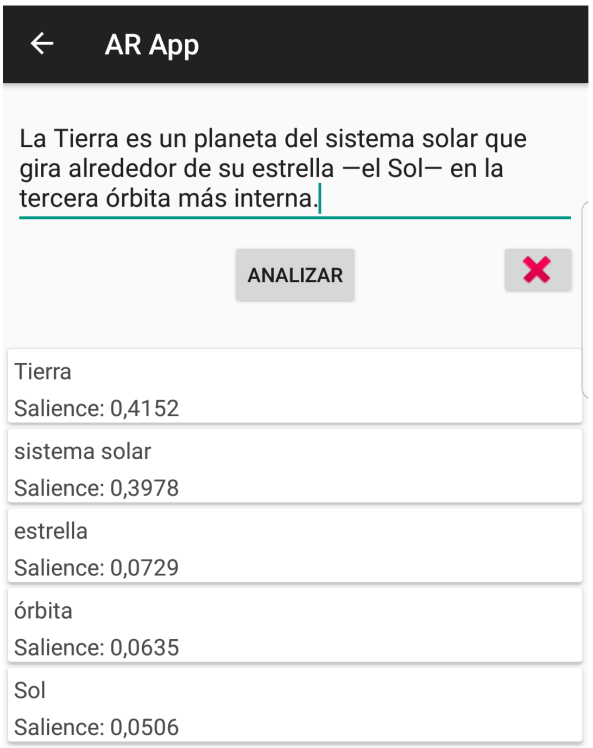


Figura 4.9: Ejemplo listado de palabras clave y su factor de importancia.

El listado de pares obtenido y almacenado en un *HashMap* es enviado al Servidor para que mediante el algoritmo de selección de objeto 3D genere una respuesta adecuada a las tuplas (ver Listado 4.8). La suma de todos los factores de importancia es 1. Un valor más alto en el *FI* significa que es más importante. Por ejemplo en este caso como el texto contiene la definición de «Planeta Tierra», la palabra detectada como más importante en el análisis es «Tierra».

```
{
  "Sol": 0.050592184,
  "orbita": 0.06347552,
  "estrella": 0.07293502,
  "Tierra": 0.4151733,
  "sistema solar": 0.397824
}
```

Listado 4.8: Ejemplo de pares obtenidos en el módulo de PLN

Gestor de peticiones y respuestas

La manera de gestionar el flujo de comunicación con el servidor está diferenciada en dos partes: las peticiones y las respuestas. Para ello es necesario preparar la información que se va a enviar o recibir, de forma que pueda adaptarse a los requerimientos de los procesos (ver Figura 4.10).

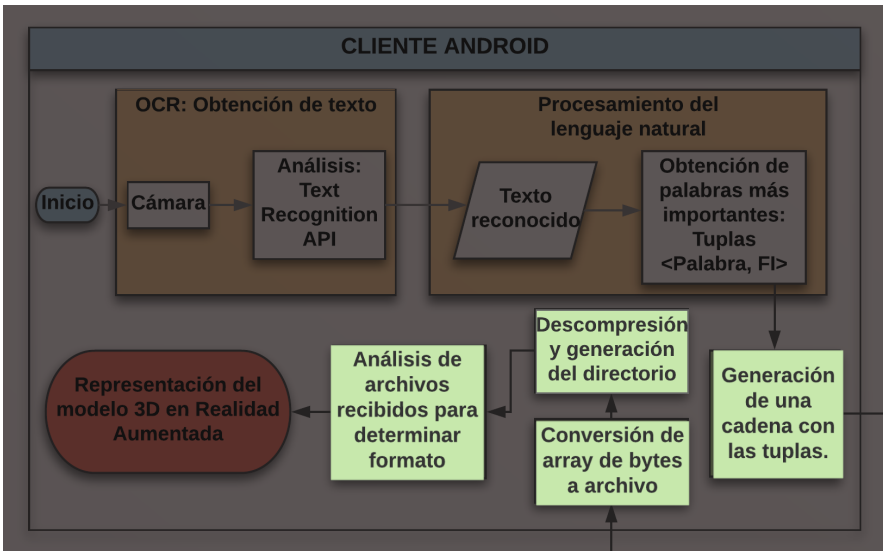


Figura 4.10: Gestión de peticiones y respuestas del Cliente Android.

Para poder hacer la petición al servidor, el resultado almacenado en un *HashMap* debe ser un *String*, por ello es necesario transformarlo mediante la librería `Gson` (ver Listado 4.9). De esta forma el listado de palabras clave junto a su factor de importancia está preparado para que el módulo de comunicación pueda realizar la petición.

```
1 String stringResultado=new Gson().toJson(mapResultado);
```

Listado 4.9: Uso de la librería Gson para pasar *HashMap* a *String*

La gestión de la respuesta consiste en preparar los archivos recibidos para que el módulo de Realidad Aumentada reciba el directorio donde se encuentra el objeto 3D. Si en la respuesta se recibe un código HTTP 200, entonces significa que se ha encontrado un resultado y se ha recibido correctamente. Esta respuesta contiene un archivo comprimido en formato ZIP. Se debe descomprimir y almacenar en una variable el directorio de estos archivos (ver Listado 4.10). Para el manejo de archivos comprimidos se ha utilizado la librería *zt-zip*.

```
1 String path= Environment.getExternalStorageDirectory().getPath();
2 File file=new File(path+"/dataARView/Test1.zip");
3 ZipUtil.unpack(file, new File(path+"/dataARView"));
4 file.delete();
```

Listado 4.10: Descomprimiendo respuesta y preparando directorio

Módulo de Realidad Aumentada

Este módulo forma parte del Cliente Android y el resultado final obtenido es la representación de un modelo 3D en Realidad Aumentada (ver Figura 4.11).

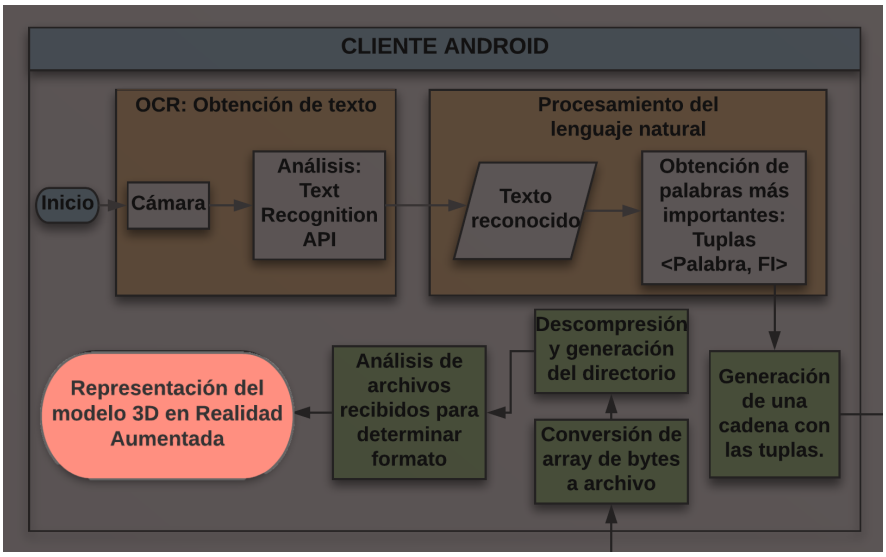


Figura 4.11: Módulo de Realidad Aumentada del Cliente Android.

Para representar el objeto 3D, recibido y descomprimido, en Realidad Aumentada se ha utilizado ViroCore, que es un *framework* 3D para que los desarrolladores creen aplicaciones inmersivas utilizando Java. Si bien las API de nivel inferior como OpenGL requieren que se implementen con precisión algoritmos de representación complejos, ViroCore solo requiere descripciones de escenas de alto nivel y códigos para la interactividad y las animaciones que quieres que la aplicación realice.

El directorio de los archivos descomprimidos del objeto 3D se le pasa a la *activity* mediante los métodos `putExtra` y `getExtras`. En el Listado 4.11 se almacena como un String el directorio para poder manipularlo en el momento de cargar los archivos.

```
1 String pathModelo3d=ViroActivity.this.getIntent().getExtras().getString("
    PathModelo3d");
```

Listado 4.11: Recibiendo el directorio en la activity de RA

Una vez almacenado el directorio de los archivos hay que establecer los parámetros de configuración. El renderizador de Viro permite activar opciones que afectan a las sombras y la iluminación, en este caso la opción de HDR (*high dynamic range*) es la única que esta desactivada en el código pero sin esta opción, la de PBR (*Physically based rendering*) y Bloom (agrega un suave resplandor a las áreas brillantes de la escena) tampoco están activadas (ver Listado 4.12). Lo que hace cada una de estas cuatro opciones del *renderer* es:

- **Bloom:** agrega un suave resplandor a las áreas brillantes de la escena, simulando la forma en que los reflejos brillantes son percibidos por el ojo humano. Esto requiere el uso de múltiples objetivos de renderizado y es intensivo en el índice de llenado. Desactivando esta opción se mejora el rendimiento en dispositivos de gama baja.
- **Dynamic Shadows:** proporcionan pistas visuales sobre la profundidad en la escena y en RA mejoran el realismo de las escenas, dando pistas sobre la posición en el mundo real en la que se encuentran estos objetos virtuales. Desafortunadamente, las sombras tienen un costo de rendimiento, ya que requieren representar la escena en un mapa de sombras y luego hacer referencia a este mapa de sombras durante el ciclo de renderización. Desactivar las sombras dinámica mejora el rendimiento en dispositivos de gama baja.
- **HDR:** Cuando la renderización HDR está habilitada, Viro usa un espacio de color más profundo y renderiza la escena en una textura de punto flotante, lo que permite la preservación de detalles que pueden perderse debido a la limitación de las proporciones de contraste. Se aplica un algoritmo de asignación de tonos después del renderizado para conservar detalles tanto en las regiones brillantes como oscuras de la escena. Si HDR está desactivado, las funciones como Bloom y PBR no funcionan. HDR tiene un costo de rendimiento, por lo que deshabilitarlo mejora el rendimiento en dispositivos de gama baja.
- **PBR:** produce resultados de iluminación más realistas en las escenas mediante la incorporación de un modelo avanzado de luces y materiales del mundo real. PBR produce mejores resultados que el sombreado tradicional y mejora el flujo de trabajo con un conjunto más intuitivo de propiedades materiales. Sin embargo, PBR se basa en algoritmos de GPU avanzados y tiene un costo de rendimiento, por lo que deshabilitarlo mejora el rendimiento en dispositivos de gama baja.

```
1 RendererConfiguration config = new RendererConfiguration();
2 config.setShadowsEnabled(true);
3 config.setBloomEnabled(true);
4 config.setHDREnabled(false);
5 config.setPBREnabled(true);
```

Listado 4.12: Parámetros de configuración de *ViroViewARCore*

Una vez establecida la configuración, en el método `onCreate` inicializa un objeto de la clase `ViroViewARCore`, que es la clase encargada del contenido de la vista principal. En la inicialización de este objeto también se comienza con el establecimiento de la escena en el método `displayScene`, que es el encargado de la creación del *crosshair*[2], búsqueda de superficies y se añade a la escena los nodos de sombras, luces y del objeto 3D.

```
1 private void displayScene() {
2   arScene = new ARScene();
```

[2]Punto de mira donde se quiere poner el modelo 3D.

```
3   arScene.displayPointCloud(true);
4   initARCrosshair(arScene);
5   init3DModelProduct(arScene);
6   initARHud();
7   arScene.setListener(new ARSceneListener());
8   mViroView.setScene(arScene);
9 }
```

Listado 4.13: Método `displayScene` para la inicialización de la escena

En el método `displayScene` se crea el objeto `ARScene` en el que se va a cargar toda la experiencia de RA. Primero, se prepara la escena añadiendo los nodos correspondientes al objeto 3D, sombras, *crosshair*, etc. y, antes de establecer la escena en la vista, se inicia la interfaz de rastreo. Con el método `displayPointCloud` se activan los puntos que toma como referencia para reconocer las superficies.

La función `initARCrosshair` es la encargada de la creación del *crosshair* una vez encontrada una superficie (ver Listado 4.14). Se crea un `Object3D` para cargar el archivo que contiene el modelo del crosshair y se añade como nodo en la escena de RA. Si el objeto se ha cargado correctamente se establece el parámetro *Opacity* a 0 para que no sea visible nada más cargarlo, la escala para que tenga las dimensiones adecuadas y se establece como *ClickListener* en el modelo. En el método `onClick` cambia el *trackingStatus* a `SELECTED_SURFACE`, lo que significa que se ha seleccionado una superficie y que el usuario ha hecho *click* en el modelo, el *crosshair* vuelve a ser invisible y mostraría el objeto 3D (se explicará más adelante).

```
1   private void initARCrosshair(final ARScene scene){
2     final Object3D crosshairModel = new Object3D();
3     scene.getRootNode().addChildNode(crosshairModel);
4     crosshairModel.loadModel(Uri.parse("file:///android_asset/tracking_1.
    vrx"), Object3D.Type.FBX, new AsyncObject3DListener() {
5     @Override
6     public void onObject3DLoaded(Object3D object3D, Object3D.Type type) {
7       mCrosshairModel = object3D;
8       mCrosshairModel.setOpacity(0);
9       mCrosshairModel.setScale(new Vector(0.175,0.175,0.175));
10      mCrosshairModel.setClickListener(new ClickListener() {
11    @Override
12    public void onClick(int i, Node node, Vector vector) {
13      setTrackingStatus(TRACK_STATUS.SELECTED_SURFACE);
14    }
15    }
16  }
```

Listado 4.14: Método `initARCrosshair` para la inicialización del *crosshair*

Para cargar y establecer los valores del objeto 3D iniciales se llama a la función `init3DModelProduct` (ver Listado 4.15). Este método carga los archivos del objeto 3D en un *Object3D*, una vez completada

la carga establece los valores de inicio, *Opacity* a 0 para que no sea visible hasta que el usuario seleccione la superficie, la escala adecuada y se almacena la rotación. Además de la creación de *listener* para que el objeto sea *draggable* y para que se pueda rotar utilizando dos dedos. Por último, se añaden los nodos del objeto 3D cargado y la iluminación, creado en el método `initLightingNode` explicado a continuación.

```
1  private void init3DModelProduct(ARScene scene){
2    mProductModelGroup = new Node();
3    final Object3D productModel = new Object3D();
4    productModel.loadModel(Uri.parse(pathModelo3d), Object3D.Type.OBJ, new
       AsyncObject3DListener() {
5      @Override
6      public void onObject3DLoaded(Object3D object3D, Object3D.Type type) {
7        object3D.setLightReceivingBitMask(1);
8        mProductModelGroup.setOpacity(0);
9        mProductModelGroup.setScale(new Vector(0.001, 0.001, 0.001));
10       mLastProductRotation = object3D.getRotationEulerRealtime();
11     }
12   });

14   mProductModelGroup.setOpacity(0);
15   productModel.addChildNode(initLightingNode());
16   mProductModelGroup.addChildNode(productModel);
17   scene.getRootNode().addChildNode(mProductModelGroup);
18 }
```

Listado 4.15: Método `init3DModelProduct` para cargar el objeto 3D

El nodo con la iluminación se crea con el método `initLightingNode`. En esta función se devuelve un objeto de tipo `Node`, en el cual se añaden dos tipos de iluminación y un nodo para la superficie de las sombras. Los tipos de iluminación que se añaden al nodo son *OmniLight*, que es una fuente de luz que la arroja en todas las direcciones desde una posición determinada, y *Spotlight*, que es una fuente de luz que ilumina un área en forma de cono determinada por su posición y dirección, y que permite mostrar sombras. El nodo para las superficies de las sombras que se añade al nodo final sirve para establecer la rotación, la geometría y la posición de la sombra. En la escena de RA se establece como iluminación de entorno un archivo definido para conseguir más realismo (ver Listado 4.16).

```
1  Texture environment = Texture.loadRadianceHDRTexture(Uri.parse("file
     :///android_asset/wakanda_360.hdr"));
2  arScene.setLightingEnvironment(environment);
```

Listado 4.16: Estableciendo la iluminación de entorno

Por último, para inicializar la escena se necesita crear un HUD, del inglés *head-up display*, que es la Barra de estado. Para ello, en el método `initARHud` se crean las tres parte del HUD que son, el

botón de volver atrás, el botón de la cámara y los mensajes de ayuda. El botón de volver atrás está situado en la parte superior izquierda de la pantalla y simplemente finalizaría la actividad. El botón de la cámara permite al usuario realizar una foto, mostrando un mensaje de éxito en un *Toast* y permitiendo al usuario compartir la imagen en otras aplicaciones, como por ejemplo cualquier app de mensajería actual (ver Listado 4.17). Los mensajes de ayuda se muestran en la parte inferior de la pantalla, para interferir lo menos posible en la visibilidad de la pantalla, y dependiendo del estado muestra un mensaje diferente, por ejemplo, nada más entrar en la ventana de RA mostraría el mensaje «Inicializando AR....» y una vez terminada la carga mostraría «Apunta con la cámara a una superficie plana.» para que el usuario pueda entender que es necesario encontrar una superficie.

```
mViroView.getRecorder().takeScreenShotAsync("screenShot"+ System.nanoTime
    (), true, new ViroMediaRecorder.ScreenshotFinishListener() {
  @Override
  public void onSuccess(Bitmap bitmap, String s) {
    final Intent shareIntent = new Intent(Intent.ACTION_SEND);
    shareIntent.setType("image/png");
    shareIntent.putExtra(Intent.EXTRA_STREAM, Uri.parse(s));
    startActivity(Intent.createChooser(shareIntent, "Share image using"))
    ;
    Toast.makeText(ViroActivity.this, "Captura realizada.", Toast.
   LENGTH_LONG).show();
  }
});
```

Listado 4.17: Captura foto y menú para compartir en otras aplicaciones

Una vez establecidos todos los parámetros de configuración iniciales y construcción de la escena, se deben actualizar de manera constante todos los componentes de la escena para que la experiencia en Realidad Aumentada tenga un comportamiento adecuado. Para ello, los *Listener*, asociados a modelos 3D (*crosshair* u objeto 3D) para desplazar, rotar o responder a un *click* del usuario, actualizan el `TRACK_STATUS` mediante una llamada al método `setTrackingStatus` (ver Listado).

`TRACK_STATUS` es un tipo *enum* para representar un conjunto fijo de constantes. Los estados definidos como constantes en el Listado 4.19 corresponden, por orden, a «buscando superficie», «superficie no encontrada», «superficie encontrada» y «superficie seleccionada».

```
private enum TRACK_STATUS{
  FINDING_SURFACE,
  SURFACE_NOT_FOUND,
  SURFACE_FOUND,
  SELECTED_SURFACE;
}
```

Listado 4.18: Conjunto de constantes definidas para el estado.

El método `setTrackingStatus` es el encargado de actualizar la escena (ver Listado 4.19). Primero, comprueba si el estado es «SELECTED_SURFACE» y si no ha cambiado, entonces no sería necesario ningún tipo de actualización o modificación en el HUD, *crosshair* u objeto 3D. Si no, continua y comprueba si el estado nuevo es «SELECTED_SURFACE», entonces el *listener* del *crosshair* es desactivado. La variable `mStatus`, que representa al estado actual, se actualiza con el nuevo estado. Y por último, para actualizar el HUD, *crosshair* y objeto 3D, se llama a las funciones correspondiente de cada uno para actualizar su contenido.

```
1  private void setTrackingStatus(TRACK_STATUS status) {
2    if (mStatus == TRACK_STATUS.SELECTED_SURFACE || mStatus == status){
3      return;
4    }
5    if (status == TRACK_STATUS.SELECTED_SURFACE){
6      ((ViroViewARCore)mViroView).setCameraARHitTestListener(null);
7    }
8    mStatus = status;
9    updateUIHud();
10   update3DARCrosshair();
11   update3DModelProduct();
12 }
```

Listado 4.19: Método `setTrackingStatus`.

Dependiendo del estado almacenado en la variable `mStatus`, en el método `updateUIHud` se modifica el texto de ayuda mostrado al usuario y la visibilidad del botón de la cámara. En la Figura 4.12 el estado es «SELECTED_SURFACE», es decir, se ha seleccionado una superficie y se está representando un modelo 3D y para ayudar al usuario a comprender que posibilidades tiene se explica que debe utilizar un dedo para mover el objeto y dos para rotarlo.

Figura 4.12: Ejemplo HUD. Mensaje de ayuda y botón de la cámara.

El método encargado de actualizar el *crosshair* dependiendo del estado es `update3DARCrosshair`. Sólo pueden ocurrir dos casos: si el estado actual es «superficie seleccionada» entonces se establece la opacidad a 0 en el modelo 3D del *crosshair* o si el estado actual es «superficie encontrada», entonces oculta los *PointCloud* que servían como referencia para detectar superficies en la escena y establece la opacidad del *crosshair* en 1.

La última actualización que se realiza, cuando hay un cambio de estado, es la del objeto 3D representado. Este método es muy importante, ya que es donde se actualiza la rotación y la posición del objeto 3D. En el método `update3DModelProduct` (ver Listado 4.20), primero comprueba que haya sido seleccionada una superficie previamente, si no establece la opacidad del modelo 3D a 0 para que no sea visible. Si el estado actual indica que hay una superficie seleccionada entonces toma la posición y la rotación en un objeto del tipo *Vector* y establece ambas variables en el objeto 3D. Además, se establece la opacidad a 1 para que sea visible.

```
1 private void update3DModelProduct(){
2   if (mStatus != TRACK_STATUS.SELECTED_SURFACE){
3     mProductModelGroup.setOpacity(0);
4     return;
5   }
6   Vector position = mCrosshairModel.getPositionRealtime();
7   Vector rotation = mCrosshairModel.getRotationEulerRealtime();

9   mProductModelGroup.setOpacity(1);
```

```
    mProductModelGroup.setPosition(position);
    mProductModelGroup.setRotation(rotation);
}
```

Listado 4.20: Función `update3DModelProduct`.

Con el fin de poder mostrar un objeto 3D con animaciones, se ha incorporado un ejemplo de un modelo 3D de una película llamada «*Black Panther*» (ver Listado 4.21). Este modelo trae dos animaciones, una de inicio donde aparece saltando y otra moviendo levemente el cuerpo mientras está en espera. En el caso de este objeto 3D, se crea una variable de tipo `Animation` donde se coge del nodo del objeto 3D la animación «01» correspondiente al salto. Se inicializa el *listener* de esta animación y, en el método `onAnimationFinish` se crea la variable correspondiente a la animación del modelo en espera o *Idle*. Para que esta segunda animación se mantenga en bucle establecemos el método `setLoop` a *true*. Y por último, se llama al método `play` de ambas animaciones para iniciar el movimiento.

```
private void startPantherExperience(){
  final Animation animationJump = mProductModelGroup.getAnimation("01");
  animationJump.setListener(new Animation.Listener(){
    @Override
    public void onAnimationFinish(Animation animation, boolean canceled){
      final Animation animationIdle = mProductModelGroup.getAnimation("02
    ");
      animationIdle.setLoop(true);
      animationIdle.play();
    }
  });
  animationJump.play();
}
```

Listado 4.21: Método para utilizar las animaciones de un objeto 3D.

4.1.2 Cliente Web

La aplicación Web es la encargada de que el usuario pueda compartir sus objetos 3D y visualizar objetos 3D de otros usuarios. Esta solución colaborativa permite que haya un repositorio amplio de objetos 3D en función de la participación de los usuarios, de forma que, el algoritmo de selección de objeto 3D pueda encontrar resultados en las búsquedas y mejorar la calidad de dichos resultados.

La estructura del diseño y desarrollo de esta aplicación web está organizada en cuatro partes (ver Figura 4.13):

- Gestión de usuarios.
- Inserción de modelos 3D con su perfil asociado.
- Gestión de objetos del usuario.

- Visualización de objetos del repositorio.

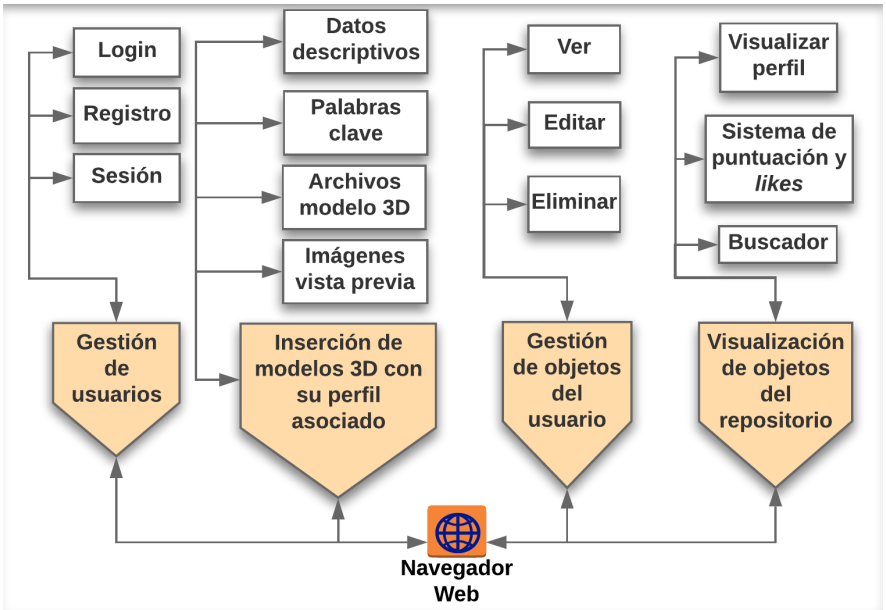


Figura 4.13: Esquema Cliente Web.

Gestión de usuarios

Las sesiones de usuario son importantes para tener acceso a gran parte de las funcionalidades, ya que sin estar logeado el usuario solamente tiene acceso a la visualización de perfiles de objetos 3D subidos por otros usuarios y sin opción a votar ni guardar en favoritos.

