UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR

DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN

MENCIÓN EN TELEMÁTICA

DISEÑO Y DESARROLLO DE UNA APLICACIÓN ANDROID PARA EL RECONOCIMIENTO ÓPTICO DE CARACTERES

AUTOR:
D. Alfonso Jiménez Hernández
TUTOR:
D. Javier Manuel Aguiar Pérez

Valladolid, 6 de Septiembre de 2018

TÍTULO: DISEÑO Y DESARROLLO DE UNA APLICACIÓN ANDROID PARA RECONOCIMIENTO ÓPTICO DE CARACTERES

AUTOR: ALFONSO JIMÉNEZ HERNÁNDEZ

TUTOR: JAVIER MANUEL AGUIAR PÉREZ

DEPARTAMENTO: TEORÍA DE LA SEÑAL Y COMUNICACIONES E INGENIERÍA TELEMÁTICA

Miembros del Tribunal

PRESIDENTE: MARÍA ÁNGELES PÉREZ JUÁREZ

SECRETARIO: JAVIER MANUEL AGUIAR PÉREZ

VOCAL: JAIME GÓMEZ GIL

SUPLENTE 1: DAVID GONZÁLEZ ORTEGA

SUPLENTE 2: MARIO MARTÍNEZ ZARZUELA

FECHA DE LECTURA