Para el manejo de sesiones se ha utilizado el paquete *servlet.http* de la librería *javax*. En concreto, la interfaz *HttpSession*, que proporciona una forma de identificar a un usuario en más de una solicitud de página o visitar un sitio web y almacenar información sobre ese usuario.

El contenedor de servlets usa esta interfaz para crear una sesión entre un cliente y un servidor HTTP. La sesión persiste durante un período de tiempo específico, a través de más de una solicitud de conexión o página del usuario. Una sesión corresponde a un usuario, que puede visitar un sitio muchas veces. El servidor mantiene la sesión mediante el uso de *Session Cookies*. Esta interfaz permite a los servlets:

- Ver y manipular información sobre una sesión, como el identificador de sesión, la hora de creación y la última hora de acceso.
- Enlazar objetos a las sesiones, permitiendo que la información del usuario persista en múltiples conexiones de usuario.

La clase *SessionUtils* se encarga de las operaciones con las sesiones de usuario, para ello necesita acceso al entorno JSF. *SessionsUtils* es un *backbean*, es decir, una clase simple que no conoce nada del resto de la aplicación. Para acceder al entorno JSF utiliza el mecanisco de contexto JSF FacesContext. El FacesContext es una clase que sirve al bean de puente al exterior, ya que le permite acceso no solo

al contexto JSF sino también al contexto HTTP.

Las operaciones definidas en la clase *SessionUtils* son las necesarias para obtener la información de la sesión (ver Listado 4.22). En la *HttpSesion* que se almacena en la sesión está un objeto de tipo User, que contiene los atributos que se definen al establecer la sesión en el registro o en el login explicado a continuación. Con el método `getSession` se extrae la sesión activa en el contexto de JSF y la devuelve como un *HttpSession*. Los métodos `getUserName`, `getUserName` y `getUserSession` extraen datos de la sesión mediante la función `getAttribute` más el nombre del atributo almacenado.

```
1  public class SessionUtils {
2    public static HttpSession getSession() {
3      return (HttpSession) FacesContext.getCurrentInstance().
       getExternalContext().getSession(false);
4    }
5    public static String getUserName() {
6      HttpSession session = (HttpSession) FacesContext.getCurrentInstance()
       .getExternalContext().getSession(false);
7      return session.getAttribute("user").toString();
8    }
9    public static String getUserId() {
10     HttpSession session = getSession();
11     if (session != null)
12       return (String) session.getAttribute("userid");
13     else
14       return null;
15   }
16   public static User getUserSession() {
17     HttpSession session = SessionUtils.getSession();
18     User user = ((User) session.getAttribute("user"));
19     return user;
20   }
21 }
```

Listado 4.22: Clase *SessionUtils*.

La sesión tiene dos formas de crearse, con el registro de nuevos usuarios o con el *login*. El registro de usuarios se realiza mediante la clase de modelo *User*. Para ello en la vista *register.xhtml*, cuando el usuario introduce los datos y completa el formulario se actualizan los valores del modelo *User*. Los valores que guarda en *User* son el *email* y la *password* si cumple con los requisitos definidos en la validación de la contraseña. La validación de contraseñas la hace mediante el componente «validator» de JSF que se define dentro del componente para escribir la contraseña (ver Listado 4.24).

```
1 <f:validator validatorId="passwordValidator" />
2 <f:attribute name="confirmPassword" value="#{confirmPassword}" />
```

Listado 4.23: Inserción de componente *validator* de JSF.

El bean *passwordValidator* es el encargado de hacer la comprobación de la validación de contraseñas. Implementa la clase *Validator* de JSF que permite integrar esta comprobación correctamente en la vista del registro. Las comprobaciones que realiza es que la contraseña no esté vacía o sea nula, y que la contraseña y la contraseña de confirmación coincidan. Para que se pueda llamar desde el componente de JSF en la vista se difine la clase de validación mediante la anotación «@FacesValidator("passwordValidator")».

```
1  @FacesValidator("passwordValidator")
2  public class PasswordValidator implements Validator {
3    @Override
4    public void validate(FacesContext context, UIComponent component,
       Object value) throws ValidatorException {
5      String password = value.toString();
6      UIInput uiInputConfirmPassword = (UIInput) component.getAttributes().
       get("confirmPassword");
7      String confirmPassword = uiInputConfirmPassword.getSubmittedValue().
       toString();
8      //Primero comprueba que se ha escrito algo
9      if (password == null || password.isEmpty() || confirmPassword == null
       || confirmPassword.isEmpty()) {
10       return;
11     }
12     //Comprueba que ambas contrasenas coinciden
13     if (!password.equals(confirmPassword)) {
14       uiInputConfirmPassword.setValid(false);
15       throw new ValidatorException(new FacesMessage("Las contrasenas no
       coinciden. Vuelve a intentarlo."));
16     }
17   }
18 }
```

Listado 4.24: Componente *validator* de JSF.

El formulario de registro llama al método `registrar` de la clase *User*. En este método se realizan dos acciones (ver Listado 2), una es insertar los datos en la base de datos y otra es establecer la sesión del nuevo usuario. La inserción de usuario en la base de datos se realiza mediante la clase de persistencia *DAOUsers* y utiliza un procedimiento almacenado encargado de insertar usuarios y encriptar la contraseña con un algoritmo SHA1 combinado con operaciones UNHEX, que interpreta cada par de dígitos hexadecimal como número y lo convierte a carácter. El valor es devuelto como una cadena de 40 digitos hexadecimales.

Proceso 2 Método `registrar`.

1: Insertar nuevo usuario
2: $session \leftarrow getSession$
3: $session \leftarrow establecer\_atributo(user)$
4: Redirigir vista a index

La cabecera de la página depende del estado de la sesión. Si existe una sesión entonces devuelve una cabecera con todas las opciones disponibles (Mis favoritos, Mis objetos, Subir nuevo objeto y Desconectar) y en el caso de que no exista tendría opción de hacer login o de registrarse como en la Figura 4.15. Para introducir la cabecera en la vista JSF utiliza un *facelet*, es decir, un template o plantilla que permite ser incluido. En el caso de esta cabecera, como existe la opción de que exista una sesión o no, se han creado dos *facelets*. El método `getHeaderName` de la clase *Utilities* es el encargado de comprobar la sesión y devolver el *facelet* adecuado. La manera en la que esta incluida en todas las vistas la cabecera es la siguiente:

```
1 <ui:include src="#{utilities.headerName}"></ui:include>
```

Listado 4.25: Inserción de cabecera en las vistas JSF.

En cuanto al funcionamiento del método `getHeaderName` de la clase *Utilities*, realiza la comprobación de que la sesión actual exista, en caso afirmativo devolvería la cadena «header2.xhtml» que contiene la cabecera con todas las opciones disponibles para un usuario logeado como en la Figura 4.14, y en caso negativo, devuelve la cadena «header.xhtml», que es un *template* que contiene las opciones para un usuario sin logear, como en la Figura 4.15.

```
1 <ui:include src="#{utilities.headerName}"></ui:include>
```

Listado 4.26: Inserción de cabecera en las vistas JSF.

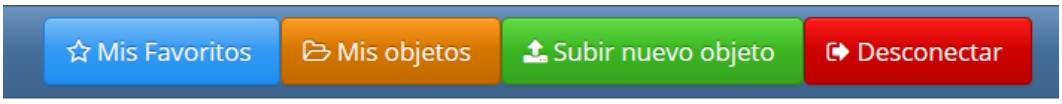


Figura 4.14: Cabecera con usuario logeado.

Figura 4.15: Cabecera login - Usuario no logeado.

Para realizar el login de un usuario ya existente se ha incorporado en la cabecera de la vista la posibilidad de introducir el *email* y la *password* (ver Figura 4.15). El formulario de login de la cabecera almacena el email en la variable *email* de User y la contraseña en la variable *password*, y al ejecutar el *Submit* del botón «Acceder» ejecuta el método `login` de la clase *User* vinculado al listener de este componente. El método `login` es el encargado de contruir la petición para hacer la llamada a la clase de persistencia y comprobar en la lista de usuarios de MySQL si los credenciales introducidos son correctos. Como el listado de usuarios esta almacenado como «user» más el ID del usuario en la tabla Users, es necesario hacer un «SELECT» mediante el email introducido. La comprobación la realiza el método de la clase *DAOUsers*, si devuelve un valor mayor que 0 significa que ha logeado correctamente.

Proceso 3 Método `login` de la clase *Users*.

1: $user \leftarrow selectByEmail$
2: $nuevoId \leftarrow mysql\_login(userid, password)$
3: **si** $nuevoId > 0$ **entonces**
4: $session \leftarrow user$
5: Redireccionar a index
6: **si no**
7: Mostrar mensaje de error “Credenciales incorrectos”
8: **fin si**

La existencia de sesión también afecta a la funcionalidad de puntuar y de favoritos en la visualización de los objetos, de tal forma que, si un usuario no está logeado no tiene permisos para votar ni darle a «Me gusta», ya que esta opción se encuentra deshabilitada.

Para terminar la sesión el usuario puede deslogear pulsando el boton «Desconectar» de la cabecera (ver Figura 4.14). Este botón ejecuta el método `logout` de la clase *User*, que tomando el contexto de la vista de JSF con *FacesContext* permite invalidar la sesión y el *browser* de este usuario estaría desvinculado con ninguna sesión en este sitio web (ver Listado 4.27).

```
1 public void logout() {
2   FacesContext.getCurrentInstance().getExternalContext().
      invalidateSession();
3   FacesContext.getCurrentInstance().getApplication().getNavigationHandler
      ().handleNavigation(FacesContext.getCurrentInstance(), null, "index.
      xhtml");
4 }
```

Listado 4.27: Método `logout` - Invalidar sesión.

Inserción de modelos 3D con su perfil asociado

La vista JSF de la aplicación para insertar modelos 3D en el repositorio sólo es accesible para usuarios registrados y con sesión iniciada. La interfaz (ver Figura 4.16) está compuesta por tres campos para rellenar con datos descriptivos del objeto 3D y la selección de categoría, y en la parte de la derecha por dos componentes para subida de múltiples archivos. Uno para subida de archivos en formato OBJ, MTL, FBX o VRX (son los formatos integrados en el módulo de Realidad Aumentada de la aplicación Android) y otro para las imágenes utilizadas en la vida previa de los perfiles de modelos 3D en el repositorio.

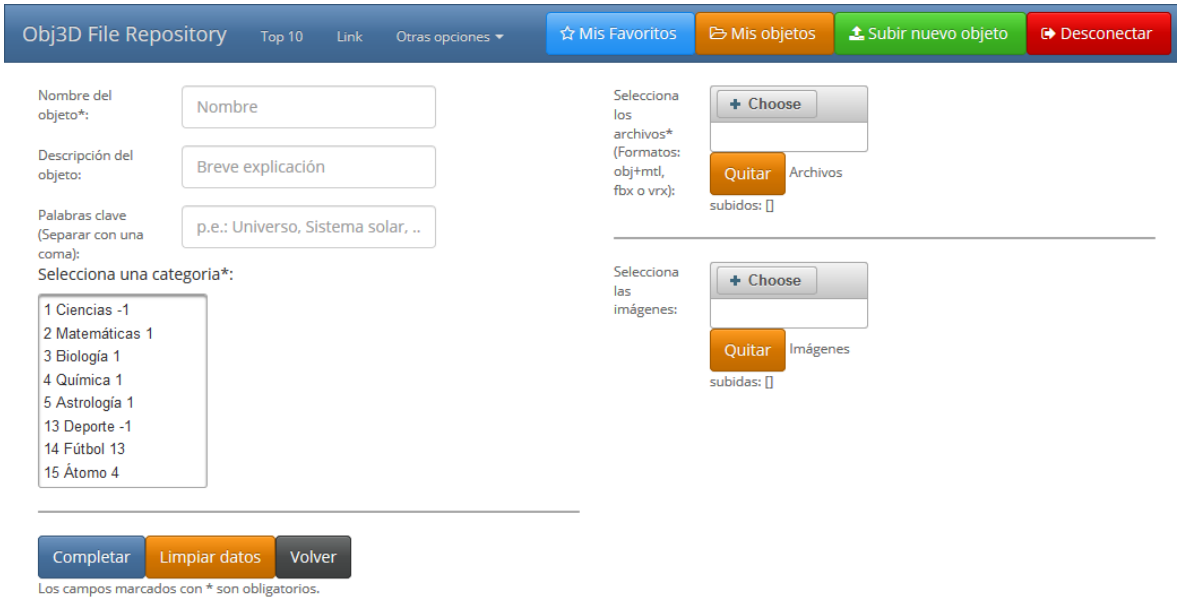


Figura 4.16: Vista para insertar modelos 3D en el repositorio.

Para la subida de archivos e imágenes se ha utilizado el componente *fileUpload* de *Primefaces*. Este componente tiene asociado como métodos de escucha a los eventos de subida de archivos el método `handleFileUpload`, para los archivos, y `handleImgUpload`, para las imágenes de la vista previa. Ambos métodos son de la clase *Item* (ver Listado 4).

Proceso 4 Subida de archivos e imágenes.

Entrada: event: evento recibido del componente fileUpload

1: Establecer el valor de owner en Item
2: $uploadedFile \leftarrow event\_getFile$
3: $file \leftarrow establece\_directorio$
4: **si** No existe el directorio **entonces**
5: Crear directorio
6: **fin si**
7: Escribir uploadedFile en directorio
8: Añadir nombre_archivo a Item

Gestión de objetos del usuario

En esta vista llamada «myItems.jsf» es donde el usuario puede visualizar sus objetos y gestionarlos más fácilmente, de manera que puede verlos y eliminarlos desde el listado. Las opciones de gestión toman los datos de un «service» que proporciona un listado de *items*, este servicio es una especie de factoría que se gestiona como una propiedad del *ManagedBean*. En esta vista se ha definido asi: «@ManagedProperty("#itemService")», de forma que se pueda hacer referencia a ella.

Figura 4.17: Visualización de perfil del objeto 3D.

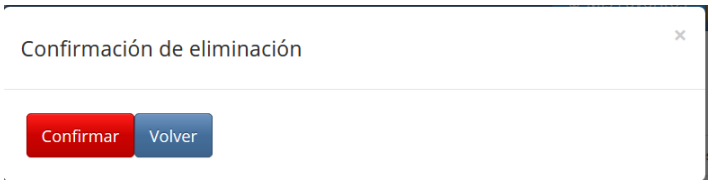


Figura 4.18: Eliminación de perfil del objeto 3D.

Visualización de objetos del repositorio

La vista principal de la aplicación web es donde se visualizan todos los perfiles de objetos 3D disponibles en la base de datos. Esta vista está disponible para todos los usuarios independientemente de si tienen una sesión activa o no. La única restricción es que un usuario sin sesión activa, es decir, no logeado, no puede votar ni darle a «Me gusta».

El nombre de esta vista JSF es «index.xhtml». Esta formada, principalmente, por la plantilla de la cabecera o «header» y un componente *dataTable* de la librería Bootsfaces. Ambas partes se ven afectadas por la sesión, la cabecera muestra opciones adicionales explicadas en la sección de la Sesión y el componente de la tabla con los datos de los objetos, que la opción de puntuar y de *Likes* está desactivada si no hay sesión activa.

Figura 4.19: Visualización del repositorio.

Cuando la sesión esta activa el sistema de puntuaciones y *likes* pasa a estar activo, en la Figura 4.20 se muestran las dos situaciones.

(a) Sesión inactiva (b) Sesión activa

Figura 4.20: Funcionalidad de puntuaje y *likes*.

Para visualizar los perfiles de los objetos utiliza la misma interfaz que en la Figura 4.17.

4.1.3 Servidor - Algoritmo de Selección de Obj 3D y SGBD

El servidor es el encargado de encontrar los objetos candidatos conforme a las palabras clave recibidas por el cliente Android, seleccionar cuál es el resultado más óptimo mediante el algoritmo de selección de objeto 3D y a su vez de realizar las llamadas al sistema gestor de bases de datos necesarias para que el cliente Web y Android tengan acceso a la información. Por lo tanto, esta sección estará dividida en dos subsecciones: algoritmo de selección de objeto 3D y sistema gestor de base de datos.

Algoritmo de Selección de Objeto 3D

Este algoritmo tiene como función principal generar un único resultado, el más adecuado para el texto analizado y el mejor valorado de los candidatos encontrados en el repositorio. Un resultado es un objeto 3D que es compatible con el formato necesario para reproducirlo en Realidad Aumentada en la aplicación Android. El sistema diseñado tiene como objetivo permitir el acceso a un gran número de usuarios, por lo tanto, sería un caso muy común que existieran múltiples objetos candidatos con *keywords* idénticas. Para ello, se ha diseñado un algoritmo que permite realizar una valoración detallada de los objetos que posibilita diferenciar los objetos 3D más allá de las coincidencias de palabras clave.

Antes de entrar en detalle es necesario entender el flujo del algoritmo (ver Figura 4.21). El cliente Android envía un listado de tuplas que contiene palabras clave y su factor de importancia correspondiente. Este listado es el argumento de entrada que utiliza el algoritmo de selección de objeto 3D. Un ejemplo es el Listado 4.28, obtenido de la definición de «Planeta Tierra». En el listado tenemos tuplas compuestas por palabras y su factor de importancia que han sido obtenidas mediante el proceso explicado anteriormente en la Sección 4.1.1. La suma de todos los factores de importancia es 1.

```
{
  "Sol": 0.050592184,
  "orbita": 0.06347552,
  "estrella": 0.07293502,
  "Tierra": 0.4151733,
  "sistema solar": 0.397824
}
```

Listado 4.28: Ejemplo del listado de entrada que recibe el Algoritmo.

La Figura 4.21 contiene el diagrama de flujo o actividades del algoritmo de selección de objeto 3D, donde se muestra detalladamente los pasos del algoritmo que conectan los puntos de inicio y de fin del proceso.

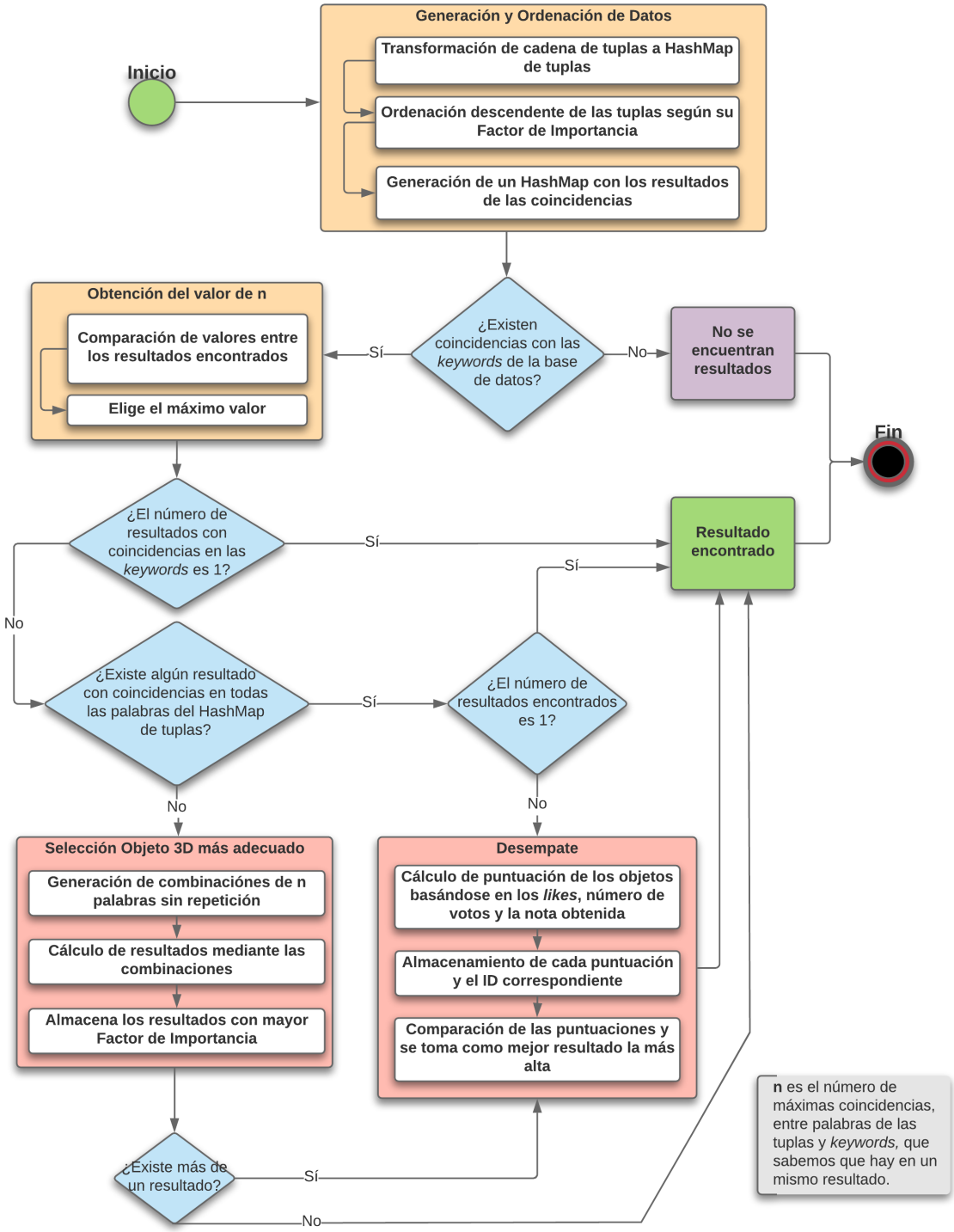


Figura 4.21: Diagrama de flujo - Algoritmo de Selección de Objeto 3D.

La generación y ordenación de los datos en base al listado recibido por el cliente Android se divide en tres fases:

1. Transformación del listado recibido a una estructura de datos adecuada, como *HashMap*, para realizar la búsqueda.

2. Ordenación de forma descendente de las tuplas de la nueva estructura de datos.
3. Generación de un nuevo HashMap con los resultados candidatos.

El listado recibido está almacenado en un *String* por lo que es necesario transformarlo en otra estructura de datos donde se pueda ordenar por Factor de Importancia en orden descendente, ya que uno de los factores de decisión está basado en este orden. Para facilitar la transmisión del listado de tuplas como texto plano entre el cliente Android y el servidor, se ha utilizado la librería Gson de Google, que permite la serialización y deserialización entre objetos Java y su representación en notación JSON. Para transformar el *String* recibido en un *HashMap* se realiza llamando al método `fromJson` y pasándole como argumentos la cadena a convertir y el tipo de estructura que se necesita, en este caso es *HashMap <String, Double>* (ver Listado 4.29).

```
1  //transforma el String recibido y guarda el resultado en un HashMap
2 HashMap<String, Double> mapResultado = new Gson().fromJson(getStrMap(),
      new TypeToken<HashMap<String, Double>>() {}.getType());
```

Listado 4.29: Transformación del listado.

Para realizar el desempate entre varios objetos se utiliza el HashMap de palabras clave y factores de importancia ordenado de manera descendente según el *value*, es decir, su factor de importancia en este caso. Desde la versión 8 de Java se puede utilizar un método que permite ordenar estructuras mediante un *Comparator* (ver Listado 4.30). Además, permite aplicar la función `reversed` y de esta manera se obtiene el *HashMap* ordenado de manera descendente.

```
1 //ordena el contenido del HashMap en orden descendente
2 //por su Factor de Importancia
3 HashMap<String, Double> sortedMapResultado = mapResultado.entrySet()
4 .stream()
5 .sorted(Entry.<String, Double>comparingByValue().reversed())
6 .collect(Collectors.toMap(Entry::getKey, Entry::getValue, (e1, e2) -> e1,
      LinkedHashMap::new));
```

Listado 4.30: Ordenación descendente.

Ya se ha generado el *HashMap* necesario para inicializar la búsqueda de coincidencias. Para realizar la primera búsqueda y comprobación hace una llamada al método `busquedaArrayCoincidencias` que recibe el *HashMap* ordenado (ver Listado 4.31). En este método se utiliza un *iterator* para tomar los valores de cada *entrySet* e iterar en el bucle *while* hasta que no haya más. En la variable *keywordArrayList* se almacenan estos valores, que son las palabras que se van a comprobar, por ejemplo, en la definición de planeta tierra sería «Sol», «orbita», «estrella», «Tierra» y «sistema solar».

En cada iteración del bucle *while* se comprueba si existen en la tabla *keywords* de la base de datos y se almacena el *itemId* por cada palabra encontrada. Para ello, hace una llamada al método `searchByArray` que realiza una petición SQL con la cadena «SELECT itemId FROM keywords WHE-

RE keywords LIKE ?» donde «?» es la palabra clave que quiere buscar. El *arrayList* obtenido de este método es una lista de enteros, donde cada entero es un *itemId* que tiene coincidencia de una palabra clave del listado inicial con una palabra clave de la tabla *keywords* en la base de datos.

```
1  private ArrayList<Integer> busquedaArrayCoincidencias(HashMap<String,
       Double> listaInicial) {
2    Iterator<Entry<String, Double>> it = listaInicial.entrySet().iterator()
       ;
3    ArrayList<String> keywordArrayList = new ArrayList<>();
4    ArrayList<Integer> resultadoBusqueda = new ArrayList<>();
5    try {
6      while(it.hasNext()) {
7        keywordArrayList.add(it.next().getKey().toString());
8      }
9      resultadoBusqueda = DAOKeywords.searchByArray(keywordArrayList);
10   } catch (Exception e) {
11     e.printStackTrace();
12   }
13   Collections.sort(resultadoBusqueda);
14   return resultadoBusqueda;
15 }
```

Listado 4.31: Método `busquedaArrayCoincidencias`.

Si el listado que devuelve el método `busquedaArrayCoincidencias` esta vacio significa que no ha encontrado coincidencias, entonces termina la ejecución del método `init` y devuelve 0. Ningún *itemId* corresponde a 0, por lo que el cliente Android recibe una respuesta indicando que no ha encontrado resultados.

```
1    ArrayList<Integer> listaItemId = busquedaArrayCoincidencias(
       sortedMapResultado);
2    if (listaItemId.isEmpty()) return 0;
```

Listado 4.32: Array de coincidencias vacio.

En el caso de haber encontrado resultado se pasa a la obtención del valor de ***n***. Este valor es el número de máximas coincidencias entre palabras de las tuplas y *keywords* en la base de datos, que sabemos que hay en un mismo resultado. Para obtenerlo es necesario generar un *HashMap* con las palabras y su número de coincidencias. Por ejemplo, si sólo encontrara el *itemID*=66 y el *itemID*=41 y tuviera 3 y 4 coincidencias respectivamente, almacenaría en el *HashMap* los valores: "66=3, 41=4"(ver Figura 4.22).

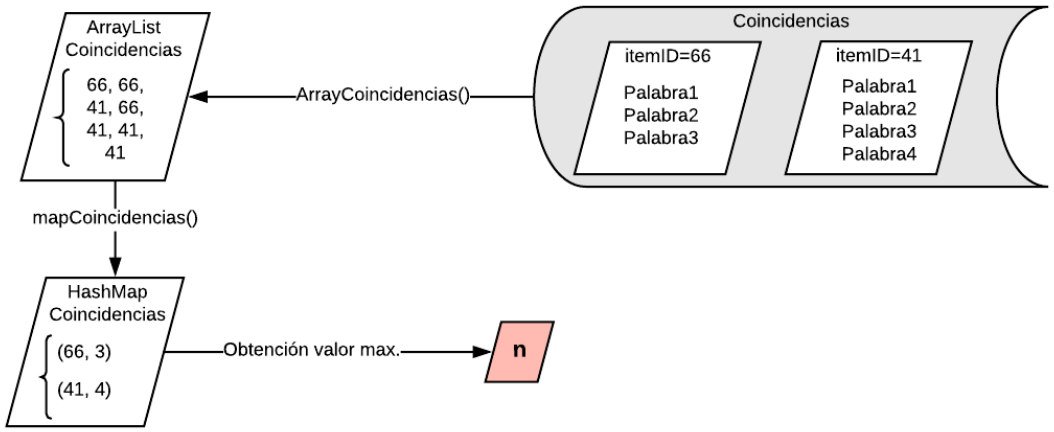


Figura 4.22: Ejemplo de obtención del valor de ***n***.

El método encargado de generar este *HashMap* con las coincidencias es `mapCoincidencias` (ver Listado 5).

Proceso 5 Generación de frecuencia de coincidencias.
Entrada: Listado de coincidencias.
Salida: resultado: HashMap con la frecuencia de coincidencias.

```
1:  Inicializar HashMap resultado
2:  Entero contador
3:  para Entero id = cada valor del listado de coincidencias hacer
4:      contador ← 0
5:      si id no existe en el resultado entonces
6:          para i < tamano_listado hacer
7:              si id = listado[i] entonces
8:                  contador+1
9:              fin si
10:         fin para
11:     fin si
12:     Escribir id y contador en resultado
13: fin para
14: devolver resultado
```

Una vez obtenido el *HashMap* con la frecuencia de los items que tienen coincidencias se realiza una comparación de los valores mediante la clase *Collections*, que permite operar con estructuras

como los *HashMaps*, para obtener el máximo, es decir el valor **n** (ver Listado 4.33).

```
Collections.max(hmCoincidencias.values());
```

Listado 4.33: Máximo número de coincidencias.

Para obtener el resultado final con el método `obtenerItemRespuesta` es necesario generar un *ArrayList* de enteros que contenga el itemID o los itemID del *HashMap* con value igual a ***n***. Esto es calculado en el método `repetidosEnArray` que toma como entrada el *HashMap* con las coincidencias (ver Listado 6).

Proceso 6 Generación de listado de *items* con más coincidencias.
Entrada: HashMap con la frecuencia de coincidencias.
Salida: resultado: ArrayList de enteros con los itemID que tienen más coincidencias.

1: Inicializar ArrayList resultado
2: $maxValueInMap \leftarrow n$
3: **para** cada valor_entrySet que hay en el HashMap **hacer**
4: **si** valor_entrySet = maxValueInMap **entonces**
5: Añadir clave al resultado
6: **fin si**
7: **devolver** resultado
8: **fin para**

Las entradas del método `obtenerItemRespuesta` son un *ArrayList* que contiene los *items* más repetidos, este objeto es el generado con el método `repetidosEnArray`, y el *HashMap* obtenido en la salida del método `mapCoincidencias`. Para conseguir el resultado en esta función, se pretende descartar los objetos sin tener que realizar el *desempate*, de esta forma se optimiza la ejecución total de la aplicación. Esta parte del algoritmo de selección de objeto 3D es la más importante, ya que es donde se produce casi toda la carga lógica del algoritmo (ver Listado 7).

La toma de decisiones sigue un flujo que permite reducir el coste de las búsquedas en algunos casos. Si existe algún *item* que tenga por coincidencia todas las palabras recibidas del cliente Android entonces no es necesario realizar combinaciones, entonces si existe un único caso ya tendríamos el resultado y si es más de uno habría que realizar un *desempate*. En el caso de que no exista ninguna coincidencia que contenga todas las palabras habría que buscar combinaciones menores que el total de palabras. Si tras generar la combinación con mayor valor en la suma de Factores de importancia sigue habiendo más de un resultado entonces se realiza un *desempate*.

Proceso 7 Obtención del itemID del resultado final.

Entrada: arrayItemsEmpatados: *ArrayList* de *items* más repetidos, map: *HashMap* con la frecuencia de coincidencias.

Salida: valorFinal: itemID del resultado final.

1: Inicializar valorFinal = 0
2: $maxReps \leftarrow n$
3: **si** n = tamaño de map **entonces**
4: **si** número de *items* con coincidencia en todas las palabras = 1 **entonces**
5: $valorFinal \leftarrow arrayItemsEmpatados[0]$
6: **si no**
7: $valorFinal \leftarrow desempate(arrayItemsEmpatados)$
8: **fin si**
9: **si no**
10: **si** tamaño de arrayItemsEmpatados = 1 **entonces**
11: $valorFinal = arrayItemsEmpatados[0]$
12: **si no**
13: $resultadoGenCombinaciones \leftarrow generarCombinaciones()$
14: **para** las combinaciones en resultadoGenCombinaciones **hacer**
15: $palabrasEncontradas \leftarrow busquedaPorCombinaciones$
16: **si** $palabrasEncontradas \neq nulo$ **entonces**
17: Termina la búsqueda, termina bucle
18: **fin si**
19: **fin para**
20: **si** $palabrasEncontras > 1$ **entonces**
21: $valorFinal = desempate(arrayEmpatados)$
22: **si no**
23: $valorFinal = arrayItemsEmpatados[0]$
24: **fin si**
25: **fin si**
26: **fin si**
27: **devolver** valorFinal

Para entender por completo el funcionamiento del método `obtenerItemRespuesta` es necesario explicar los métodos que utiliza para obtener las combinaciones y realizar el desempate.

Las combinaciones sin repetición o combinaciones ordinarias de ***m*** elementos tomados de orden ***n*** son los distintos grupos de ***n*** elementos distintos que se pueden hacer con los ***m*** elementos que tenemos, de forma que dos grupos se diferencian en algún elemento y no en el orden de colocación. El método `generarCombinaciones` es el encargado de realizar estas combinaciones. El resultado generado es la combinación de palabras basada en el número máximo de coincidencias encontradas y sólo crea

la primera combinación, porque ya se ha comprobado que existen resultados con esas palabras. Por lo tanto, el resultado es la combinación de palabras con la suma de Factores de Importancia más alto.

Proceso 8 Generar combinaciones sin repetición de n elementos. (7.1)

Entrada: arr: Array de palabras, n: Número de palabras en cada combinación.

Salida: resultadoGenCombinaciones: ArrayList de listas de palabras.

1: **si** $n = 0$ **entonces**
2: $resultadoGenCombinaciones \leftarrow palabrasGeneradas$
3: **devolver** resultadoGenCombinaciones
4: **fin si**
5: **para** i <= longitud de arr - n **hacer**
6: $palabrasGeneradas[palabrasGeneradas.length - n] \leftarrow arr[i]$
7: generarCombinaciones(arr, n-1, i + 1, palabrasGeneradas)
8: **fin para**

En algunos de los casos existen resultados encontrados en los que coinciden la misma cantidad de palabras y las mismas palabras exactamente, por lo que es necesario realizar un desempate en base a otras características, como en el Ejemplo 4.1.

Ejemplo 4.1 *Tomando como valor de entrada en el algoritmo la tupla del Listado 4.28, si existiesen sólo dos items con coincidencias y además ambos coinciden en las palabras «Tierra» y «sistema solar». En estas circunstancias es necesario calcular una puntuación en base a otros factores, es decir un desempate, de forma que sólo exista un resultado y sea el más adecuado.*

Los usuarios que acceden a la página web tienen la posibilidad de puntuar los objetos y de guardarlos como favoritos, por lo tanto, se utiliza el número de votos, los *likes* y la nota de estos *items* para que, mediante una fórmula matemática, se genere una puntuación y así saber cual es mejor resultado.

El método **`desempate`** recibe como argumentos un *ArrayList* de enteros que contiene los *itemID* de los *items* a desempatar (ver Listado 9). Primero, se obtiene los valores de cada *item*, se guardan en un objeto *Item* y se separan cada uno en *arrayVotos*, *arrayLikes* y *arrayNotasMedias* (que es la suma de las puntuaciones entre el número de votos). Además, es necesario obtener la nota más alta, el número más alto de *likes* y el más alto de votos. Para ello, se utiliza la función `max` de la clase *Collections* de Java, que permite obtener el valor más alto de las listas. Para obtener el resultado final se calcula la puntuación llamando al método `calcularPuntuación` y se almacena en un *HashMap* el *itemid* junto a la puntuación calculada. El resultado es la más alta de estas puntuaciones.

Proceso 9 Método `desempate`.

Entrada: arrayEmpatados: ArrayList de enteros que contiene los itemID.

Salida: itemResultado: entero del itemID resultado.

1: Inicializar arrayDesempateItems, arrayVotos, arrayLikes, arrayNotasMedias, puntuacionesItems
2: $itemResultado \leftarrow 0$
3: **para** cada itemid de arrayEmpatados **hacer**
4: Inicializar Item
5: $item \leftarrow selectVotosLikesNotaById(itemid)$
6: Añadir Item a arrayDesempateItems
7: **fin para**
8: **para** cada Item de arrayDesempateItems **hacer**
9: Añadir votos de Item a arrayVotos
10: Añadir Likes de Item a arrayLikes
11: Añadir getNotaMedia de Item a arrayNotasMedias
12: **fin para**
13: $votosMax \leftarrow max(arrayVotos)$
14: $likesMax \leftarrow max(arrayLikes)$
15: $notaMax \leftarrow max(arrayNotasMedias)$
16: **para** cada Item de arrayDesempateItems **hacer**
17: Añadir a puntuacionesItems (itemID, calcularPuntuacion(Item, votosMax, likesMax, notaMax))
18: **fin para**
19: $puntuacionMax \leftarrow max(valores\_puntuacionesItems)$
20: **para** cada entrySet de puntuacionesItems **hacer**
21: **si** Comprobar si coincide con puntuacionMax **entonces**
22: $itemResultado \leftarrow key_MaxValue$
23: **fin si**
24: **fin para**
25: **devolver** itemResultado.

La fórmula matemática para calcular la puntuación se define en base a las siguientes variables:

- **notaMedia:** este valor está calculado con la suma de las puntuaciones totales del item entre el número de votaciones.
- **notaMax:** es la nota más alta de todos los items que participan en el desempate.
- **likes:** es el número de personas que les ha gustado el item y lo tienen en favoritos.
- **likesMax:** este valor es el correspondiente al número más alto de likes de los items que participan en el desempate.

- **votos:** es el número de veces que ha sido votado el item.
- **votosMax:** este valor corresponde al número de votos más alto entre los items que participan en el desempate.

La fórmula del método `calcularPuntuacion` está hecha teniendo en cuenta que el sistema de votaciones. Tiene un rango de valores de 0 a 5 y el número máximo de votos y likes por el mismo usuario es de 1 para cada item. Una mala nota media penalizaría menos que si un item tiene pocas votaciones o likes.

$$puntuación = \frac{notaMedia}{notaMax} + \frac{likes}{votos} + \frac{votos}{votosMax} \times \frac{likes}{likesMax} \tag{4.1}$$

Ejemplo 4.2 *Tomando como valor de entrada en el algoritmo las tuplas del Listado 4.28 y teniendo en la tabla Items los valores de la Figura 4.23.*

idItem	nombreObj	descripcion	categoria	owner	votos	likes	sumaPuntuaciones
41	Tierra	Tierra animada	1	42	12	3	19
42	Pantera		5	42	6	1	25
43	Perro	Pitbull	2	6	1	2	3
44	Leon		5	42	0	1	0
45	AC130	Avion	14	42	2	0	7
46	Gato	Gato persa	2	42	1	0	5
47	Hexagon		1	6	1	3	5
64	Corazon		5	42	2	1	4
65	Motor	Motor v10	15	42	0	0	0
66	Tierra		1	42	2	1	7

Figura 4.23: Ejemplo de valores de la tabla *Item* en la base de datos.

Se busca coincidencias y encuentra el itemID=66 y el itemID=41.
Ambos items tienen 2 coincidencias, por lo tanto el algoritmo continua.
La combinación de 2 palabras es la misma, «Tierra» y «sistema solar».
Se procede al desempate en base a la Fórmula 4.1.
Recordemos que: notaMedia = $\frac{sumaPuntuaciones}{votos}$ *y notaMax = notaMedia más alta.*

$$puntuacionItem41 = \frac{\frac{19}{12}}{\frac{7}{2}} + \frac{3}{12} + \frac{12}{12} \times \frac{3}{3} = 1{,}7023809523809523 \tag{4.2}$$

$$puntuacionItem66 = \frac{\frac{7}{2}}{\frac{7}{2}} + \frac{1}{2} + \frac{2}{12} \times \frac{1}{3} = 1{,}5555555555555556 \tag{4.3}$$

Solución: *la puntuación del item con ID=41 es mayor, por lo tanto, en base a los cálculos del método*

`desempate` *el Algoritmo de Selección de Objetos 3D da como resultado más adecuado al item 41.*

La conclusión del Ejemplo 4.2 es que lo que más valor tiene en un *item* es la cantidad de *votos* y de *likes*, por otra parte la *notaMedia* y la proporción de *likes* respecto a *votos* también suma pero no es tan determinante, de esta manera se valora sobretodo que a los usuarios les guste ese objeto y que sea popular.

Sistema Gestor de Base de Datos

La base de datos es relacional ya que archiva los datos en diferentes tablas, en lugar de almacenar todos los datos en un único archivo. Esto aporta mejoras de eficiencia y flexibilidad. Entre las tablas se definen relaciones que permiten combinaciones de datos de diferentes tablas (ver Figura 4.24). Los datos están organizados en ocho tablas:

- **items:** es la tabla principal, donde están definidos los datos descriptivos del objeto 3D. Tiene como FK (*foreign key*) la columna *owner* con *idUser* de la tabla *users*.
- **users:** contiene el email del usuario y la contraseña encriptada mediante el procedimiento almacenado `insertarUsuario` explicado a continuación. No tiene ninguna FK.
- **keywords:** almacena las palabras clave y el *itemId* que le corresponde. El *id* del *item* es una FK relacionada con la columna *idItem* de la tabla *items*.
- **likes:** almacena los *likes* otorgados por usuarios a *items* de otros usuarios o propios. Contiene el *userId* del usuario que ha otorgado el *like* y el *itemId* del *item* que lo ha recibido. Ambas son claves foráneas, *userId* con *idUser* de la tabla *users* y *itemId* con *idItem* de la tabla *items*.
- **scores:** es parecida a la tabla *likes*, almacena las puntuaciones otorgadas por usuarios a *items* de otros usuarios o propios en la columna *score*. Contiene el *userId* del usuario que ha otorgado el like y el *itemId* del *item* que lo ha recibido. Ambas son claves foráneas, *userId* con *idUser* de la tabla users y *itemId* con *idItem* de la tabla items.
- **images:** en esta tabla se almacena el nombre del archivo de imagen subido del objeto 3D. Para relacionarlo con su objeto tiene una FK de *itemId* con la columna *idItem* de la tabla *items*.
- **files:** realiza la misma función que la tabla *images*, almacena el nombre de los archivos del objeto 3D. Para relacionarlo con su objeto tiene una FK de *itemId* con la columna *idItem* de la tabla *items*.
- **category:** esta tabla no tiene claves relacionadas con ninguna de las otras tablas. Contiene las categorías padres e hijos, y para saber que categoría es la categoría padre hay otra columna llamada *idCat_padre* que tiene el *id* del padre, y en el caso de que no tenga más categorías por encima su valor es -1.

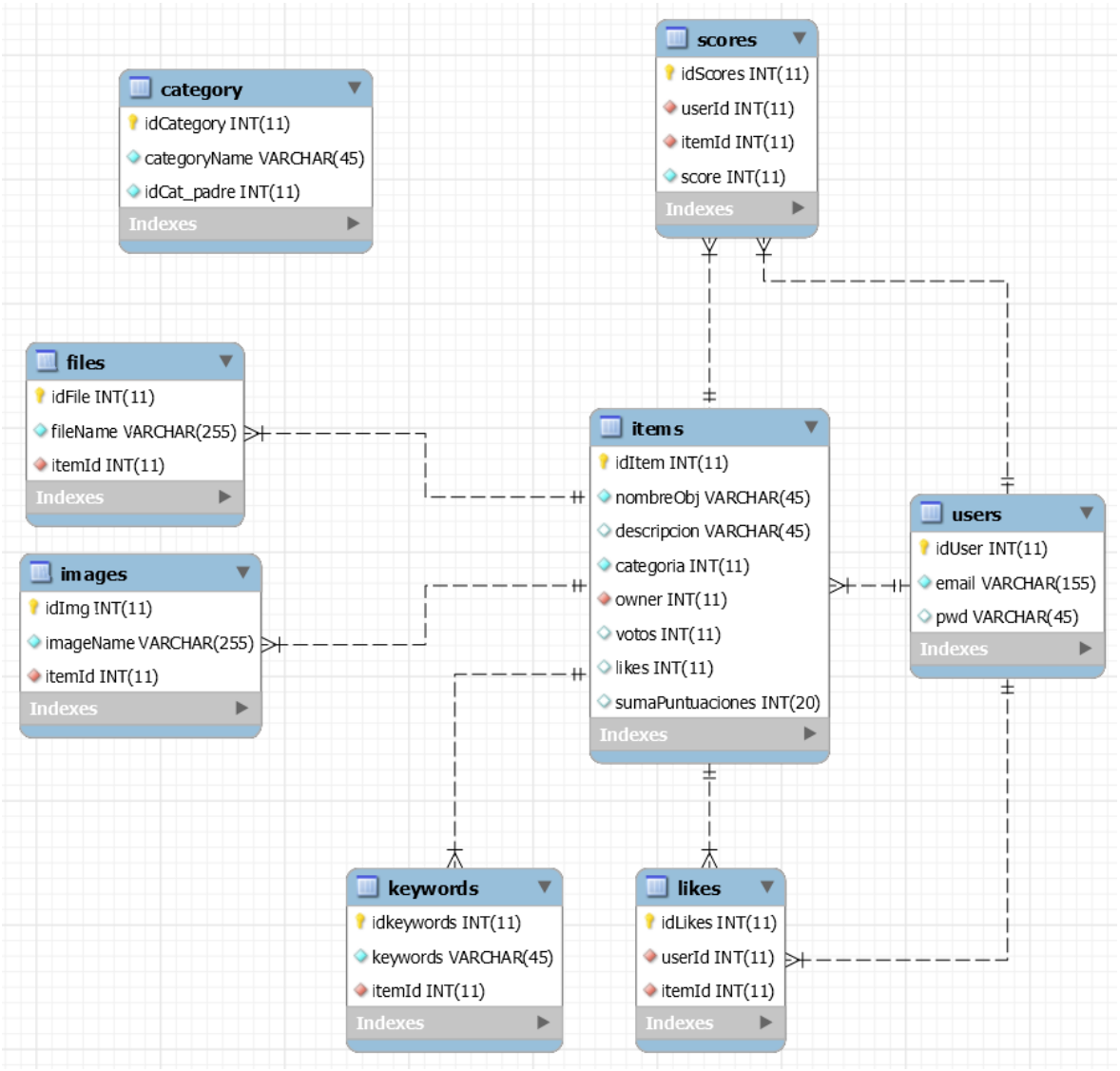


Figura 4.24: Diagrama de tablas de la base de datos.

Para la inserción de usuarios en la base de datos se ha utilizado un procedimiento almacenado. Es una funcionalidad que aporta MySQL, la cual permite tener un script almacenado en el servidor con un conjunto de comandos SQL. Existen múltiples situaciones donde un procedimiento almacenado es realmente útil:

1. Cuando diferentes aplicaciones cliente están escritas en distintos lenguajes o funcionan en diferentes plataformas, pero necesitan realizar la misma operación en la base de datos. Esta situación actualmente no fue la principal razón aunque de cara al futuro y posibles cambios puede ayudar a hacer más fácil la integración de clientes en otras plataformas o lenguajes.
2. Cuando la seguridad es importante. Entidades como bancos, usan procedimientos almacenados para todas las operaciones comunes. Esto proporciona un entorno seguro y consistente. Con el fin de tener un sistema de encriptación de contraseñas seguro se ha realizado la inserción de usuarios mediante un procedimiento almacenado.

Los procedimientos almacenados aportan mejoras de rendimiento ya que se necesita enviar menos información entre servidor y cliente. El servidor web tiene que estar capacitado para realizar numerosas peticiones, es decir, ser escalable.

El procedimiento almacenado `insertarUsuario` de la base de datos permite almacenar el usuario y la contraseña y que no sea visible en las tablas (ver Listado 4.34). El procedimiento recibe como parámetros el email y la contraseña. El algoritmo de encriptación que utiliza es *SHA1* combinado con la operación *UNHEX*, que interpreta cada par de dígitos hexadecimal como número y lo convierte a carácter. La ejecución del algoritmo nativo de MySQL para las contraseñas sería equivalente a: « SHA1(UNHEX(SHA1("password"))) ». Esto significa que realiza un segundo *SHA1* aplicado a la información binaria devuelta del primer *SHA1* y no a su representación hexadecimal.

El flujo que sigue el procedimiento almacenado `insertarUsuario` es el siguiente:

1. Inserta en usuarios el email recibido por parámetros, de manera que se autogenera un *id* de usuario.
2. Establece el valor del *query* para crear usuario concatenando «user» con el último *id* (que es el del email insertado) e identificado por la contraseña. Por ejemplo, si el *id* creado fuese 120 y la contraseña «pass» la *query* concatenada sería: «CREATE USER 'user120'@'%' IDENTIFIED BY 'pass'». Ejecuta la *query* de crearUsuario.
3. Se asigna permisos a los usuarios para hacer SELECT, INSERT, DELETE y UPDATE en el *schema*. Para ello se concatena la siguiente cadena: «grant SELECT,INSERT,DELETE,UPDATE on appdb.* to \'', @ultimoId, '\'@\'%\';» y se establece como valor de la *query* para asignar los permisos. Por último, se ejecuta la asignación de permisos.

Las salidas del procedimiento de inserción de usuarios son un mensaje de éxito y el *id* del nuevo usuario. El mensaje de éxito es «OK» en caso satisfactorio y diferente para cada tipo de error.

```
1  CREATE DEFINER=`root`@`localhost` PROCEDURE `insertarUsuario`(in pEmail
       varchar(155), in pPwd varchar(45), out pExito varchar(200), out
       pIdUsuario int)
2  BEGIN
3    DECLARE queryCrearUsuario VARCHAR(200);
4    DECLARE queryAsignarPermisos VARCHAR(200);
5
6    /** comprobacion y
7     manejo de errores **/
8
9    START TRANSACTION;
10     Insert into users (email) values (pEmail);
11     set @ultimoId=concat('user', LAST_INSERT_ID());
12
13     Set @queryCrearUsuario = CONCAT('create user \'' , @ultimoId , '\'@
       \'%\' identified by \'', pPwd , '\';');
14     PREPARE crearUsuario FROM @queryCrearUsuario;
15     EXECUTE crearUsuario;
```

```
17      Set @queryAsignarPermisos = CONCAT('grant SELECT,INSERT,DELETE,UPDATE
        on appdb.* to \'', @ultimoId, '\'@\'%\';');
18      PREPARE asignarPermisos FROM @queryAsignarPermisos;
19      EXECUTE asignarPermisos;

21      Set pExito='OK';
22      set pIdUsuario=LAST_INSERT_ID();
23    COMMIT;
24  END
```

Listado 4.34: Procedimiento almacenado `insertarUsuario`.

El mecanismo de persistencia utilizado para comunicarse con la base de datos es DAO (objeto de acceso a datos, del inglés *data access object*), que es un objeto que proporciona una interfaz abstracta. Los Objetos DAO son un patrón de diseño y son considerados una buena práctica. Las clases utilizadas para la persistencia son ocho para DAO y una para el manejo de la conexión y el pool de conexiones (ver Figura 4.25). Estas clases DAO contienen las operaciones SQL necesarias para obtener, actualizar, insertar o borrar datos de las tablas de la base de datos.

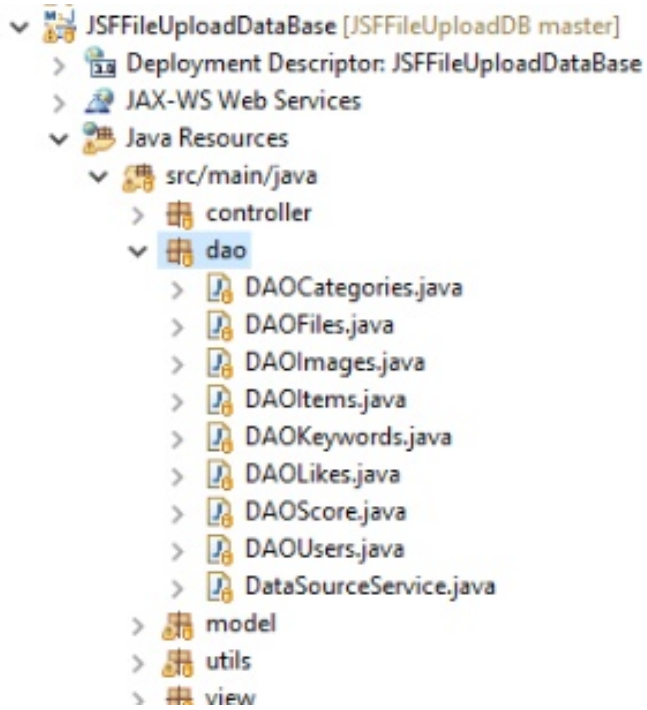


Figura 4.25: Listado de clases del SGBD.

Un *pool* de conexiones es una clase java que tiene abiertas varias conexiones a base de datos. Si se necesita una conexión a la base de datos, en lugar de crearla directamente con *DriverMana-*

ger.getConnection(), se la pide al *pool* utilizando el método `getDataSource().getConnection()`. Este *pool* toma una de las conexiones que ya existía, y la marca como utilizada para evitar que otra persona la utilice. Por lo que, otras llamadas usarían una conexión que esté libre y la marcarían como ocupada y así sucesivamente. La librería utilizada para la conexión y el *pool* es *commons-dbcp*. La clase *DataSourceService* es la encargada de gestionar la conexión, y su funcionamiento consiste en crear una instancia donde su método constructor define los parámetros necesarios, que son: *Username*, *Password* y *Url* principalmente (ver Listado 4.35).

```
1  public DataSourceService(){
2    if (null==basicDataSource){
3      basicDataSource = new BasicDataSource();
4      basicDataSource.setDriverClassName("com.mysql.jdbc.Driver");
5      basicDataSource.setUsername(user);
6      basicDataSource.setPassword(pass);
7      basicDataSource.setUrl("jdbc:mysql://127.0.0.1:3306/AppDB?
       noAccessToProcedureBodies=true");
8      basicDataSource.setMaxTotal(200);
9      basicDataSource.setMinIdle(50);
10     basicDataSource.setMaxIdle(100);
11   }
12 }
```

Listado 4.35: Método constructor de la clase *DataSourceService*.

Todas las clases DAO realizan la conexión de la misma manera, creando una instancia de la clase *DataSourceService* y con una llamada al método `getDataSource().getConnection()` obtiene el objeto *Connection* (ver Listado 4.36).

```
1 DataSourceService dss = new DataSourceService();
2 Connection bd = dss.getDataSource().getConnection();
```

Listado 4.36: Ejemplo de conexión con la base de datos.

4.1.4 Comunicación

La comunicación entre el cliente Android y el servidor consiste en darle al servidor la información que necesita para devolver el objeto 3D más adecuado a esa información. Para ello diferenciamos dos partes: la del cliente y la del servidor.

El módulo de comunicación del cliente Android recibe como entrada la URL del servidor, que es «“http://”+“ip”+“:8080/JSFFileUploadDataBase/rest/request/generarItemId”» donde *ip* es la dirección IP del servidor y el resto es la ruta determinada para hacer uso del servicio definido por el servidor. Además, toma como parámetro la salida generada por el módulo del procesamiento del lenguaje natural (ver Sección 4.1.1), es decir, la cadena de tuplas con las palabras clave y su factor de importancia. Este listado de tuplas es de tipo *String*, aunque luego el servidor para poder realizar

las operaciones lo transforme en un *HashMap*, de forma que se puede enviar como texto plano (ver Figura 4.26).

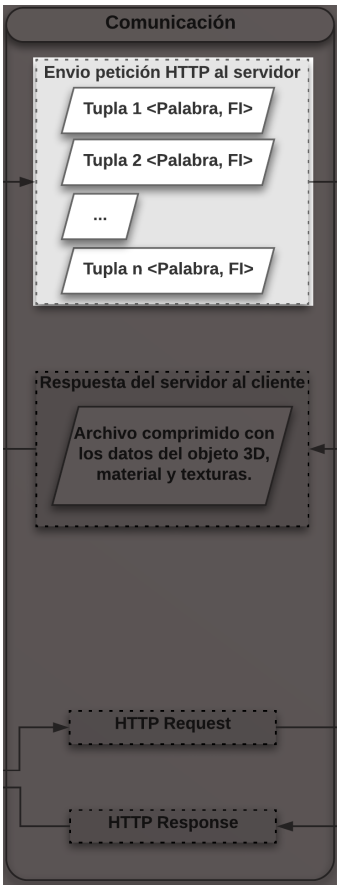


Figura 4.26: Comunicación Cliente a Servidor - Petición.

El método `PostMethod` es el encargado de gestionar la *request* y la *response* (ver Listado 10). Para establecer una conexión HTTP con el servidor y hacer la llamada al servicio hay que establecer primero la URL y los parámetros del contenido. Una vez realizada la inicialización, se escribe los datos (bytes) de la cadena a enviar como flujo de salida de datos en un *DataOutputStream*, se envía y cierra el *stream*. Si la respuesta recibida contiene el código HTTP 200 (HttpURLConnection.HTTP_OK es igual a 200) significa que ha recibido el objeto 3D correctamente.

Proceso 10 Método de comunicación Cliente Android.

Entrada: ServerURL: String con la URL para hacer petición al servicio.

Entrada: strResultado: String con las tuplas - salida del módulo PLN.

Salida: DataInputStream: respuesta como cadena de bytes.

1: Inicializar la conexión HTTP con la URL.
2: Establecer a VERDADERO que se va a enviar contenido en la petición.
3: Establecer el tipo de contenido enviado a "text/plain"
4: Establecer el tipo de contenido recibido a "application/octet-stream"
5: Establecer tipo de petición a POST
6: Establecer a VERDADERO que se espera contenido en la respuesta.
7: Realizar conexión
8: Escribir *strResultado* en un stream de salida
9: Enviar datos
10: Cerrar el stream
11: $respuesta \leftarrow get\_CodigoRespuesta$
12: **si** $respuesta = OK$ **entonces**
13: $\quad DataInputStream \leftarrow get\_Respuesta$
14: **fin si**
15: **devolver** *DataInputStream*

La parte del Servidor es la encargada de recibir esta petición y generar una respuesta, que puede ser que ha encontrado resultado y envía los archivos del objeto 3D o simplemente que no ha encontrado nada en el repositorio (ver Figura 4.27).

La clase *ClientRequestCom* es la encargada de manejar las peticiones. Para el desarrollo de un servicio JAX-RS RESTful que produce archivos se ha integrado la librería RESTeasy[3] de JBoss. Por lo que para hacer funcionar se ha definido una ruta URL en el web.xml, que es el descriptor del proyecto:

```
<servlet-mapping>
  <servlet-name>resteasy-servlet</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>

<context-param>
  <param-name>resteasy.servlet.mapping.prefix</param-name>
  <param-value>/rest</param-value>
</context-param>
```

Listado 4.37: URL del servicio en web.xml.

[3] http://docs.jboss.org/resteasy/docs/3.5.1.Final/userguide/html/index.html

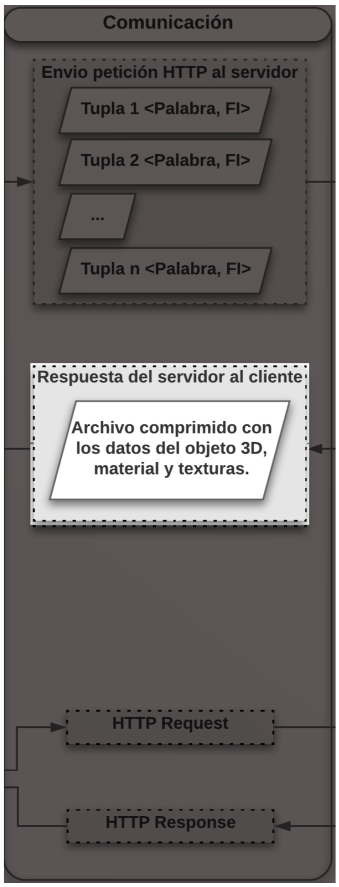


Figura 4.27: Comunicación Servidor a Cliente - Respuesta.

La ruta URL es completada en la clase *ClientRequestCom* y el método `getFile` por anotaciones "@Path(/request)" y "@Path(/generarItemId)". Por tanto, la ruta completa utilizada para hacer las peticiones por el cliente Android es: «/rest/request/generarItemId». El método `getFile` tiene dos anotaciones más además de *@Path*:

- "@POST" para definir el tipo de llamada.
- "@Produces(MediaType.APPLICATION_OCTET_STREAM)" para especificar el tipo de respuesta producida.

El método `getFile` obtiene el resultado con el itemId mediante el uso del Algoritmo de selección de objeto 3D. Con el itemId obtiene la ruta de los archivos para reproducirlo en Realidad Aumentada y los comprime todos en un mismo archivo ZIP que es enviado como respuesta al cliente Android.

Proceso 11 Método `getFile`.

Entrada: strResultado: String con las tuplas recibido del cliente Android.

Salida: Response: respuesta con el archivo comprimido.

1: Instanciar clase del algoritmo de selección de objetos 3d.
2: $resultado \leftarrow seleccion\_Item$
3: **si** $resultado = 0$ **entonces**
4: $\quad Response \leftarrow mensaje\_error$
5: **si no**
6: $\quad Response \leftarrow mensaje\_OK + archivoZIP$
7: **fin si**
8: **devolver** *Response*

La compresión y decompresión de archivos se realiza mediante la librería ***zt-zip***. Es una librería construida usando paquetes *java.util.zip.** para accesos basados en *stream*. El servidor comprime los archivos utilizando el método `pack` y como parámetros de entrada necesita el directorio de los archivos a comprimir y el nombre del archivo comprimido de salida. La parte del Cliente Android, una vez que ha convertido el *stream* recibido a archivo comprimido, mediante el método `unpack` descomprime el archivo en el directorio indicado con el nombre asignado por parámetros.

4.1.5 Patrones de diseño

Los analistas y programadores han ido desarrollando a diario habilidades que les permiten solucionar problemas muy comunes en el desarrollo del *software*. Para cada uno de estos problemas se presentan diferentes formas de resolverlos. Estas soluciones se han ido documentando a lo largo de los años con el fin de que podamos reutilizarlas e incluso mejorarlas y adaptarlas a los cambios que se van aconteciendo en el sector de la informática.

Estas soluciones, conocidas como patrones de diseño, tienen como principal objetivo proponer soluciones a problemas de diseño que se repiten. Por lo tanto, los patrones de diseño son soluciones exitosas a problemas comunes.

Los patrones de diseño que se han utilizado para este TFG son:

- Singleton
- Rendezvous
- Data Access Object (DAO)
- Modelo-Vista-Controlador (MVC)

Singleton

El propósito de este patrón es asegurar que una clase tenga una única instancia y proporcionar un punto de acceso global a la misma, es decir, restringe la creación de más instancias del mismo tipo. Uno de los beneficios más importantes de este patrón es que es *resource friendly*, es decir no gasta

memoria en un nuevo objeto cuando realmente no necesitas uno nuevo y se evita la sobrecarga de instanciación. Esta es la principal razón por la que se ha utilizado el patrón Singleton ya que permite la gestión de repositorios host de servicios en arquitecturas orientadas a servicios. El siguiente diagrama representa el proceso de manejo de la solicitud del cliente utilizando el modo de instancia de Singleton (ver Figura 4.28).

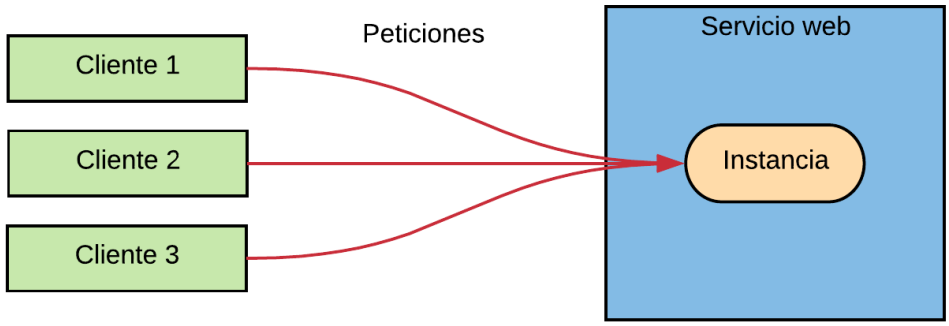


Figura 4.28: Manejo de *request* del cliente.

Este patrón de diseño se ha utilizado para la parte de la comunicación entre la aplicación móvil y el servidor, en concreto en el servicio web REST. Mediante la clase *MessageApplication* extendida de `javax.ws.rs.core.Application`, donde se define un *Set* que contiene una colección de objetos de la clase *ClientRequestCom*.

```
1  public class MessageApplication extends Application{
2    private Set<Object> singletons = new HashSet<Object>();
3
4    public MessageApplication(){
5      singletons.add(new ClientRequestCom());
6    }
7    @Override
8    public Set<Object> getSingletons(){
9      return singletons;
10   }
11 }
```

Listado 4.38: Clase *MessageApplication*

En el Listado 4.39 se muestra el parámetro que define el servlet en el descriptor *web.xml*, de esta manera al invocar la URL del servicio web el servidor lo reconoce y puede añadir una instancia al *Set*:

```
<servlet>
  <servlet-name>resteasy-servlet</servlet-name>
  <servlet-class>
    org.jboss.resteasy.plugins.server.servlet.HttpServletDispatcher
  </servlet-class>
  <init-param>
    <param-name>javax.ws.rs.Application</param-name>
    <param-value>utils.MessageApplication</param-value>
  </init-param>
</servlet>
```

Listado 4.39: Definición del *servlet* en el descriptor

Rendezvous

El patrón Rendezvous es una forma simplificada del patrón de llamada protegida y se utiliza para sincronizar un conjunto de hilos o permitir el intercambio de datos coordinado entre dos procesos concurrentes, el solicitante y el llamado (ver Figura 4.29).

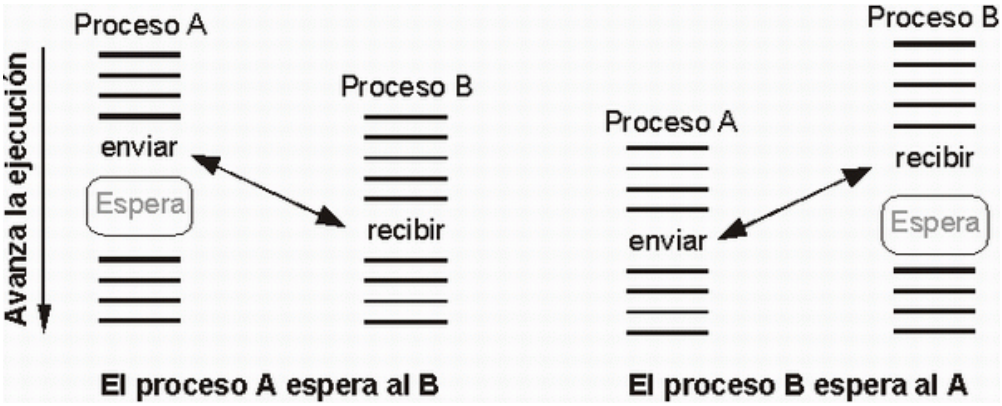


Figura 4.29: Modelo de sincronización Rendezvous.

En este proyecto aparece como un servicio a la espera de ser solicitado, se desbloquea y el solicitando se bloquea a la espera de la respuesta del servicio. Cuando el servicio se completa, el servidor se vuelve a bloquear y el solicitante continúa su trabajo.

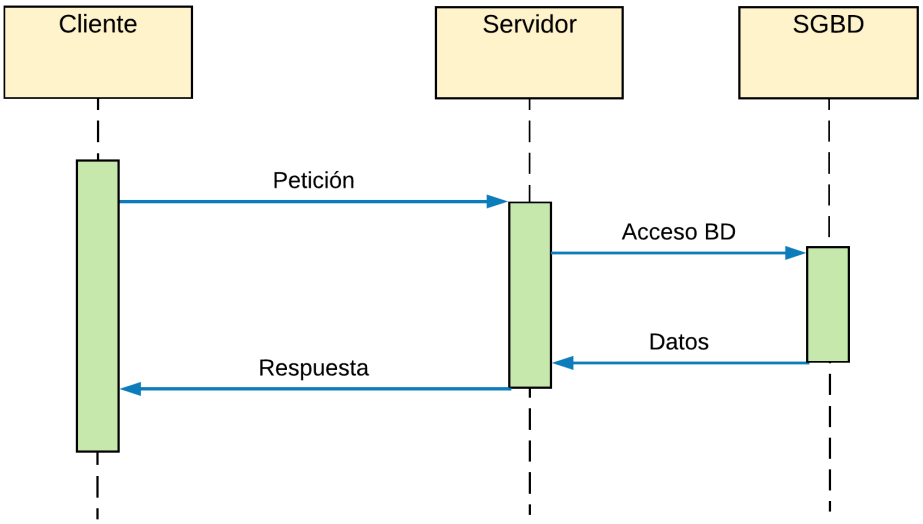


Figura 4.30: Sincronización Cliente-Servidor.

Data Access Object (DAO)

El patrón DAO (***D**ata **A**ccess **O**bject*) es una solución sencilla para que la aplicación sea lo más independiente posible de la base de datos. La aplicación consigue los datos, haciendo llamadas a métodos de clases DAO, por lo que no existen parámetros tipo *Connection* por ejemplo en las clases de Modelo. Las principales ventajas de utilizar este patrón que afectan a este proyecto son:

- Las llamadas para recibir objetos son muy parecidas siempre.
- Una vez creada una clase DAO que contenga operaciones de crear, lectura, actualización y borrar, el diseño general puede repetirse para otras clase DAO.

Los componentes de acceso a datos encapsulan la tecnología empleada para acceder a la capa de datos, separando completamente la lógica necesaria para acceder a sistema gestor de bases de datos.

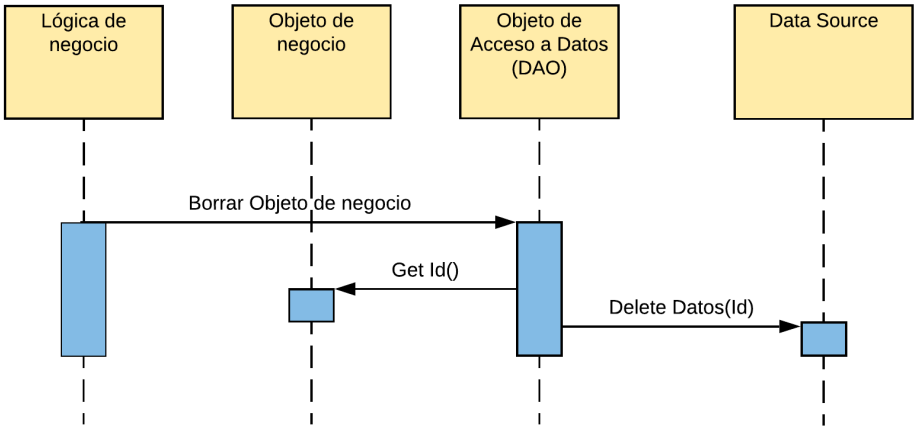


Figura 4.31: Diagrama de interacción para realizar una operación *delete*.

La clase *DataSourceService* es una implementación de fuente de datos para la base de datos relacional. En esta clase está contenida la configuración de la conexión a la base de datos desde el servidor. Las clases DAO, son componentes de acceso a datos que instanciando la clase *DataSourceService* realizan una conexión, de esta manera las clases de la capa de negocio no necesitan conocer como se realizan las operaciones con la base de datos.

Modelo-Vista-Controlador (MVC)

Para el diseño de aplicaciones con interfaces, como la del Cliente Web, se recomienda utilizar el patrón Modelo-Vista-Controlador. JSF utiliza el patrón de diseño MVC para gestionar las aplicaciones web. MVC es una solución efectiva al problema de arquitectura que plantea la necesidad de separar la parte de presentación o vistas y la parte del manejo de datos o modelo.

La tres partes que definen la arquitectura del patrón MVC son:

- **Modelo:** contiene la lógica de negocio que procesa datos. En JSF se implementa las clases *managed bean*, también conocidos como *backing beans*.
- **Vista:** esta parte la forman elementos para la entrada y salida de datos de usuario formados por componentes de la tecnología *Facelets* de JSF y con el lenguaje de expresiones EL. La vista puede estar formada por elementos de JSF como páginas *xml Facelets*, clases que definen propiedades de la vista, clases para extender funcionalidades de componentes, validadores, conversores, *listeners*, etc.
- **Controlador:** viene implementado por JSF a través de la clase *FacesServlet*. Se encarga de manejar cualquier acción del usuario sobre las vistas y de mostrar la información correspondiente.

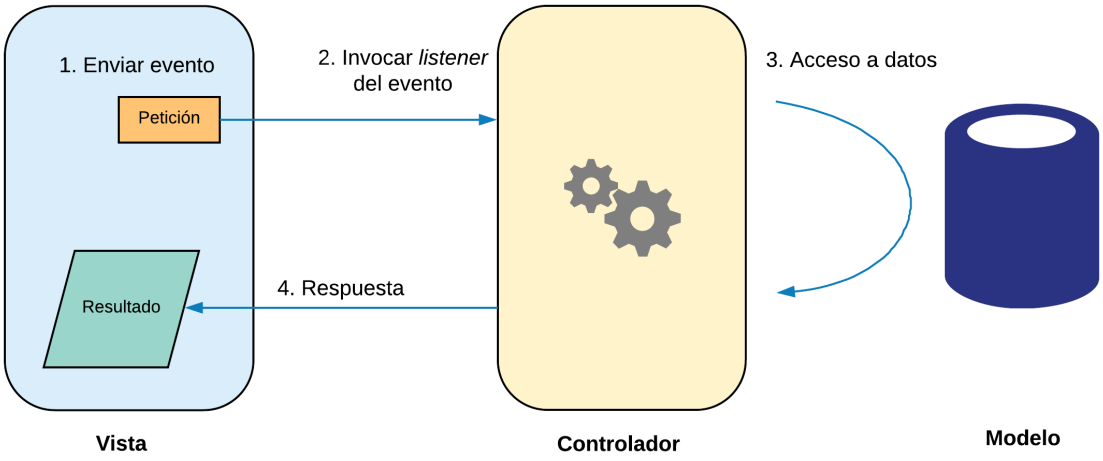


Figura 4.32: Modelo-Vista-Controlador.

4.2 Desarrollo de la Metodología Ágil

Scrum adopta una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto. Una de las ventajas de esta estrategia es la paralelización de las tareas, es decir, que varios recursos trabajen simultáneamente de forma que exista un solapamiento de las fases de desarrollo. Como en este proyecto el único desarrollador es el alumno, no se ha trabajo de forma paralela.

4.2.1 Roles en el proyecto

En este proyecto solo han participado dos personas, el alumno y el director del proyecto. El *product owner* es el director del proyecto, ya que tiene conocimiento suficiente sobre el producto y las funciones que debe realizar. Por otra parte, el alumno adoptará los roles de *Scrum Master* y *Development Team*.

A continuación, en esta sección, se muestra el *Product Backlog*, en el se exponen las historias de usuario cuya implementación es importante y que describen una funcionalidad que el sistema debe tener. Estas historias de usuario se asignan en iteraciones o *Sprint*.

4.2.2 Product Backlog

El *Product Backlog* contiene la lista de Historias de Usuario que están repartidas en *Sprints*. La asignación de Historias de Usuario en cada iteración se ha realizado siguiendo un orden, es decir, todos los *Sprints* generan un resultado que es necesario para poder completar el siguiente. Los resultados de las pruebas mencionadas en cada Historia de Usuario se encuentran en el capítulo de Resultados.

SPRINT	HISTORIA DE USUARIO
0	HdU 1: Elección y configuración de los entornos de desarrollo a utilizar.
	HdU 2: Análisis de frameworks para los diferentes módulos.
	HdU 3: Configuración del SGBD y diseño de la base de datos.
1	HdU 4: Integración de Sesiones HTTP.
	HdU 5: Registro de usuarios.
	HdU 6: Login de usuarios.
2	HdU 7: Insertar modelos 3D en el repositorio.
	HdU 8: Visualización de los perfiles de objetos 3D del repositorio.
	HdU 9: Gestionar los objetos del usuario en «Mis Objetos».
3	HdU 10: Obtención de la imagen con la cámara.
	HdU 11: Reconocimiento óptico de caracteres.
	HdU 12: Proceso de análisis del texto.
4	HdU 13: Desarrollo del algoritmo de selección de objeto 3D.
5	HdU 14: Preparación de E/S para la comunicación.
	HdU 15: Integración de la comunicación Cliente-Servidor.
6	HdU 16: Desarrollo del módulo integrado de Realidad Aumentada.

Cuadro 4.1: Product Backlog

4.2.3 Planificación temporal del proyecto

Una vez definidas las Historias de Usuario y la temporalidad de cada *Sprint*, se estima la duración del proyecto. Para ello se ha utilizado un diagrama de Gantt.

La fecha de inicio del proyecto es el 1 de noviembre y la fecha de finalización estimada es el 6 de junio. La jornada laboral es de 8 horas por día de lunes a viernes, siendo días no laborales los fines de semana.

Se ha definido como único recurso al alumno, por lo que al haber solo un desarrollador no pueden paralelizarse las tareas inevitablemente.

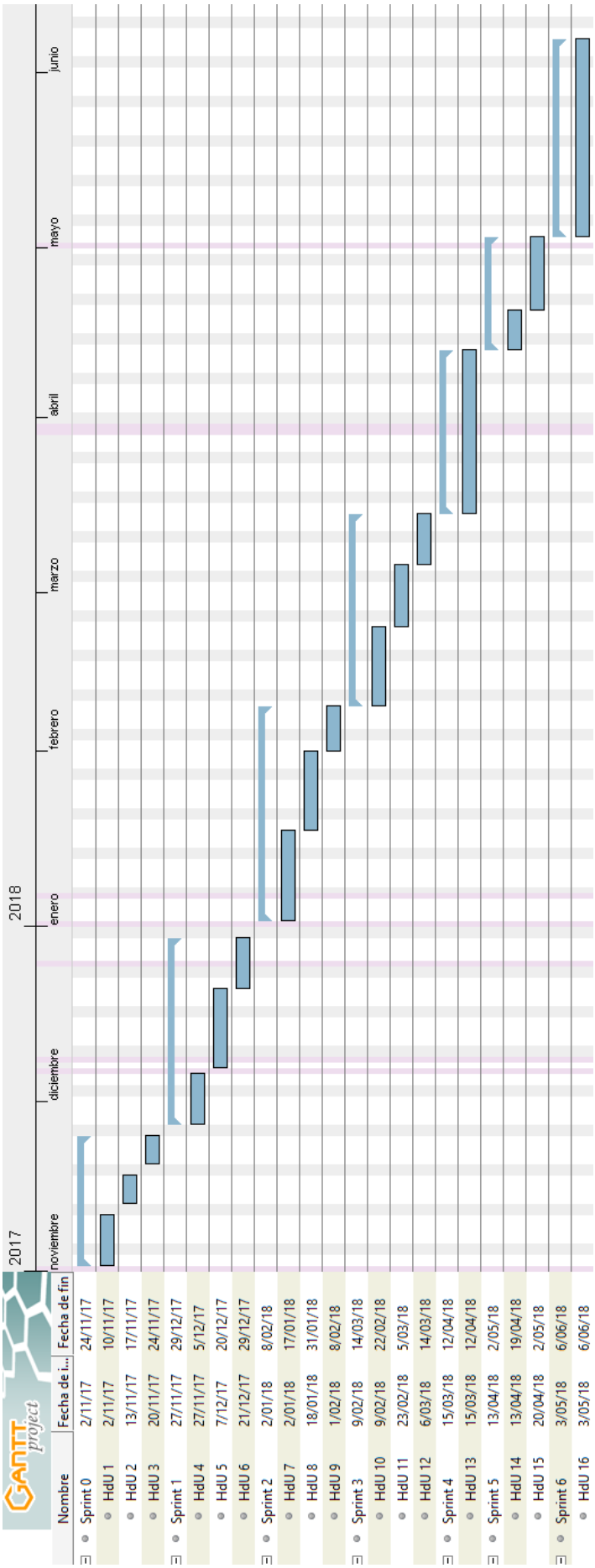


Figura 4.33: Diagrama de Gantt del TFG.

Las Historias de Usuario de cada *Sprint* que han sido planificadas se explican en detalle a continuación.

4.2.4 Marco Tecnológico

Para hacer posible el desarrollo del sistema han sido necesarios múltiples recursos, tanto *hardware* como *software*. En esta sección se van a listar cada uno de ellos.

Medios Hardware

Los medios *hardware* necesitados para el desarrollo de todas las partes de este sistema son:

- Ordenador portátil MSI PE70 7RD-644XES:
  - Procesador Intel Core i7-7700HQ
  - 8GB de RAM
  - GPU GTX1050
- Dispositivo móvil Android: Samsung Galaxy S7 Edge:
  - Procesador Exynos 8890
  - 4GB de RAM
  - GPU Mali-T880 MP12

Medios Software

Los medios *software* utilizados para el diseño y desarrollo del sistema son:

- Eclipse: IDE para la parte del cliente web, servidor y comunicación.
- Android Studio: IDE para el cliente Android y comunicación.
- ARC (Advanced REST client): pruebas de comunicación.
- MySQL WorkBench: gestión de la base de datos.
- Ganttproject: planificación temporal.
- TeXstudio: IDE de LaTeX para el desarrollo del documento.
- Lucidchart: aplicación web para diseño de esquemas y diagramas.
- Trello: aplicación web de tableros para la metodología.
- Windows 10: sistema operativo utilizado.
- ViroCore: librería para la Realidad Aumentada.
- Google Cloud Vision API: librería que permite extraer texto mediante OCR.
- Google Cloud Natural Language API: librería para análisis de textos.
- RESTeasy: *framework* para servicios web RESTful.
- JavaServer Faces (JSF): *framework* para el desarrollo web.

4.2.5 Coste estimado del proyecto

Se ha realizado un estudio económico del proyecto que abarca los ocho meses teniendo en cuenta los siguientes conceptos:

- Sueldo de trabajadores: en este proyecto sólo hay dos sueldos que pagar, el del desarrollador y el del *product owner*. Las horas trabajadas por el desarrollador equivalen a las duración del proyecto y las del *product owner* equivalen a 2 horas por semana, una para la reunión y otra para la revisión.
- Los medios *hardware* que se han tenido en cuenta son un portátil y un *smartphone*.
- El *software* utilizado no supone coste alguno ya que con las licencias académicas o con las versiones gratuitas es suficiente para el correcto desarrollo del sistema.
- Gastos de mantenimiento (electricidad, agua e Internet).

Teniendo en cuenta estos factores la tabla de costes es la siguiente:

Concepto	**Coste**
Sueldo trabajadores	1184 horas * 15€/hora = 17760€ 64 horas * 20€/hora = 1280€
Medios *Hardware*	900€+500€=1400€
Medios *Software*	0€
Electricidad (8 meses)	20€/mes = 160€
Agua (8 meses)	15€/mes = 120€
Internet (8 meses)	55€/mes = 440€
TOTAL	**21160€**

Cuadro 4.2: Tabla de costes estimados del proyecto.

4.2.6 Sprint 0

Este *Sprint* inicial es una fase de aprendizaje y configuración del proyecto. Por lo tanto, se hace un análisis del estado del arte, prestando especial atención a las librerías de reconocimiento óptico de caracteres (OCR), procesamiento del lenguaje natural (PLN) y de realidad aumentada integrada en dispositivos Android.

Se han incluido las Historias de Usuario a la vez, ya que en este incremento no existen todas las fases del ciclo como son análisis, diseño, implementación y pruebas. Es un *Sprint* de inicio de proyecto con el que establecer las bases necesarias para su desarrollo.

HISTORIA DE USUARIO 1	
Número: 1	**Usuario:** Scrum Master
Nombre Historia: Elección y configuración de los entornos de desarrollo a utilizar	
Prioridad en negocio: Medio	**Riesgo en desarrollo:** Alto
Esfuerzo: 56 horas	**Sprint asignado:** 0
Programador responsable: Alberto González Sánchez	
Descripción: Conocer y configurar los entornos de desarrollo necesarios.	
Validación: Tener un entorno de trabajo adecuado para el desarrollo del sistema.	

Cuadro 4.3: Historia de Usuario 1.

HISTORIA DE USUARIO 2	
Número: 2	**Usuario:** Scrum Master
Nombre Historia: Análisis de *frameworks* para los diferentes módulos.	
Prioridad en negocio: Medio	**Riesgo en desarrollo:** Alto
Esfuerzo: 40 horas	**Sprint asignado:** 0
Programador responsable: Alberto González Sánchez	
Descripción: Estudiar librerías necesarias para el desarrollo.	
Validación: Conocer e integrar las librerías necesarias para cada módulo.	

Cuadro 4.4: Historia de Usuario 2.

HISTORIA DE USUARIO 3	
Número: 3	**Usuario:** Scrum Master
Nombre Historia: Config. SGBD y creación de tablas en la base de datos.	
Prioridad en negocio: Medio	**Riesgo en desarrollo:** Alto
Esfuerzo: 40 horas	**Sprint asignado:** 0
Programador responsable: Alberto González Sánchez	
Descripción: Configurar un SGBD, las tablas y sus relaciones.	
Validación: La persistencia del sistema que permita el almacenamiento de datos.	

Cuadro 4.5: Historia de Usuario 3.

Estas Historias de Usuario hacen referencia a tareas para la elección y configuración de los entorno de trabajo. Se ha elegido Eclipse IDE para el diseño y desarrollo de la aplicación web y el servidor, y para la aplicación móvil se ha trabajado con Android Studio. En ambos entornos de desarrollo se trabaja con Java.

El Sistema Gestor de Bases de Datos elegido es MySQL, que permite trabajar con bases de datos relacionales y crear procedimientos almacenados para realizar operaciones como la encriptación de la contraseña del usuario. Además, se ha diseñado un sistema de tablas que cumple con las necesidades del repositorio y del servidor, que son los que acceden a la base de datos directamente.

Para la gestión del proyecto se ha configurado un repositorio para el versionado del código entre los integrantes del grupo de trabajo en BitBucket.

4.2.7 Sprint 1

En este primer *Sprint* se realizan las tareas relativas a la gestión de usuarios, es decir, la creación de sesiones HTTP, registro de usuarios y autentificación en el sitio web.

Las historias de usuario definidas para este Sprint son:

Historia de Usuario 4

Las sesiones de usuario son imprescindibles para poder realizar la autentificación de usuarios. Mediante la sesión el usuario puede navegar por las diferentes vistas con los parámetros almacenados de este usuario.

HISTORIA DE USUARIO 4	
Número: 4	**Usuario:** Cliente
Nombre Historia: Integración de Sesiones HTTP	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Esfuerzo: 56 horas	**Sprint asignado:** 1
Programador responsable: Alberto González Sánchez	
Descripción: Poder navegar entre las diferentes opciones del sitio.	
Validación: Integración de sesiones para mantener a un usuario autenticado.	

Cuadro 4.6: Historia de Usuario 4.

Esta característica permite tener acceso a funcionalidades adicionales como votar o guardar los perfiles de objetos 3D que más le gusten al usuario. Permiten ver y manipular información sobre una sesión, como el identificador de sesión, la hora de creación y la última hora de acceso, además de poder enlazar objetos a las sesiones lo que permite que la información del usuario persista en múltiples conexiones.

Se realizan pruebas para comprobar que las siguientes funcionalidades cumplen con lo esperado:

- Se crea adecuadamente la sesión en el contexto JSF cuando el usuario realiza el *login* o registro.
- Se destruye la sesión cuando el usuario hace *logout* o alcanza el tiempo de inactividad.

Historia de Usuario 5

El registro de usuarios es imprescindible para realizar aportaciones al repositorio, tanto para poder compartir objetos 3D como para poder valorar los perfiles de objetos 3D de otros o guardar en favoritos.

HISTORIA DE USUARIO 5	
Número: 5	**Usuario:** Cliente
Nombre Historia: Registro de usuarios	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Esfuerzo: 72 horas	**Sprint asignado:** 1
Programador responsable: Alberto González Sánchez	
Descripción: Poder crear cuentas de usuario.	
Validación: Inserción de usuarios en el sistema.	

Cuadro 4.7: Historia de Usuario 5.

El registro de usuarios tiene una vista propia donde se comprueba que los datos introducidos son correctos, es decir, no existe el email y las contraseñas coinciden. Además, existe cierta seguridad almacenando la contraseña ya que se utiliza el algoritmo nativo de MySQL que tiene una encriptación SHA1 combinada con operaciones UNHEX, que interpreta cada par de dígitos hexadecimal como número y lo convierte a carácter.

Se realizan pruebas para comprobar que las siguientes funcionalidades cumplen con lo esperado:

- No permite completar el registro si la contraseña y la confirmación de contraseña no coinciden.
- No permite registrarse si el campo *email* ya existe.
- La contraseña no se puede visualizar en las tablas de la base de datos y está encriptada correctamente.

Historia de Usuario 6

Para que el usuario pueda acceder a su cuenta, votar perfiles de objetos de otros usuarios, darle a «Me gusta», entre otras funcionalidades que requieren sesión activa es necesario un sistema de autentificación.

HISTORIA DE USUARIO 6	
Número: 6	**Usuario:** Cliente
Nombre Historia: Login de usuarios	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Esfuerzo: 48 horas	**Sprint asignado:** 1
Programador responsable: Alberto González Sánchez	
Descripción: Poder acceder a las cuentas de usuario.	
Validación: Sistema de autentificación de usuarios.	

Cuadro 4.8: Historia de Usuario 6.

Las comprobaciones realizadas para comprobar el correcto funcionamiento son:

- Permite autentificarse únicamente si los credenciales son correctos.

4.2.8 Sprint 2

En esta iteración del proyecto se van a diseñar e implementar las funciones del repositorio. De manera que el usuario pueda insertar modelos 3D con un perfil que lo describe, visualizar los perfiles de los objetos 3D del repositorio y poder gestionar los objetos del usuario.

Las historias de usuario definidas para este Sprint son:

Historia de Usuario 7

Permite al usuario compartir los archivos de un objetos 3D junto a un perfil descriptivo que permite tener una cierta organización para que el repositorio pueda ser amplio, y que tanto el usuario como el algoritmo de selección de objeto 3D tengan información de cada objeto 3D compartido.

HISTORIA DE USUARIO 7	
Número: 7	**Usuario:** Cliente
Nombre Historia: Insertar modelos 3D en el repositorio	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Esfuerzo: 96 horas	**Sprint asignado:** 2
Programador responsable: Alberto González Sánchez	
Descripción: Permitir al usuario subir objetos 3D al repositorio.	
Validación: Compartición de objetos 3D y su perfil descriptivo.	

Cuadro 4.9: Historia de Usuario 7.

Se realizan pruebas para comprobar si las siguientes funcionalidades cumplen con lo esperado:

- Almacena los archivos del objeto 3D y de las imágenes previas en el directorio adecuado.

- Valida si se han rellenado los campos obligatorios.
- Almacena los datos descriptivos del formulario en el *item* correcto.

Historia de Usuario 8

La vista principal de la aplicación web permite visualizar todos los perfiles de objetos 3D independientemente de si existe sesión activa o no. La sesión solo afecta para poder votar o darle a «Me gusta» sobre los perfiles de objetos 3D mostrados. Tiene una función que permite ver una ficha de cada perfil en un *modal*.

HISTORIA DE USUARIO 8	
Número: 8	**Usuario:** Cliente
Nombre Historia: Visualización de perfiles de objetos 3D del repositorio	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Esfuerzo: 80 horas	**Sprint asignado:** 2
Programador responsable: Alberto González Sánchez	
Descripción: Poder visualizar los perfiles de los objetos del repositorio.	
Validación: Se muestra y permite visualizar los perfiles de los objetos 3D.	

Cuadro 4.10: Historia de Usuario 8.

Las comprobaciones corresponden con pruebas de visibilidad, es decir, si se visualizan los datos correctos y si la ventana *modal* para ver los detalles carga las imágenes de cada *item*.

Historia de Usuario 9

Cada usuario registrado tiene una vista donde puede ver sus objetos subidos. En esta vista el usuario, además de visualizar todas las características de cada perfil, también puede editar los datos que contiene y eliminar por completo el perfil.

HISTORIA DE USUARIO 9	
Número: 9	**Usuario:** Cliente
Nombre Historia: Gestión de los objetos del usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Esfuerzo: 48 horas	**Sprint asignado:** 2
Programador responsable: Alberto González Sánchez	
Descripción: Gestión de los objetos subidos por el usuario.	
Validación: Permite ver, editar o eliminar objetos subidos por el usuario.	

Cuadro 4.11: Historia de Usuario 9.

Se realizan pruebas para comprobar que:

- Se visualizan correctamente todos los parámetros.
- La edición y eliminación de los objetos se realiza adecuadamente.

4.2.9 Sprint 3

En este *Sprint* se comienza con el desarrollo de la aplicación móvil para Android. Se definen tres tareas para desarrollar las siguientes parte del Cliente Android:

- Cámara con las funcionalidades adicionales como el rectángulo para encuadrar el texto deseado.
- Módulo de reconocimiento óptico de caracteres.
- Módulo de procesamiento del lenguaje natural.

Historia de Usuario 10

La entrada del Cliente Android es una fotografía tomada por el usuario a un texto determinado para que sea reconocido. La calidad de esta fotografía es determinante en el porcentaje de acierto, para ello la cámara tiene funcionalidades adicionales internas para la focalización de la imagen y el encuadre del texto.

HISTORIA DE USUARIO 10	
Número: 10	**Usuario:** Cliente
Nombre Historia: Obtención de la imagen con la cámara	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Esfuerzo: 80 horas	**Sprint asignado:** 3
Programador responsable: Alberto González Sánchez	
Descripción: Tomar una foto para reconocer texto.	
Validación: Cámara con encuadre para la toma de foto al texto deseado.	

Cuadro 4.12: Historia de Usuario 10.

Las pruebas realizadas con la cámara son:

- La comprobación del correcto funcionamiento del encuadre y del *focus*.
- Se comprueba si la imagen generada se almacena en el directorio que debe.
- Se comprueba si la imagen generada tiene las dimensiones adecuadas.

Historia de Usuario 11

En esta Historia de Usuario se desarrolla el módulo de OCR que recibe como entrada la imagen generada por la cámara. Mediante la librería de Google se analiza la imagen y se extrae el texto en formato digital para que el módulo del análisis del texto pueda reconocer las palabras clave.

HISTORIA DE USUARIO 11	
Número: 11	**Usuario:** Cliente
Nombre Historia: Reconocimiento óptico de caracteres	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Esfuerzo: 56 horas	**Sprint asignado:** 3
Programador responsable: Alberto González Sánchez	
Descripción: Obtener el texto en formato digital.	
Validación: Implementación del módulo OCR para digitalizar texto.	

Cuadro 4.13: Historia de Usuario 11.

Las pruebas realizadas tienen como objetivo, además de comprobar que funciona correctamente, generar una estadísticas aplicando posibles factores que afecten al resultado como el ángulo de realización de la foto, la iluminación, distancia, etc.

Historia de Usuario 12

El objetivo es desarrollar el módulo encargado del análisis del texto mediante un algoritmo de procesamiento del lenguaje natural de Google. La salida generada por este módulo es el listado de palabras junto a su factor de importancia, que indica la relevancia que tiene esa palabra en el texto analizado.

HISTORIA DE USUARIO 12	
Número: 12	**Usuario:** Cliente
Nombre Historia: Análisis de texto mediante PLN	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Esfuerzo: 56 horas	**Sprint asignado:** 3
Programador responsable: Alberto González Sánchez	
Descripción: Extraer las palabras importantes del texto.	
Validación: Obtención de listado de palabras clave y su relevancia en el texto.	

Cuadro 4.14: Historia de Usuario 12.

Para comprobar que el uso del algoritmo realiza la tarea para la cual fue diseñado, debe ejecutarse a mano. Para esto deben utilizarse datos representativos y anotarse los valores que toman las variables.

4.2.10 Sprint 4

El Cliente Android sólo puede recibir un objeto 3D para representar en Realidad Aumentada por lo tanto se ha desarrollado un algoritmo que permite determinar qué objeto 3D es el más adecuado. Este

algoritmo recibe el listado generado por el módulo de procesamiento del lenguaje natural del Cliente Android. Debido al volumen de trabajo que se emplea al desarrollo de este algoritmo se ha dedicado todo el tiempo del *Sprint* 4 a esta Historia de Usuario.

Historia de Usuario 13

La entrada de este algoritmo es un listado de tuplas compuestas por las palabras clave y su factor de importancia en el texto analizado, para obtener el resultado más adecuado se tienen en cuenta sobretodo el número de palabras coincidentes en los *items* existentes en el repositorio y qué combinación de palabras debido a que cada una tiene un factor de importancia diferente. En el caso de que hubiera más de un resultado habría que realizar un desempate teniendo en cuenta otro tipo de valores de cada *item* como los *likes*, votos y nota media. El algoritmo y sus subprocesos están explicados en detalle en la Sección 4.1.3.

HISTORIA DE USUARIO 13	
Número: 13	**Usuario:** Cliente
Nombre Historia: Algoritmo de Selección de Objeto 3D	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Esfuerzo: 152 horas	**Sprint asignado:** 4
Programador responsable: Alberto González Sánchez	
Descripción: El servidor elije que objeto es el más adecuado.	
Validación: Obtención del resultado más adecuado mediante un Algoritmo.	

Cuadro 4.15: Historia de Usuario 13.

Para comprobar que tiene el comportamiento esperado se van a probar las siguientes combinaciones y su combinaciones derivadas:

- Sólo hay un *item* con coincidencias en todas las palabras.
- Hay más de un *item* con coincidencias en todas las palabras.
- Hay más de un *item* con coincidencias pero no en todas las palabras.
- No se encuentra ninguna coincidencia.

4.2.11 Sprint 5

En este *Sprint* se crea un sistema de comunicación entre el servidor y la aplicación móvil permitiendo así enviar el listado de tuplas al servidor desde el cliente y que la respuesta generada por el Algoritmo de Selección de objeto 3D pueda ser enviada al cliente.

Historia de Usuario 14

Para poder realizar la comunicación hay que tener en cuenta dos partes: la petición del cliente y la respuesta del servidor. La petición se envía como texto plano, por lo que es necesario convertir todas

las tuplas generadas en un *HashMap* a un *String*. Por otra parte, la respuesta del servidor se envía como un único archivo comprimido en formato ZIP que será descomprimida en el cliente.

HISTORIA DE USUARIO 14	
Número: 14	**Usuario:** Cliente
Nombre Historia: Preparación de E/S para la comunicación	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Esfuerzo: 40 horas	**Sprint asignado:** 5
Programador responsable: Alberto González Sánchez	
Descripción: Enviar una cadena y recibir un único archivo.	
Validación: Transformar resultado a cadena para enviar y compresión de archivos.	

Cuadro 4.16: Historia de Usuario 14.

Historia de Usuario 15

La comunicación se realiza mediante una arquitectura REST. Su función en darle al servidor la información que necesita para devolver el objeto 3D más adecuado a esa información. Para ello diferenciamos dos partes: la del cliente y la del servidor.

HISTORIA DE USUARIO 15	
Número: 15	**Usuario:** Cliente
Nombre Historia: Integración de la comunicación Cliente-Servidor	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Esfuerzo: 64 horas	**Sprint asignado:** 5
Programador responsable: Alberto González Sánchez	
Descripción: El cliente debe enviar el listado de palabras y recibir el objeto 3D.	
Validación: Comunicación Cliente-Servidor mediante RESTful WS.	

Cuadro 4.17: Historia de Usuario 15.

Las pruebas realizadas deben corroborar que se envía correctamente la cadena con las tuplas desde el cliente al servidor y que el archivo comprimido por el servidor llega al cliente.

4.2.12 Sprint 6

Como ocurre en el *Sprint* 4, por la cantidad de horas y complejidad que supone integrar el módulo de Realidad Aumentada únicamente existe la Historia de Usuario dedicada a ello.

Historia de Usuario 16

En esta tarea, se desarrolla un módulo de Realidad Aumentada integrado para Android mediante la librería ViroCore. Este módulo recibe los archivos del objeto 3D ya descomprimidos en un directorio y es capaz de reproducirlo sobre una superficie real.

HISTORIA DE USUARIO 16	
Número: 16	**Usuario:** Cliente
Nombre Historia: Módulo integrado de Realidad Aumentada para Android	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Esfuerzo: 200 horas	**Sprint asignado:** 6
Programador responsable: Alberto González Sánchez	
Descripción: Poder representar el objeto 3D sobre una superficie con RA.	
Validación: Integración de librería de RA para Android.	

Cuadro 4.18: Historia de Usuario 16.

Las pruebas realizadas sobre este módulo consiste en:

- Comprobación del funcionamiento con los diferentes formatos soportados.
- Comportamiento de la detección de superficie en diferentes situaciones de luminosidad, posición, etc.
- Comportamiento del modelo 3D cuando el dispositivo que lo reproduce se desplaza.
- Comportamiento del modelo 3D cuando el usuario lo mueve o rota con los gestos de los dedos sobre la pantalla táctil.

Capítulo 5

Resultados

Este capítulo abordará un estudio de los resultados obtenidos en cada parte de este TFG. Para ello, en base a los objetivos se pondrán en práctica pruebas para las diferentes funcionalidades implementadas y se determinará si la salida obtenida es la esperada.

5.1 Cámara y OCR

El reconocimiento óptico de caracteres tiene múltiples aplicaciones comerciales como el análisis de documentos. Este proceso debe realizarse en cuestión de segundos para que suponga una ventaja respecto a la realización manual de la tarea. Los resultados obtenidos del módulo de OCR dependen de numerosos factores, entre ellos están la iluminación, la calidad de la imagen, el ángulo, la distancia o tamaño de texto, etc.

Se han realizado unos casos de prueba donde se combinan los diferentes factores y se han analizado los resultados obtenidos. Un 100 % de acierto representa un reconocimiento total de las palabras que forman parte del texto. Cada palabra mal reconocida decrementa ese porcentaje de acierto.

Una iluminación «Mala» significa que está en penumbra o con escasa luz mientras que una iluminación «Muy buena» significa que la superficie donde se va a realizar la toma de la fotografía está completamente iluminada. El tamaño del texto se ha probado con dos valores con la fuente *Tahoma*, el 100 % significa que el tamaño del texto es 10 y el 50 % significa que es tamaño 5.

Caso	Iluminación	Ángulo	Tamaño texto	Acierto
1	Muy buena	90º	100 %	>95 %
2	Muy buena	45º	100 %	~85 %
3	Muy buena	90º	50 %	~75 %
4	Muy buena	45º	50 %	~60 %
5	Media	90º	50 %	~40 %
6	Media	90º	100 %	~70 %
7	Media	45º	100 %	~60 %
8	Mala	90º	100 %	~50 %
9	Mala	90º	50 %	~20 %
10	Mala	45º	50 %	~5 %

Cuadro 5.1: Resultados OCR.

Las condiciones que afectan al reconocimiento óptico de caracteres son muy importantes, una mala iluminación puede afectar enormemente al resultado obtenido como se puede ver en el Caso 8 de la Tabla 5.1, que reduciendo la iluminación a «Mala» la tasa de acierto se reduce aproximadamente al 50 %. Por otra parte, el ángulo afecta menos a la tasa de acierto, en el Caso 2 donde el ángulo es de 45º y los otros factores son los ideales, se reduce aproximadamente al 85 %. En el Caso 3, se comprueba que reduciendo únicamente el tamaño del texto a la mitad se obtiene un acierto aproximadamente de 75 %.

La calidad de la imagen obtenida por la cámara es un factor también muy decisivo en la precisión del módulo OCR. Para medir la calidad de la imagen se ha utilizado los valores óptimos de iluminación, ángulo y tamaño de texto. La medida usada es DPI, que significa puntos por pulgada (en inglés *dots per inch*), y sirve para describir la resolución final de una imagen.

Calidad (DPI)	<100 DPI	150 DPI	200 DPI	>300 DPI
Tasa de fracaso	>90 %	~50 %	~15 %	despreciable

Cuadro 5.2: Resultados OCR - Calidad de la imagen.

Como se puede apreciar en la Tabla 5.2 a partir de 200 DPI la precisión en el proceso de reconocimiento de caracteres aumenta considerablemente.

En conclusión, si queremos obtener unos resultados óptimos del OCR se debe tener una buena iluminación, el ángulo de realización de la fotografía debe ser lo más próximo a 90º con respecto a la superficie del texto, el tamaño del texto no debe ser muy pequeño y la calidad de la impresión superior a 300 DPI. Estas condiciones no se tienen siempre por desgracia y esto puede afectar notablemente al resultado del sistema, proporcionando resultados diferentes a los deseados. La principal consecuencia que puede tener un resultado con baja tasa de acierto en el análisis del texto es la obtención errónea

de las palabras clave, lo que alteraría completamente el resultado final.

5.2 Procesamiento del Lenguaje Natural

El análisis del texto obtenido con técnicas de reconocimiento óptico de caracteres se ha realizado mediante el procesamiento del lenguaje natural. El resultado obtenido del análisis del texto depende totalmente del resultado obtenido del OCR, esto significa que si el resultado del módulo OCR ha tenido una tasa de acierto baja es muy probable que también sea desacertado su análisis y las fases siguientes.

Para realizar los casos de prueba se va a utilizar textos que están perfectamente formados y escritos suponiendo que ha recibido un resultado con una tasa de acierto del 100 % del OCR.

Se han realizado tres pruebas por cada caso. Cada prueba corresponde a un texto de diferente extensión pero tratando el mismo tema. Se comparará si el resultado obtenido es parecido al resultado esperado. En el resultado obtenido se representa entre paréntesis el factor de importancia de cada palabra y se ha limitado a 5 palabras.

El resultado esperado contiene las palabras que se han considerado con más relación ó que son más propensas a aparecer en cualquier definición de la palabra utilizada en cada caso. Estas están ordenadas de más concreta a más general con respecto al significado genérico. Por ejemplo, en el caso de «Tierra» y «Planeta» para la definición de «Planeta Tierra», interesa antes lo más concreto frente a lo más general ya que con «Planeta» podría obtener un planeta distinto a la Tierra y no se correspondería bien con el contexto, sólo en el caso de que no exista en el repositorio lo especifico se muestra algo más general.

Caso 1: Planeta Tierra

En este caso se va a tomar la definición proporcionada para «Planeta Tierra» con tres longitudes diferentes. El resultado esperado para esta definición debería ser el siguiente listado de palabras ordenado de mayor a menor factor de importancia:

1. Tierra
2. Sistema solar

En la tabla 5.3 se muestran los resultados obtenidos con respecto a cada longitud de texto.

Texto	Resultado obtenido
La Tierra es un planeta del sistema solar.	1. Tierra (0.76) 2. sistema solar (0.24)
La Tierra es un planeta del sistema solar que gira alrededor de su estrella —el Sol— en la tercera órbita más interna.	1. Tierra (0.42) 2. sistema solar (0.40) 3. estrella (0.07) 4. órbita (0.06) 5. Sol (0.05)
La Tierra es un planeta del sistema solar que gira alrededor de su estrella —el Sol— en la tercera órbita más interna. Es el más denso y el quinto mayor de los ocho planetas del sistema solar. También es el mayor de los cuatro terrestres o rocosos.	1. sistema solar (0.43) 2. Tierra (0.25) 3. órbita (0.09) 4. estrella (0.09) 5. Sol (0.03)

Cuadro 5.3: Resultados Caso 1.

En este caso el resultado más acertado es con la definición más corta mientras que en los textos más largos la variación de importancia se debe a que se menciona en más ocasiones el «sistema solar» y es un cambio lógico. El texto más extenso se asemeja más a una situación real, y a pesar de tener esta extensión es un resultado adecuado y se corresponde con lo esperado.

Caso 2: Arteria aorta

En este caso se va a tomar la definición proporcionada para «Arteria aorta» con tres longitudes diferentes. El resultado esperado para esta definición debería ser el siguiente listado de palabras ordenado de mayor a menor factor de importancia:

1. Aorta
2. Arteria
3. cuerpo
4. Sangre

En la tabla 5.4 se muestran los resultados obtenidos con respecto a cada longitud de texto.

Texto	Resultado obtenido
La aorta es la arteria principal.	1. aorta (0.53) 2. arteria (0.47)
La aorta es la arteria principal. Cumple la función de transportar sangre oxigenada hacia todas las demás arterias del cuerpo, excepto las pulmonares.	1. arteria (0.30) 2. aorta (0.22) 3. sangre (0.21) 4. arterias (0.07) 5. pulmonares (0.07)
La aorta es la arteria principal. Cumple la función de transportar sangre oxigenada hacia todas las demás arterias del cuerpo, excepto las pulmonares. Esta arteria nace del ventrículo izquierdo del corazón y desde allí baja hacia el abdomen.	1. arteria (0.52) 2. aorta (0.12) 3. arterias (0.08) 4. cuerpo (0.06) 5. sangre (0.06)

Cuadro 5.4: Resultados Caso 2.

Para este caso de prueba los tres resultados obtenidos son acertados aunque el texto más corto obtiene el más preciso. En los textos más largos la palabra «arteria» cobra importancia debido a que se repite, por lo tanto el resultado es adecuado y se corresponde con lo esperado.

Caso 3: Avión de caza

En este caso se va a tomar la definición proporcionada para «Avión de caza» con tres longitudes diferentes. El resultado esperado para esta definición debería ser el siguiente listado de palabras ordenado de mayor a menor factor de importancia:

1. Caza
2. Avión
3. Combate
4. Guerra

En la tabla 5.5 se muestran los resultados obtenidos con respecto a cada longitud de texto.

Texto	Resultado obtenido
Un avión de caza (también llamado avión de combate), o simplemente caza, es una aeronave militar.	1. avión de caza (0.64) 2. combate (0.22) 3. caza (0.14)
Un avión de caza (también llamado avión de combate), o simplemente caza, es una aeronave militar diseñada fundamentalmente para la guerra aérea con otras aeronaves.	1. avión de caza (0.55) 2. caza (0.18) 3. aeronaves (0.11) 4. combate (0.08) 5. guerra (0.07)
Un avión de caza (también llamado avión de combate), o simplemente caza, es una aeronave militar diseñada fundamentalmente para la guerra aérea con otras aeronaves, en oposición a los bombarderos, que están diseñados principalmente para atacar objetivos terrestres mediante el lanzamiento de bombas.	1. avión de caza (0.42) 2. bombarderos (0.14) 3. guerra (0.09) 4. combate (0.08) 5. bombas (0.05)

Cuadro 5.5: Resultados Caso 3.

En este caso a pesar de tener una extensión mayor sigue destacando «avión de caza» como término más importante e incluso destaca «bombarderos» en segundo lugar, que es un adjetivo que va muy ligado a este tipo de vehículo. Por lo tanto, en situaciones reales donde se suele tener un texto extenso, se obtiene un resultado adecuado y se corresponde con lo esperado.

La **conclusión** a estos resultados es que el éxito es si aparecen las palabras que se están buscando. A medida que el texto crece, la interpretación se complica. En los resultados se ve que el algoritmo extrae el contexto de manera correcta en todos los casos, incluso ampliando correctamente los términos asociados cuando aparecen en escena. La prioridad puede variar en función del número de repeticiones, posición que ocupa en la frase y si se trata de términos generalistas o no.

5.3 Algoritmo de Selección de Objeto 3D

Este proceso es el encargado de generar un *itemid* que indica qué objeto 3D del repositorio es el más adecuado para reproducir en Realidad Aumentada en base al análisis del texto realizado por el módulo del procesamiento del lenguaje natural, que ha generado un listado de tuplas con las palabras y su factor de importancia.

Para alguno de estos casos se toma el resultado obtenido de los casos de prueba del PLN, de manera que se observará si el comportamiento es el esperado, ya que como se ha explicado anteriormente si un módulo produce resultados desacertados estos serán arrastrados a los módulos siguientes y generarán también malos resultados.

En cada caso de prueba se mostrará el contenido de la tabla *items* y la tabla *keywords* de la base de

datos, de manera que se pueda apreciar los valores de cada campo y así poder comprender el resultado obtenido.

Caso 1: Planeta Tierra

Esta situación es un caso muy común en un repositorio de objetos 3D. Existen tres objetos 3D que tienen coincidencias en las mismas palabras y la única forma de elegir el más adecuado es mediante el sistema de puntuación, votos y *likes.*

El contenido de la tabla *items* está representado en la Figura 5.1 y el de la tabla *keywords* en la Figura 5.2. El contenido de las tablas está filtrado para mostrar únicamente las filas que nos interesan para esta prueba.

idItem	nombreObj	descripcion	categoria	owner	votos	likes	sumaPuntuaciones
41	Tierra	Tierra animada	1	42	12	3	18
42	Planeta Tierra	Earth	5	42	6	1	25
66	Tierra		1	42	2	1	7
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 5.1: Tabla *items* para las pruebas del Caso 1.

idkeywords	keywords	itemId
14	Tierra	41
16	sistema solar	41
25	estrella	41
19	Tierra	42
20	Sistema Solar	42
26	estrella	42
10	Tierra	66
11	Sistema Solar	66
12	estrella	66
NULL	NULL	NULL

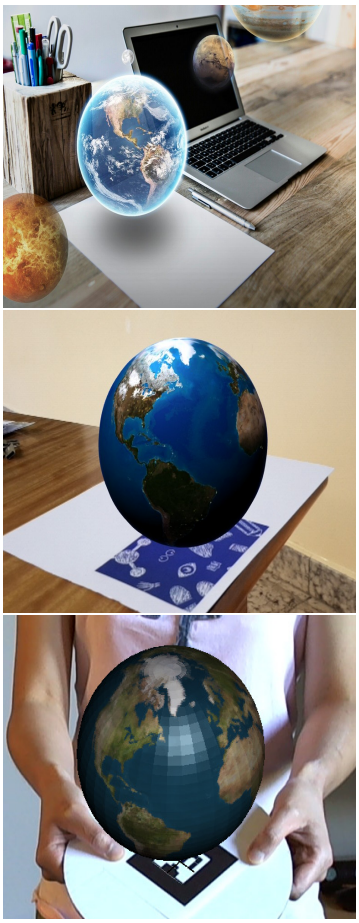
Figura 5.2: Tabla *keywords* para las pruebas del Caso 1.

En este caso el listado de tuplas obtenido como salida del módulo PLN tras el análisis del texto que contiene la definición de «Planeta Tierra» es el siguiente:

1. Tierra (0.42)
2. sistema solar (0.40)
3. estrella (0.07)

4. órbita (0.05)
5. Sol (0.05)

En esta situación podemos apreciar que hay más de un posible candidato. En la Tabla 5.6 se muestra la forma en la que el algoritmo soluciona esta situación de conflicto con éxito.

Descripción	Candidatos	Resultados
▪ Existen coincidencias en más de un *item.* ▪ Los itemID de los candidatos mostrados son: 41, 42, 66. ▪ El número de coincidencias máximas es el mismo en varios *items*. ▪ Se generan las combinaciones de palabras. ▪ Coinciden en las mismas palabras. ▪ Se procede al Desempate.		La puntuación obtenida para cada *item* al realizar el desempate es: ▪ 41=1.61 ▪ 42=1.3333 ▪ 66=1.3956 El modelo seleccionado correspondería al itemID=41. 

Cuadro 5.6: Resultados Algoritmo Selección Objeto 3D - Caso 1.

El conflicto encontrado se ha resuelto mediante el *desempate*, que gracias al sistema de puntuación, votos y *likes* donde han participado los usuarios se ha podido elegir el objeto mejor valorado en una situación de igualdad en coincidencias de palabras clave.

La fórmula para calcular la puntuación utilizada en el desempate es:

$$puntuación = \frac{notaMedia}{notaMax} + \frac{likes}{votos} + \frac{votos}{votosMax} \times \frac{likes}{likesMax} \tag{5.1}$$

Recordemos que: *notaMedia* = $\frac{sumaPuntuaciones}{votos}$ y *notaMax* = notaMedia más alta.

$$puntuaciónItem41 = \frac{\frac{18}{12}}{\frac{25}{6}} + \frac{3}{12} + \frac{12}{12} \times \frac{3}{3} = 1,61 \tag{5.2}$$

$$puntuaciónItem42 = \frac{\frac{25}{6}}{\frac{25}{6}} + \frac{1}{6} + \frac{6}{12} \times \frac{1}{3} = 1,3333 \tag{5.3}$$

$$puntuaciónItem66 = \frac{\frac{7}{2}}{\frac{25}{6}} + \frac{1}{2} + \frac{2}{12} \times \frac{1}{3} = 1,3956 \tag{5.4}$$

El modelo seleccionado correspondería al itemID=41, ya que ha obtenido la puntuación más alta.

Caso 2: León

La situación de este caso es que hay dos objetos con coincidencias en alguna palabra clave de la tupla al analizar un texto referido a «León», esto da lugar a un conflicto de homografía, ya que uno se refiere a un automóvil y el otro a un mamífero. El texto en cuestión es: «El león (Panthera leo) es un mamífero carnívoro de la familia de los félidos y una de las cinco especies del género Panthera.»

A continuación se muestran las figuras que contienen los valores almacenados en las tablas *items* y *keywords* de la base de datos. El contenido de las tablas está filtrado para mostrar únicamente las filas que nos interesan para esta prueba.

idItem	nombreObj	descripcion	categoria	owner	votos	likes	sumaPuntuaciones
44	Leon	León africano	5	42	0	1	0
67	Coche	Seat Leon	15	6	8	3	32
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 5.3: Tabla *items* para las pruebas del Caso 2.

idkeywords	keywords	itemId
42	automóvil	67
41	mamífero	44
40	félido	44
39	félidos	44
38	depredador	44
37	motor	67
36	Coche	67
35	Felino	44
34	Seat	67
33	León	67
32	León	44

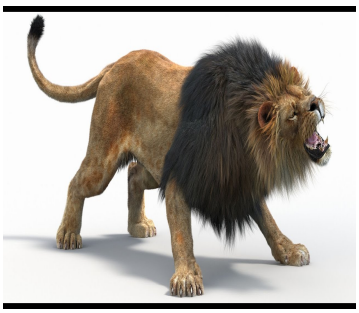
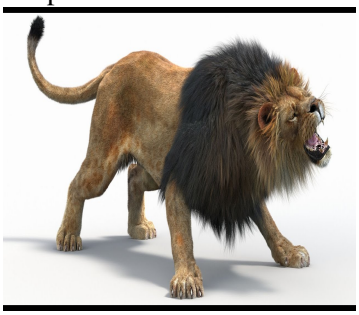
Figura 5.4: Tabla *keywords* para las pruebas del Caso 2.

En este caso el listado de tuplas obtenido como salida del módulo PLN tras el análisis de un texto que contiene la definición de «León» refiriéndose al felino es el siguiente:

1. Panthera (0.33)
2. león (0.15)
3. familia (0.14)
4. mamífero (0.10)
5. félidos (0.08)

En esta situación podemos apreciar que hay más de un posible candidato y además existe conflicto por la homografía. En la Tabla 5.7 se muestra la forma en la que el algoritmo soluciona esta situación de conflicto con éxito.

En esta situación hay conflicto de homografía que se ha resuelto gracias a las palabras clave que identificaban cada objeto 3D, por lo tanto el modelo seleccionado corresponde al itemID=44 por mayor número de coincidencias en las palabras clave. El resultado obtenido es el correcto a pesar del conflicto encontrado.

Descripción	Candidatos	Resultados
▪ Existen coincidencias en más de un *item.* ▪ Los itemID de los candidatos mostrados son: 44 y 67. ▪ El número de coincidencias máximas es 3 en un único *item*. ▪ No se generan las combinaciones de palabras. ▪ No es necesario el desempate.	 	El número de coincidencias de los *items* es: ▪ 67=1 ▪ 44=3 El modelo seleccionado correspondería al itemID=44. 

Cuadro 5.7: Resultados Algoritmo Selección Objeto 3D - Caso 2.

Caso 3

La situación a resolver por el algoritmo en este caso es la siguiente: recibe un listado de tres tuplas y las coincidencias son tres items con dos palabras diferentes encontradas en cada uno.

A continuación se muestran las figuras que contienen los valores almacenados en las tablas *items* y *keywords* de la base de datos. El contenido de las tablas está filtrado para mostrar únicamente las filas que nos interesan para esta prueba.

idItem	nombreObj	descripcion	categoria	owner	votos	likes	sumaPuntuaciones
68	Pelota de futbol	pelota de futbol	13	6	2	1	7
69	Pelota	de golf	13	42	15	8	60
70	Bola de golf	Pelota de golf	13	43	7	3	28

Figura 5.5: Tabla *items* para las pruebas del Caso 3.

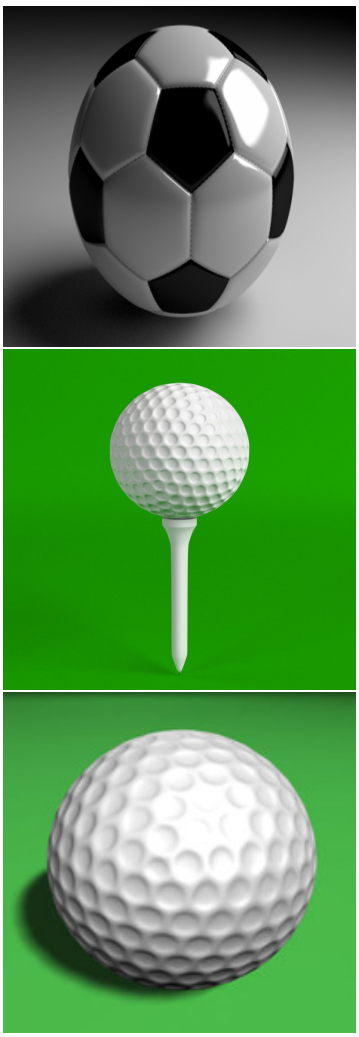
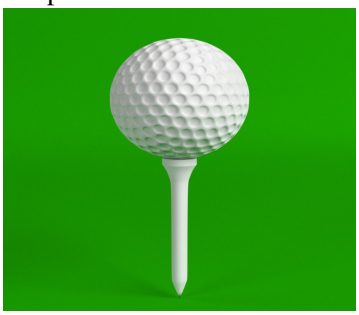
idkeywords	keywords	itemId
43	bola	68
46	pelota	68
49	futbol	68
44	deporte	69
47	pelota	69
50	golf	69
45	deporte	70
48	pelota	70
51	golf	70

Figura 5.6: Tabla *keywords* para las pruebas del Caso 3.

El texto analizado es: «La bola de golf es una bola específicamente diseñada para jugar al golf, usada con los palos de golf. No debe pesar más de 45,93 gramos.» En este caso el listado de tuplas obtenido como salida del módulo PLN tras el análisis de este texto que contiene la definición de «Pelota de golf» es el siguiente:

1. golf (0.46)
2. bola (0.19)
3. pelota (0.14)
4. palos (0.13)

En esta situación el conflicto es doble, porque existen 3 coincidencias con 2 palabras clave pero tiene que determinar si los 3 *items* son adecuados al contexto y de los que son adecuados realizar un desempate. En la Tabla 5.8 se muestra la forma en la que el algoritmo soluciona esta situación de conflicto con éxito.

Descripción	Candidatos	Resultados
▪ Existen coincidencias en más de un *item.* ▪ Los itemID de los candidatos mostrados son: 68, 69 y 70. ▪ El número de coincidencias máximas es 2 en un único *item*. ▪ Se generan las combinaciones de palabras. ▪ 2 de 3 candidatos tienen mayor suma de *FI*. ▪ Se procede al desempate.		El número de coincidencias de todos *items* es 2. Se generan combinaciones de grupos de 2 palabras con mayor suma de *FI*. ▪ 69=[golf, pelota] ▪ 70=[golf, pelota] Se procede al desempate con estos 2 candidatos: ▪ 69=2.5333 ▪ 70=1.6036 El item con ID=69 recibe una mayor puntuación por lo tanto es el más adecuado. El modelo seleccionado correspondería al itemID=69. 

Cuadro 5.8: Resultados Algoritmo Selección Objeto 3D - Caso 3.

En este caso existen tres candidatos pero uno de ellos no es un resultado adecuado para este contexto, mediante la obtención de la suma del *FI* en los grupos de 2 palabras generados se filtra este resultado y quedan los que si son adecuados que, mediante el desempate, se identifica al objeto 3D con mejor valoración según los usuarios.

La fórmula para calcular la puntuación utilizada en el desempate es:

$$puntuación = \frac{notaMedia}{notaMax} + \frac{likes}{votos} + \frac{votos}{votosMax} \times \frac{likes}{likesMax} \tag{5.5}$$

Recordemos que: *notaMedia* = $\frac{sumaPuntuaciones}{votos}$ y *notaMax* = notaMedia más alta.

$$puntuaciónItem69 = \frac{\frac{60}{15}}{4} + \frac{8}{15} + \frac{15}{15} \times \frac{8}{8} = 2,5333 \tag{5.6}$$

$$puntuaciónItem70 = \frac{\frac{28}{7}}{4} + \frac{3}{7} + \frac{7}{15} \times \frac{3}{8} = 1,6036 \tag{5.7}$$

El modelo seleccionado correspondería al itemID=69, ya que ha obtenido la puntuación más alta. Por lo tanto, el algoritmo ha resuelto con éxito este doble conflicto obteniendo el resultado esperado.

5.4 Realidad Aumentada

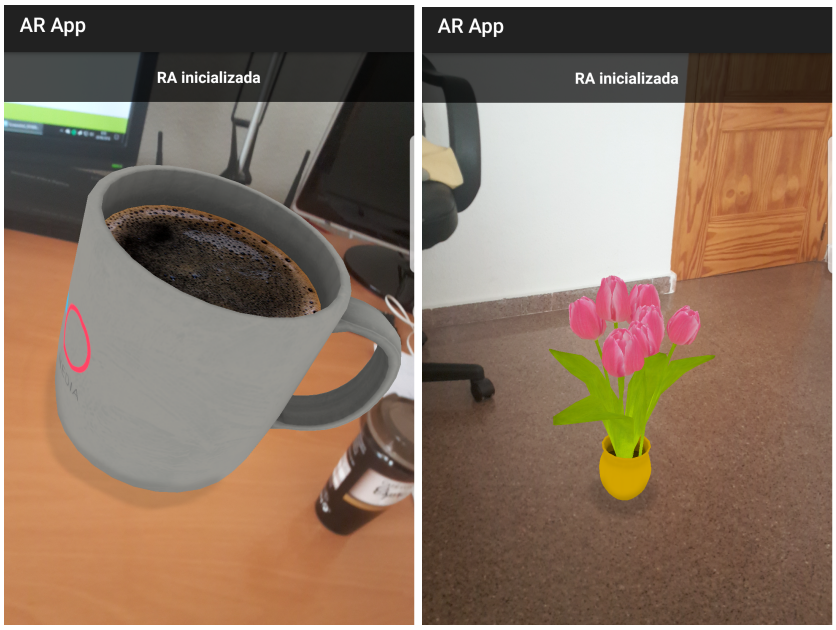
En esta sección se va a mostrar los resultados de la representación de modelos 3D con la librería integrada ViroCore para Android y la posibilidad que proporciona de desplazar objetos, rotarlos y modificar su tamaño con gestos de los dedos sobre la pantalla táctil permitiendo al usuario visualizar el objeto 3D desde diferentes ángulos, tamaños y distancia.

ViroCore soporta la carga de modelos 3D en formato FBX y OBJ, el formato FBX necesita ser convertido al formato propio de Viro mientras que el OBJ se puede cargar directamente. Se van a probar dos casos diferentes uno para el formato VRX proporcionado por la propia librería y otro con formato OBJ, que es el más extendido entre los objetos 3D.

Caso VRX

Las resultados obtenidos van a estar basados en la comprobación del cambio de superficie, rotación, desplazamiento, cambio de tamaño y visualización desde ángulos diferentes.

En la Figura 5.7 formada por dos imágenes se obtiene como resultado una perfecta integración en dos superficies diferentes, en la Figura 5.7a el objeto está sobre una mesa y en la Figura 5.7b está situado en el suelo. Ambos modelos 3D tienen el mismo formato.



(a) Superficie A (b) Superficie B

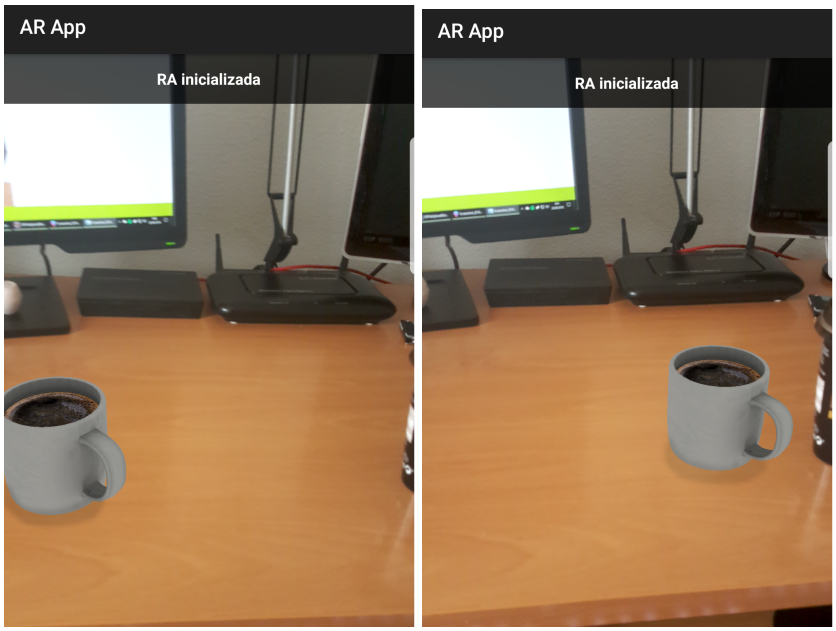
Figura 5.7: Superficies diferentes.

En el mosaico de la Figura 5.8 se muestra el mismo objeto 3D que se ha modificado su posición rotándolo sobre si mismo de manera satisfactoria mediante un gesto con dos dedos sobre la pantalla táctil.

(a) Rotación A (b) Rotación B

Figura 5.8: Cambio de posición por rotación.

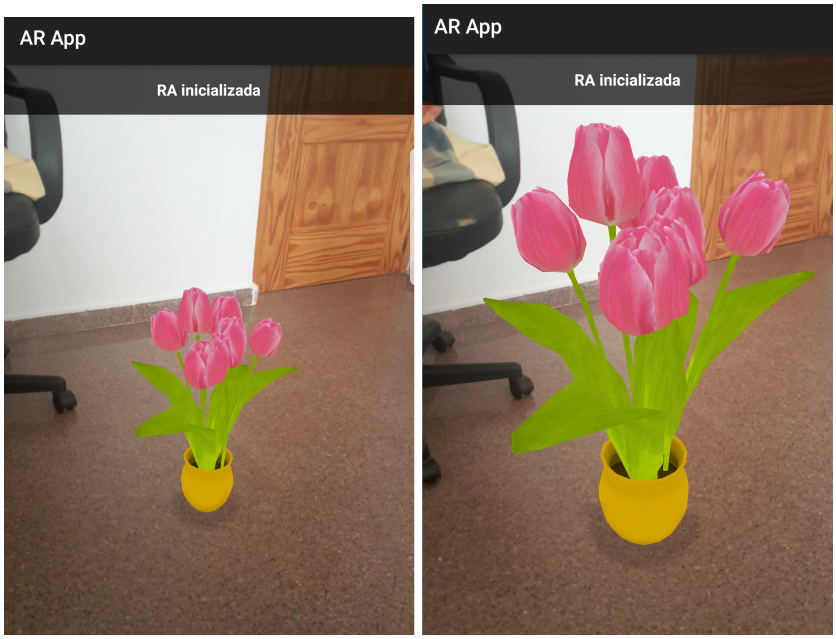
El desplazamiento de un objeto 3D se realiza mediante el gesto con un sólo dedo sobre la pantalla táctil arrastrando el objeto hacia dónde el usuario desea. En la composición de imágenes de la Figura 5.9 se puede apreciar como un mismo objeto 3D ha sido desplazado entre dos puntos de una misma superficie.



(a) Posición A (b) Posición B

Figura 5.9: Cambio de posición por desplazamiento.

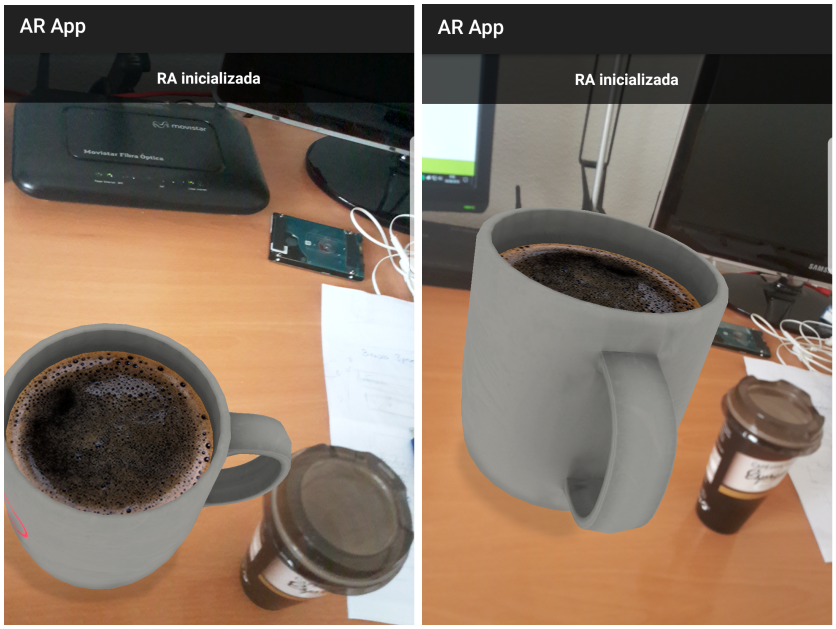
En la Figura 5.10 se muestra el resultado de haber cambiado de tamaño un objeto 3D mediante un gesto con dos dedos en la pantalla táctil que consiste en separar los dedos para agrandar y acercar para encoger de tamaño.



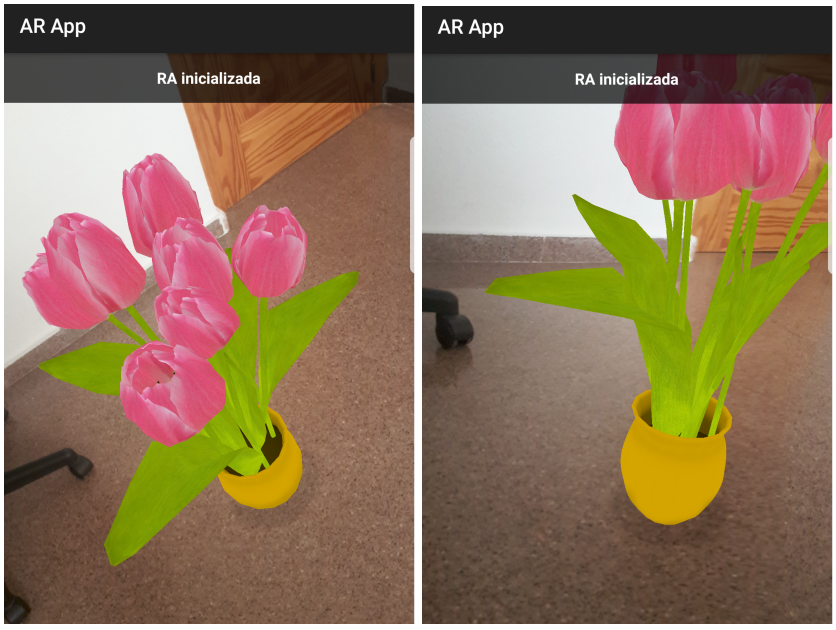
(a) Tamaño A (b) Tamaño B

Figura 5.10: Cambio de tamaño.

A continuación se va a mostrar como los modelos 3D se pueden visualizar desde diferentes ángulos con un simple desplazamiento del dispositivo móvil como si fuera un objeto real sobre una superficie.



(a) Ángulo A1 (b) Ángulo A2



(c) Ángulo B1 (d) Ángulo B2

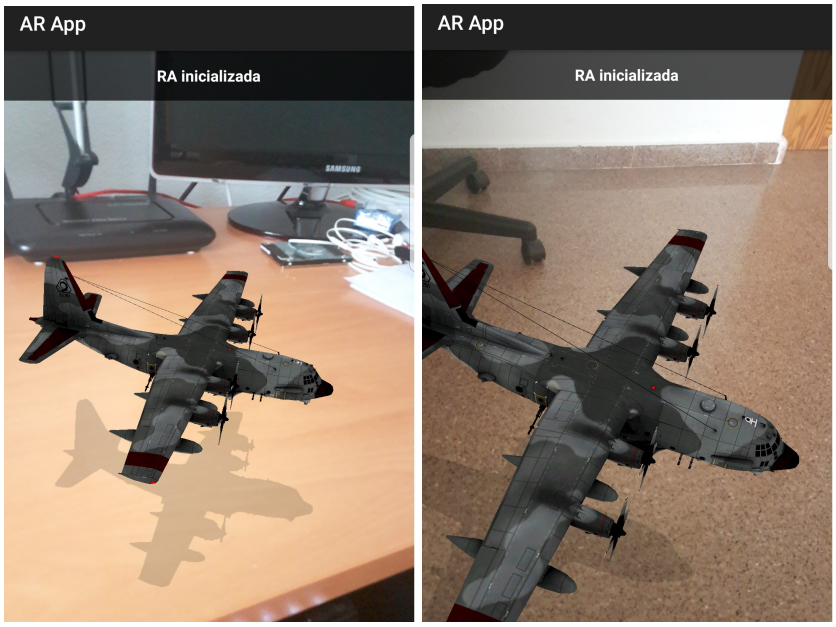
Figura 5.11: Cambio de ángulo.

Los resultados obtenidos con los modelos VRX en cada prueba son los esperados, tanto la representación y carga del modelo como el comportamiento de los gestos con los dedos para poder interactuar responden correctamente.

Caso OBJ

Se va a mostrar un modelo con formato OBJ más realista de un avión AC130 utilizado por las fuerzas aéreas de Estados Unidos comprobando los resultados en todas las situaciones mencionadas anteriormente.

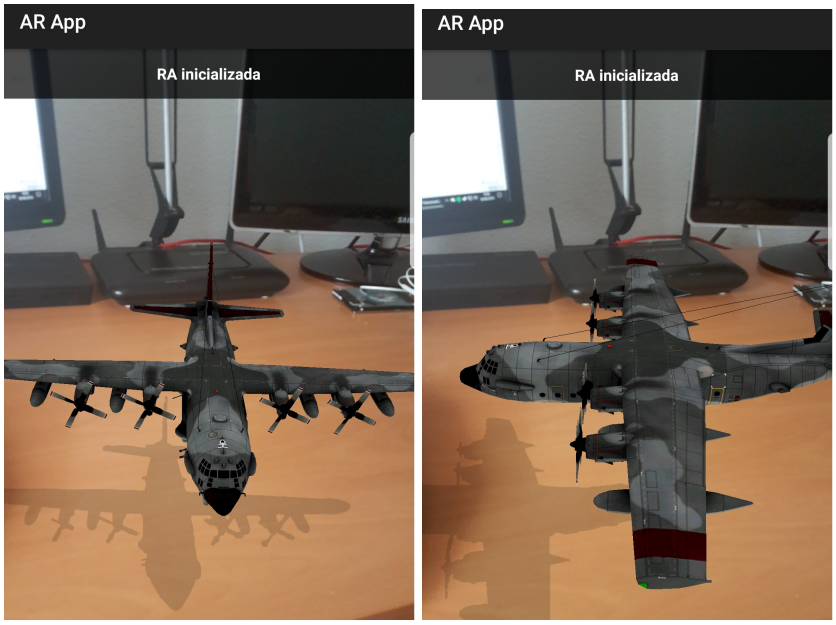
El modelo 3D se puede representar en cualquier superficie que detecte, en la siguiente figura se muestra como se ha reproducido sobre una mesa y en el suelo.



(a) Superficie A (b) Superficie B

Figura 5.12: Superficies diferentes.

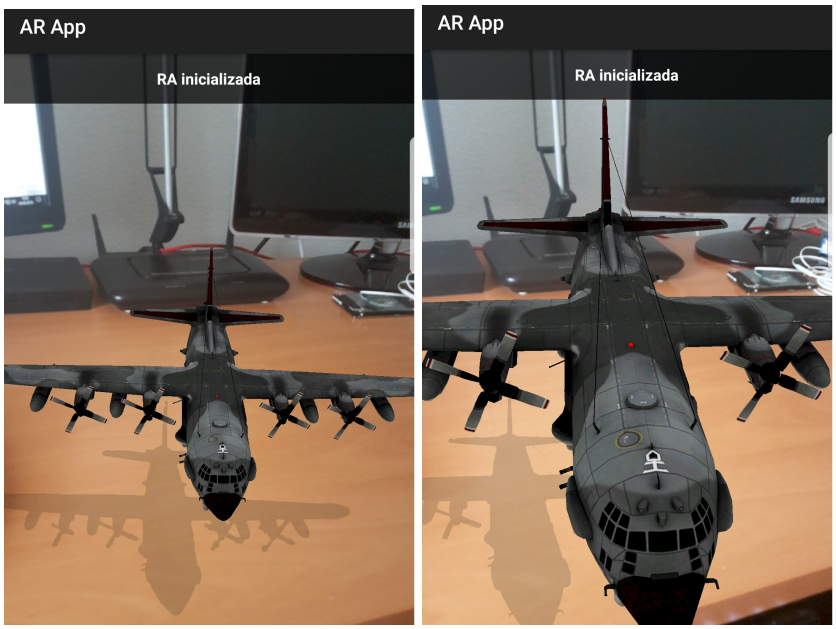
Mediante los gestos de dos dedos permite realizar una rotación del objeto 3D como se puede observar en los resultados de la Figura 5.13.



(a) Rotación A (b) Rotación B

Figura 5.13: Cambio de posición por rotación.

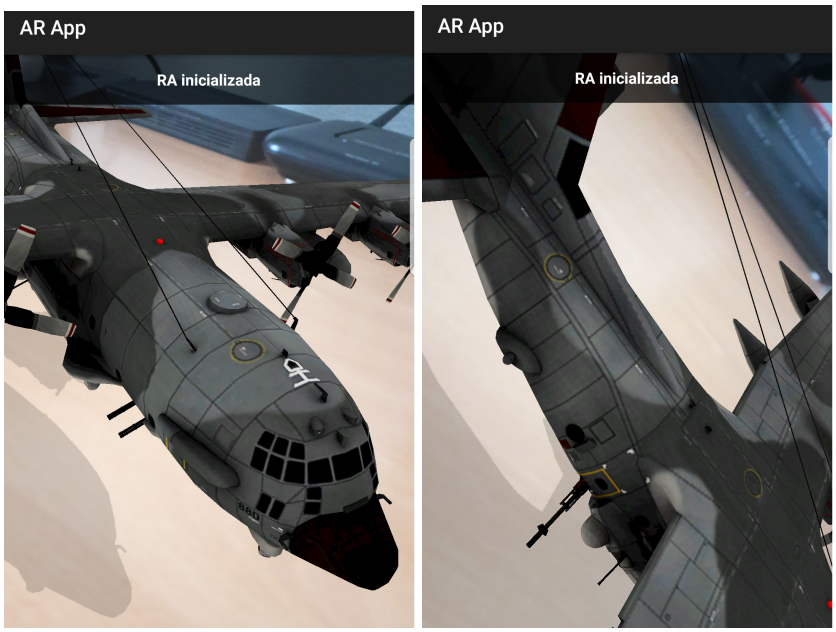
En la Figura 5.14 se muestra el resultado de haber cambiado de tamaño un objeto 3D mediante un gesto con dos dedos en la pantalla táctil que consiste en separar los dedos para agrandar y acercar para encoger de tamaño.



(a) Tamaño A (b) Tamaño B

Figura 5.14: Cambio de tamaño.

En la Figura 5.15 se muestra como el modelo 3D se puede visualizar desde diferentes ángulos, permitiendo al usuario obtener una información visual enriquecida.



(a) Ángulo A (b) Ángulo A

Figura 5.15: Cambio de ángulo.

Los resultados obtenidos con el modelo 3D en formato OBJ son los esperados, tanto en la carga del modelo como en la funcionalidad que permite al usuario interactuar con gestos de los dedos para cambiar la posición, la rotación o el tamaño.

La **conclusión** es que representando el modelo 3D en Realidad Aumentada no sólo se consigue que el usuario pueda cambiar la posición con gestos de los dedos (que se podría hacer con cualquier aplicación para representar objetos 3D sin RA), sino que además tienes la libertad de movimiento con el móvil lo que permite al usuario visualizar el objeto 3D sobre una superficie real desde diferentes perspectivas.

Capítulo 6

Conclusiones y trabajo futuro

La escritura es un medio de comunicación que apareció hace 5.000 años y que actualmente se sigue utilizando como uno de los medios más importantes, aunque tiene limitaciones que otros medios más modernos basados en nuevas tecnologías no tienen. Por ejemplo, la visualización de un objeto impreso en papel tiene la limitación de movimiento, sonido, capacidad de interacción en general, etc. Estas limitaciones podrían ser solventadas con la fusión con ciertas tecnologías. La Realidad Aumentada durante la última década está en auge, esto es debido a la aparición de dispositivos como los *smartphones* que permiten hacer uso de ella en cualquier lugar. Combinando estos nuevos medios tecnológicos con la información a papel se podría disfrutar de una experiencia de lectura innovadora y enriquecedora.

Recordemos que el objetivo principal de este trabajo era enriquecer la información textual impresa en papel con información virtual integrada, de tal forma que se pueda animar y haya libertad de movimiento para examinar cualquier parte tal como si existiera un objeto real. Para ello se ha diseñado y desarrollado un sistema de Realidad Aumentada en dispositivos móviles que mediante la cámara el usuario puede obtener una representación de un objeto 3D relacionado con el contexto de la información textual.

Para obtener el objeto 3D, el sistema debe hacer un reconocimiento óptico de caracteres para obtener el texto digitalizado y así poder obtener las palabras clave mediante técnicas de procesamiento del lenguaje natural. En base a este listado de palabras clave el servidor encuentra un objeto 3D del repositorio mediante un algoritmo propio que hace la selección teniendo en cuenta varios factores como el número de coincidencias en las *keywords*, el número de *likes*, el número de votos y la puntuación. Este objeto 3D seleccionado es enviado al dispositivo móvil para poder reproducirlo mediante técnicas de Realidad Aumentada. En definitiva, se trata de mostrar un objeto 3D que mejor se ajuste al contexto de lectura del usuario y enriquezca visualmente aquello que está leyendo.

Con el fin de tener un amplio repositorio de objetos 3D, también se ha desarrollado un sitio web que permite la compartición de contenidos 3D. Así, se incrementa la probabilidad de que existan contenidos disponibles para cualquier temática.

6.1 Objetivos alcanzados

Los objetivos planteados en el Capitulo 2 se han cumplido, para ello han hecho falta alrededor de 8 meses de trabajo. El funcionamiento del trabajo realizado es satisfactorio como se ha mostrado en el capítulo anterior ya que en las pruebas realizadas se extrae la información de un texto en papel

y se puede ver representado un objeto 3D en Realidad Aumentada que se ajusta al contexto. Los subobjetivos se han alcanzado de la siguiente manera:

- El reconocimiento óptico de caracteres se ha realizado mediante la librería Google Cloud Vision API. Previamente se probaron otras librerías como Tesseract, que fue descartada por su ligera tasa de error superior a la librería elegida, o IBM Cloud Vision Recognition API, que su versión gratuita daba peores prestaciones que la de Google. La parte más problemática fue la cámara, ya que es muy importante la focalización de la imagen, la iluminación, el encuadre y la calidad. La focalización automática se realiza cuando el usuario presiona el botón para tomar la fotografía, esto mejora notablemente los resultados del OCR y, por otra parte, el encuadre que se ha conseguido mediante el rectángulo de selección en la cámara y que es determinante a la hora de realizar el análisis de texto como se ha explicado en el capítulo anterior.
- El análisis del texto se ha realizado mediante la librería Google Cloud Natural Language API, aunque antes de llegar a esta librería se probaron otras como ParallelsDots o IBM Watson Natural Language pero se adaptaban peor a las necesidades del proyecto y no aportaban los resultados deseados. Tal y como se muestra en los resultados la elección final es adecuada porque extrae las palabras clave de manera más precisa y la versión gratuita para desarrolladores se adapta mejor al uso que se le ha dado en este trabajo.
- La comunicación entre el Cliente Android y el Servidor es una parte muy importante para el funcionamiento del sistema. Había muchas posibilidades de comunicar estas dos partes pero la óptima para que sea escalable y rápida es mediante servicios web, en este caso JAX-RS. Esta librería usa anotaciones lo que permite simplificar el desarrollo y despliegue del cliente y del servicio web, además de formar parte de Java EE (a partir de la versión 6).
- Para que el servidor pueda generar la respuesta se ha realizado un algoritmo de selección de objeto 3D que mediante decisiones y cálculos matemáticos encuentra el resultado más adecuado en el repositorio. En el capítulo de resultados se muestra como resuelve los casos conflictivos donde existen perfiles de objetos muy similares y hay que desempatarlos en base a una fórmula matemática compuesta por factores que permiten diferenciar unos objetos de otros, como son los votos, puntuación, *likes*, etc.
- La representación del objeto 3D en Realidad Aumentada es la parte del proyecto que más esfuerzo ha requerido. Se han probado varias librerías y finalmente la que mejor resultados ha ofrecido es ViroCore que tiene actualizaciones muy continuadas. Se trata de una librería muy completa, la cual, permite implementar un gran número de funcionalidades. La comunidad colabora bastante en foros, hay una muy buena documentación y se adapta bien para todo tipo de desarrolladores. En general, las librerías integradas de Android para Realidad Aumentada que son gratuitas actualmente tienen muchos problemas pendientes por resolver en comparación con otras plataformas. Las librerías que se han descartado hasta llegar a la utilizada han sido: EasyAR, que proporciona muy buena información para aprender pero tiene mucho que mejorar con la compatibilidad con algunos formatos; ARToolKit, la versión de Android esta totalmente desactualizada o falta información para poder integrarlo; ARCore, (a pesar de que ViroCore funcione con ARCore integrado también) se intentó hacerlo con ARCore el problema es que

para hacer una implementación sin un framework tienes que conocer a fondo el funcionamiento de la librería OpenGL ya que necesitas programar a muy bajo nivel, por lo que se considera la mejor opción pero complementada con un *framework*; Vuforia, es una librería completa, de calidad y con mucho soporte pero con la necesidad de incorporar librerías externas que la complementen para poder leer modelos 3D con gran cantidad de polígonos; Wikitude, es una librería con su propio renderizador de RA que está muy actualizada pero su integración en comparación con ViroCore es bastante más confusa ya que la documentación es poco intuible y no ayuda lo suficiente.

- El sitio web colaborativo se ha realizado desde el principio con Java EE y JavaServer Faces siguiendo un patrón de diseño MVC que es una solución efectiva al problema de arquitectura que plantea la necesidad de separar la parte de presentación o vistas y la parte del manejo de datos o modelo.
- El repositorio permite al usuario tener un perfil, visualizar perfiles de objetos 3D del repositorio, votar y darle a *like*, insertar nuevos objetos 3D y visualizar los perfiles de objetos propios o que le han gustado al usuario. El sistema de votos, puntuación y *likes* es muy importante para el algoritmo de selección de objeto 3D a la hora de deshacer casos de empate.

6.2 Aportación personal

Después de varios años estudiando y de muchos retos en las asignaturas, creo que lo más importante es la habilidad adquirida para aprender y asimilar lo novedoso. En este TFG ha habido conocimientos de asignaturas que me han aportado mucho directamente porque su contenido tenia relación con ciertas partes del TFG y otras que me han aportado mucho indirectamente, es decir, me han enseñado como enfrentarme a esos retos empezando desde cero.

En concreto en este proyecto ha habido muchos campos que eran totalmente nuevos para mí, como por ejemplo, los módulos de OCR, PLN, Realidad Aumentada y Comunicación; y por otra parte, para el desarrollo del algoritmo del servidor, el SGBD y el cliente web tenía ciertos conocimientos aprendidos en algunas asignaturas.

Las asignaturas que considero que me han aportado conocimientos útiles en este TFG son Tecnologías y Sistemas Web en la parte del Cliente Web, SGBD y servicios web de la Comunicación, que aunque no hayamos usado las mismas tecnologías el funcionamiento guarda similitud; Bases de Datos en el diseño y comprensión del funcionamiento de las bases de datos relacionales; Metodología de la programación y Estructuras de Datos para una buena estructuración del código mediante patrones de diseño entre otros conocimientos; Ingeniería del Software para todo lo relacionado con la gestión del proyecto; Sistemas Inteligentes y Sistemas Distribuidos me han aportado conocimientos que han sido útiles para el desarrollo del sistema, en especial del servidor y el algoritmo de selección; *Computer Graphics* me ha servido bastante como base de conocimientos para aprender en el campo de Realidad Aumentada.

Antes de empezar con el TFG los campos de la informática que más me llamaban la atención eran los relacionados con el desarrollo web y la seguridad informática, después de haberlo terminado puedo decir que la Realidad Aumentada y los temas relacionados también.

Uno de mis dos requisitos al elegir un tema para el TFG es que el resultado final fuera algo útil y el otro es que el tema en sí me gustara. Dicho esto, espero que el proyecto sea de utilidad para la gente ya que considero que aporta una herramienta novedosa a un medio de comunicación tan tradicional como la información en papel.

6.3 Trabajo futuro

El tema tratado en este TFG es muy amplio, innovador y puede ser mejorado en múltiples aspectos aunque por limitaciones de tiempo se ha tenido que acotar el trabajo. Las mejoras que podrían aplicarse a este proyecto son:

- Implementación de una funcionalidad extra en el repositorio que permita ver representado el objeto 3D en el navegador mediante librerías de Realidad Aumentada y el uso de una cámara o simplemente que permita la visualización del objeto 3D de forma que se pueda apreciar el modelo sin depender de la aplicación móvil. Actualmente hay librerías para JavaScript que permiten de manera sencilla reproducir objetos 3D y que encajarían perfectamente con la arquitectura del Cliente Web.
- El diseño del repositorio puede tener forma de *grid* de manera que tendría una interfaz más atractiva visualmente para el usuario.
- En un escenario en el que la aplicación móvil recibe el objeto 3D y lo reproduce, puede permitir al usuario votar y darle a *like*. Esto también implicaría un sistema de gestión de usuarios en el Cliente Android. La implementación de esta funcionalidad fomentaría la participación de los usuarios en el sistema de votaciones, puntuación y *likes*, lo que incrementaría la información que hay de cada objeto 3D en el repositorio y mejoraría los resultados obtenidos por el algoritmo de selección de objetos 3D.
- Implementación de una vista en la aplicación móvil que permita al usuario entrar al repositorio con acceso completo a sus funcionalidades. De esta manera, la parte colaborativa del sistema estaría mucho más relacionada para que mayor número de usuarios participen.
- Añadir una biblioteca en la aplicación móvil que permita reproducir sin conexión los objetos 3D previamente vistos o los que el usuario seleccione.
- Con el fin de llegar a más usuarios, adaptar la implementación de la aplicación móvil para otras plataformas muy conocidas como iOS. Actualmente iOS y Android lideran en el mundo de los *smartphones*.
- Dar soporte a varios idiomas tanto en la aplicación móvil como en el sitio web, ya que actualmente está sólo en castellano.

ANEXOS

Anexo A

Otros resultados

A.1 SGBD

En esta sección se incluye la operaciones que aportan las clases DAO y la encriptación de la contraseña del usuario mediante el procedimiento almacenado en el SGBD.

Las operaciones con las tablas de la base de datos que realizan las clases DAO corresponden con las instrucciones SELECT, INSERT, UPDATE y DELETE. Como para cada tabla hay que repetir el método de cada instrucción se va a realizar dos pruebas, de manera que el resultado sea lo más completo posible.

El método `selectSumaPuntuaciones` de la clase *DAOItems* realiza un **SELECT** para obtener la columna *SumaPuntuaciones* del *itemid* especificado de la tabla *items* (ver Listado A.1).

```
1  public static int selectSumaPuntuaciones(int itemId) throws Exception {
2  Connection bd = null;
3  int resultado = 0;
4  try {
5  String sql = "SELECT sumaPuntuaciones FROM items WHERE idItem=?";
6  DataSourceService dss = new DataSourceService();
7  bd = dss.getDataSource().getConnection();

9  PreparedStatement ps = bd.prepareStatement(sql);
10 ps.setInt(1, itemId);
11 ResultSet rs = ps.executeQuery();
12 while (rs.next()) {
13 resultado = rs.getInt(1);
14 }
15 return resultado;
16 } catch (Exception e) {
17 throw e;
18 } finally {
19 bd.close();
20 }
21 }
```

Listado A.1: Método `selectSumaPuntuaciones` de la clase *DAOItems*

Si se toma como ejemplo los valores de la Figura A.1, al realizar: «selectSumaPuntuaciones(41)» donde 41 es un *itemid* con *sumaPuntuaciones*=19 devuelve el resultado esperado.

idItem	nombreObj	descripcion	categoria	owner	votos	likes	sumaPuntuaciones
41	Tierra	Tierra animada	1	42	12	3	19
42	Pantera		5	42	6	1	25
43	Perro	Pitbull	2	6	1	2	3
44	Leon		5	42	0	1	0

Figura A.1: Ejemplo tabla *items*.

La instrucción **INSERT** se va a probar con el método `insert` de la clase *DAOKeywords* que pasándo por parámetros la *keyword* y el *itemid* inserta en la tabla *keywords* las palabras clave y su el *item* al que hace referencia.

```
1  public static void insert(String keyword, int itemId) throws Exception {
2  Connection bd = null;
3  try {
4  String sql = "Insert into keywords (keywords, itemId) values (?, ?)";
5  DataSourceService dss = new DataSourceService();
6  bd = dss.getDataSource().getConnection();
7  PreparedStatement ps = bd.prepareStatement(sql);
8  ps.setString(1, keyword);
9  ps.setInt(2, itemId);
10 ps.executeUpdate();
11 } catch (Exception e) {
12 throw e;
13 } finally {
14 bd.close();
15 }
16 }
```

Listado A.2: Método `insert` de la clase *DAOKeywords*

Para probar que inserta correctamente en la tabla *keywords* se hace una llamada al método con los parámetros (*testing*, 41), por lo que inserta la palabra «testing» referenciada al *itemid* 41 (ver Figura A.2).

	idkeywords	keywords	itemId
	14	Tierra	41
	15	sistema solar	41
▶	21	testing	41
	19	datos	42

Figura A.2: Inserción de palabra clave en la tabla *keywords*.

Por otra parte, si los parámetros son (*testing*, 39) siendo 39 un *itemid* que no existe su comportamiento es el adecuado mostrando una excepción SQL.

En el proceso de inserción de usuarios en la base de datos para el registro se encripta la contraseña. La **contraseña está encriptada** mediante el algoritmo *mysql_native_password* de MySQL. En el Listado A.3 se muestra el *script* que contiene la instrucción que lista los usuarios y la contraseña encriptada almacenada.

```
1 SELECT User, authentication_string FROM mysql.user;
```

Listado A.3: Instrucción del *script* que lista los usuarios del SGBD

El resultado mostrado al ejecutar esta instrucción en MySQL (ver Figura A.3) sirve para comprobar que las contraseñas de los usuarios están almacenadas correctamente, mediante el algoritmo propio del SGBD que está explicado en la Sección 4.1.3.

User	authentication_string
user35	*91D9861DFC07DD967611B8C96953474EF270AD5E
user36	*23AE809DDACAF96AF0FD78ED04B6A265E05AA257
user37	*07B5C78C48C5DAA9A6C97AEA6F2961F7CC071682
user38	*86EDD3F7A323A0FE27E9FABBA4BA6900E0DFC943
user39	*667F407DE7C6AD07358FA38DAED7828A72014B4E
user41	*1A256E4E2FE95B8BF7349C168991EA8035D1359B

Figura A.3: Contraseñas encriptadas en el SGBD.

A.2 Comunicación

La comunicación entre el Cliente Android y el Servidor es bidireccional, es decir, el cliente envía un texto plano que contiene las tuplas y el servidor debe devolver una respuesta que puede ser un archivo comprimido o un aviso de que no existe respuesta.

Para comprobar que el servicio web por parte del servidor recibe y envía la respuesta se ha utilizado

ARC (Advanced REST Client), que permite simular el funcionamiento de un cliente.

Para comprobar el resultado se crea una *request* con los siguientes parámetros definidos:

- **Request URL:** contiene la dirección URL del servicio definido en el servidor (ver número 1 en Figura A.5).
- **Headers:** se define el tipo de contenido del cuerpo en la *request* (ver número 3 en Figura A.5)y el tipo de contenido que acepta en la *response* (ver número 2 en Figura A.5).
- **Body:** el contenido del cuerpo es un ejemplo de una cadena de tuplas que tiene el mismo formato que las generadas por el Cliente Android (ver Figura A.4).

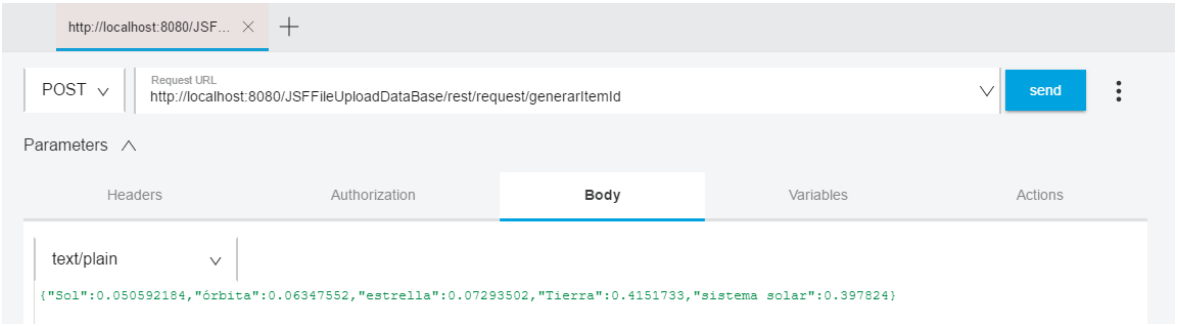


Figura A.4: Contenido del cuerpo de la *request*.

El resultado es una *response* recibida que contiene el archivo comprimido en formato ZIP con los archivos del objeto 3D y un código de respuesta «200 OK» como se espera cuando se ha encontrado el resultado (ver número 4 en la Figura A.5).

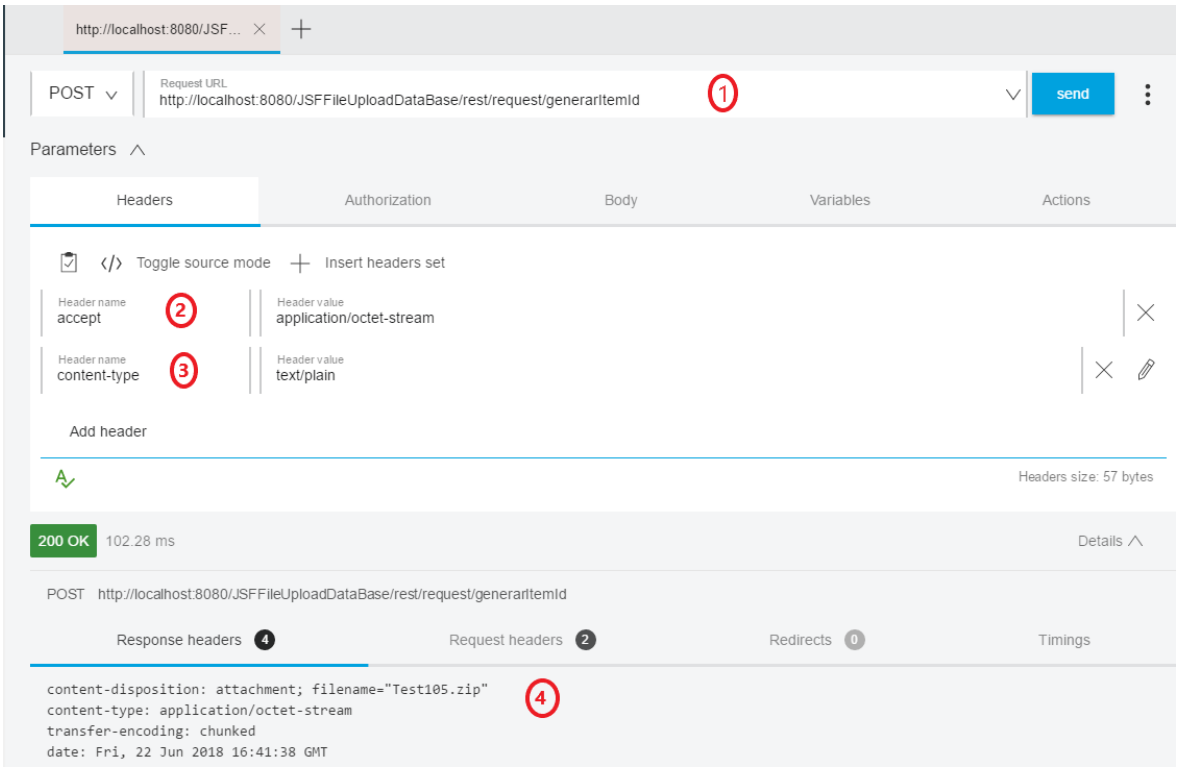


Figura A.5: Resultado de la petición.

Referencias

[ABB18] ABBYY. Abbyy mobile ocr engine website, 2018. URL https://www.abbyy.com/en-eu/mobile-ocr/features/.

[ACG17] P. Badelia A. Chaudhuri, K. Mandaviya and S.K. Ghosh. *Optical Character Recognition Systems for Different Languages with Soft Computing*. Springer, 2017.

[AH11] Leila Alem and Weidong Huang. *Recent Trends of Mobile Collaborative Augmented Reality Systems*. Springer, 2011.

[Apa13] Apache. Apache wink documentation, 2013. URL https://wink.apache.org/documentation.html.

[Apa17] Apache. Apache opennlp website, 2017. URL https://opennlp.apache.org/.

[Apa18] Apache. Apache cxf architecture guide, 2018. URL http://cxf.apache.org/docs/cxf-architecture.html#CXFArchitecture-References.

[App18] Apple. Arkit documentation, 2018. URL https://developer.apple.com/documentation/arkit.

[Aru16] U. Arumilli. *SQL the One: Microsoft SQL Server Interview Guide*. Notion Press, 2016.

[Asp18] Asprise. Asprise ocr website, 2018. URL https://asprise.com/royalty-free-library/java-ocr-api-overview.html.

[Azu97] Ronald Azuma. *A survey of augmented reality*. Presence, 1997.

[Bal17] Bogunuva Mohanram Balachandar. *RESTful Java Web Services*. Packt Publishing Ltd, 2017.

[Ban14] A. K. Bansal. *Introduction to Programming Languages*. CRC Press, 2014.

[Bea15] M. Bean. *Laravel 5 Essentials*. Packt Publising, 2015.

[Ben12] J. Bengoechea. *Microsoft Access: Diseño de aplicaciones sencillas de bases de datos*. Ideaspropias Editorial, 2012.

[Bit18] Bitbucket. Bitbucket website, 2018. URL https://bitbucket.org.

[Blo14] The Blokehead. *Scrum : Ultimate Guide to Scrum Agile Essential Practices!* Booktango, 2014.

[BZ07] G. Baklarz and Paul C. Zikopoulos. *DB2 9 for Linux, UNIX, and Windows*. IBM Press, 2007.

[CGC12] J. A. Albusac C. González, D. Vallejo and J.J. Castro. *Realidad Aumentada. Un enfoque práctico con ARToolKit y Blender*. Bubok Publishing, 2012.

[CH13] D. Cushnan and H. El Habbak. *Developing AR Games for iOS and Android*. Packt Publishing Ltd, 2013.

[Con18] SQLite Consortium. Sqlite website, 2018. URL `https://www.sqlite.org/docs.html`.

[Cor18] Oracle Corp. Jersey documentation, 2018. URL `https://jersey.github.io/documentation/latest/index.html`.

[Cun01] Ward Cunningham. Manifiesto Ágil website, 2001. URL `http://agilemanifesto.org/iso/es/principles.html`.

[Day15] Brad Dayley. *NoSQL with MongoDB in 24 Hours*. Sams, 2015.

[DE18] DB-Engines. Db-engines ranking, 2018. URL `https://db-engines.com/en/ranking`.

[Dec16] P. Deck. *Spring MVC: A Tutorial (Second Edition)*. BrainySoftware, 2016.

[DuB13] P. DuBois. *MySQL*. Kitebird, 2013.

[Eas18] EasyAR. Easyar sdk documentation, 2018. URL `https://www.easyar.com/doc/EasyAR%20SDK/EasyAR%20SDK.html`.

[Fou18a] Django Software Foundation. Django website, 2018. URL `https://www.djangoproject.com/`.

[Fou18b] Eclipse Foundation. Eclipse website, 2018. URL `https://www.eclipse.org/`.

[Fre15] A. Freeman. *Pro ASP.NET MVC 5*. Apress, 2015.

[GB18] Gk-Brown. Http-rpc documentation, 2018. URL `https://github.com/gk-brown/HTTP-RPC/blob/master/README.md`.

[GCP18] V. K. Ayyadevara G. Ciaburro and A. Perrier. *Hands-On Machine Learning on Google Cloud Platform*. Packt Publishing Ltd, 2018.

[Gmb18] Wikitude GmbH. Wikitude api documentation, 2018. URL `https://www.wikitude.com/documentation/`.

[Goo18a] Google. Arcore sdk developers, 2018. URL `https://developers.google.com/ar/develop/`.

[Goo18b] Google. Google cloud - natural language api documentation, 2018. URL `https://cloud.google.com/natural-language/docs/?hl=es`.

[Goo18c] Google. Google cloud - vision api, 2018. URL https://cloud.google.com/vision/docs/?hl=es.

[Gro14] Stanford Group. Stanford nlp website, 2014. URL https://nlp.stanford.edu/software/.

[Gro18] The PostgreSQL Global Development Group. Postgresql doc, 2018. URL https://www.postgresql.org/docs/.

[Gul13] Sunil Gulabani. *Developing RESTful Web Service with Jersey 2.0*. Packt Publishing Ltd, 2013.

[Hal16] S. Haldar. *SQLite Database System Design and Implementation*. S. Haldar, 2016.

[Han18] David Heinemeier Hansson. Ruby on rails website, 2018. URL https://rubyonrails.org/.

[Hol10] S. Holzner. *Django: Visual QuickPro Guide*. Peachpit Press, 2010.

[IBM18a] IBM. Db2 express-c website, 2018. URL https://www.ibm.com/developerworks/ssa/downloads/im/udbexp/index.html.

[IBM18b] IBM. Db2 website, 2018. URL https://www.ibm.com/analytics/us/en/db2/.

[IBM18c] IBM. Ibm - visual recognition api reference, 2018. URL https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/.

[IBM18d] IBM. Ibm - watson natural language understanding, 2018. URL https://www.ibm.com/watson/developercloud/natural-language-understanding/api/v1/.

[Inc18] Lucid Software Inc. Lucidchart website, 2018. URL https://www.lucidchart.com.

[JBo18] JBoss. Resteasy documentation, 2018. URL https://resteasy.github.io/docs.html.

[Jun14] J. Juneau. *JavaServer Faces: Introduction by Example*. Apress, 2014.

[KA17] G. Kurubacak and H. Altinpulluk. *Mobile Technologies and Augmented Reality in Open Education*. IGI Global, 2017.

[Kal09] Martin Kalin. *Java Web Services*. O'Really Media, Inc, 2009.

[Kal13] Martin Kalin. *Java Web Services: Up and Running*. O'Really Media, Inc, 2013.

[Ked09] S. Kedar. *Database Management Systems*. Pune, 2009.

[Ked15] S. Kedia. *How to learn Scrum in 60 minutes*. Subash Kedia, 2015.

[KK14] T. Kyte and D. Kuhn. *Expert Oracle Database Architecture*. Apress, 2014.

[Kud17] Kudan. Kudan documentation, 2017. URL https://kudan.readme.io/docs.

[Kur17] Mohamed Zakaria Kurdi. *Natural Lenguage Processing and Computational Linguistics 2*. Wiley, 2017.

[Lan18] Micheal Lanham. *Learn ARCore - Fundamentals of Google ARCore*. Packt Publishing Ltd, 2018.

[LB17] J. Linowes and K. Babilinski. *Augmented Reality for Developers*. Packt Publishing Ltd, 2017.

[LM18] Mark C. Layton and David Morrow. *Scrum For Dummies*. Wiley, 2018.

[Mai15] I. Maia. *Building Web Applications with Flask*. Packt Publising, 2015.

[MAL18] MALLET. Mallet website, 2018. URL `http://mallet.cs.umass.edu/`.

[Mar18] Francesco Marchioni. *Practical Java EE Development on WildFly*. ItBuzzPress, 2018.

[Max15] D. Maximini. *The Scrum Culture: Introducing Agile Methods in Organizations*. Springer, 2015.

[Max18] Maxst. Maxst documentation, 2018. URL `https://developer.maxst.com/MD/doc/3_5_x/intro`.

[Mic18a] Microsoft. Asp.net mvc website, 2018. URL `https://www.asp.net/mvc`.

[Mic18b] Microsoft. Microsoft sql server website, 2018. URL `https://www.microsoft.com/es-ES/sql-server/sql-server-2017`.

[MJ18] Don Maria McGreal and Ralph Jocham. *The Professional Product Owner: Leveraging Scrum as a Competitive Advantage*. Shwaber, 2018.

[MM02] Paul C. Zikopoulos Melnyk and Roman B. Melnyk. *Db2: The Complete Reference*. McGraw-Hill Education, 2002.

[MM16] Russ J. Martinelli and Dragan Z. Milosevic. *Project Management ToolBox*. Wiley, 2016.

[Mon18] Inc. MongoDB. Mongodb website, 2018. URL `https://docs.mongodb.com/`.

[Mor15] Alan Moran. *Managing Agile: Strategy, Implementation, Organisation and People*. Springer, 2015.

[Mul11] Tony Mullen. *Prototyping Augmented Reality*. Sybex, 2011.

[NLT09] NLTK. Natural language toolkit website, 2009. URL `https://www.nltk.org/`.

[Nod18] NodeJS. Nodejs documentation, 2018. URL `https://nodejs.org/es/docs/`.

[Nor15] NordicAPIs. Comparación rest vs soap, 2015. URL `https://nordicapis.com/rest-vs-soap-nordic-apis-infographic-comparison/`.

[Nsg16] Norituna-san and Japan Android Users group. Nyartoolkit for java documentation, 2016. URL `https://nyatla.jp/nyartoolkit/doc/nyartoolkit/5.0.8/api/`.

[OH12] R. Obe and L. Hsu. *PostgreSQL: Up and Running*. O'Reilly, 2012.

[Ora18a] Oracle. Javaserver faces website, 2018. URL http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html.

[Ora18b] Oracle. Mysql website, 2018. URL https://dev.mysql.com/doc/.

[Ora18c] Oracle. Oracle database website, 2018. URL https://www.oracle.com/es/database/index.html.

[Otw18] Taylor Otwell. Laravel website doc., 2018. URL https://laravel.com/docs/5.6.

[Par18] ParallelDots. Paralleldots nlp documentation, 2018. URL https://www.paralleldots.com/text-analysis-apis.

[PCZS10] George Baklarz Paul C. Zikopoulos and Dan Scott. *Apache Derby - Off to the Races*. Pearson Education, 2010.

[PII15] P. Kommers P. Isaías and T. Issa. *The Evolution of the Internet in the Business Sector: Web 1.0 to Web 3.0*. IGI Global, 2015.

[Pow15] PowerData. Tipos y función de los gestores de bases de datos, 2015. URL https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/406547/tipos-y-funci-n-de-los-gestores-de-bases-de-datos.

[Pro17] Apache DB Project. Apache derby website, 2017. URL http://db.apache.org/derby/.

[Ree15] Richard M. Reese. *Natural Lenguage Processing with Java*. Packt Publishing Ltd, 2015.

[Res18] Restlet. Restlet documentation, 2018. URL https://restlet.com/documentation/.

[RGS13] R. Stackowiak R. Greenwald and J. Stern. *Oracle Essentials: Oracle Database 12c*. O'Reilly, 2013.

[Roc17] Julio Roche. Ceremonias scrum - deloitte, 2017. URL https://www2.deloitte.com/es/es/pages/technology/articles/ceremonias-scrum.html.

[Ron18] Armin Ronacher. Flask website, 2018. URL http://flask.pocoo.org/docs/1.0/.

[RT00] Fielding RT. *Architectural Styles and the Design of Network-based Software Architectures*. 2000. URL http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[RT13] Nader K. Rad and Frank Turley. *The Scrum Master Training Manual: A Guide to the PSM Exam*. MP, 2013.

[Rub12] Kenneth S. Rubin. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley, 2012.

[Sad13] P. Saddington. *The Agile Pocket Guide: A Quick Start to Making Your Business Agile Using Scrum and Beyond*. Wiley, 2013.

[San09] Jose Sandoval. *RESTful Java Web Services*. Packt Publishing Ltd, 2009.

[Sch12] James Schiel. *The ScrumMaster Study Guide*. CRC Press, 2012.

[Spr18] Spring. Spring mvc website, 2018. URL https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html.

[Sta17] M. Stauffer. *Laravel: Up and Running: A Framework for Building Modern PHP Apps*. O'Reilly, 2017.

[SVS17] M. Tantawi S. Vergara, M. El-Khouly and L. Sri. *Building Cognitive Applications with IBM Watson Services: Volume 7 Natural Language Understanding*. Redbooks, 2017.

[Tes18] Tesseract. Tesseract project website, 2018. URL https://github.com/tesseract-ocr.

[TIO18] TIOBE. Tiobe, 2018. URL https://www.tiobe.com/tiobe-index/.

[Tre18] Trello. Trello website, 2018. URL https://trello.com/.

[vdZ18] Benito van der Zander. Texstudio website, 2018. URL https://www.texstudio.org/.

[Vir18a] ViroAR. Virocore documentation, 2018. URL https://virocore.viromedia.com/.

[Vir18b] ViroAR. Viroreact documentation, 2018. URL https://docs.viromedia.com/.

[Voh14] D. Vohra. *JavaServer Faces 2.0: Essential Guide for Developers*. Cengage Learning, 2014.

[Vuf18] Vuforia. Vuforia documentation, 2018. URL https://library.vuforia.com/.

[War15] G. Warin. *Mastering Spring MVC 4*. Packt Publishing, 2015.

[Wik18] Wikipedia. Programming languages used in most popular websites, 2018. URL https://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites.

[Wil07] J. Williams. *Rails Solutions: Ruby on Rails Made Easy*. friendsof, 2007.

[Wol18] D. Wolfensparger. *Apple ARKit Revealed: Augmented and Mixed Reality for iPhone and iPad*. Apress, 2018.

[Yel17] Naren Yellavula. *Building RESTful Web Services with Go*. Packt Publishing Ltd, 2017.

Este documento fue editado y tipografiado con LaTeX empleando
la clase **esi-tfg** (versión 0.20180629) que se puede encontrar en:
`https://bitbucket.org/arco_group/esi-tfg`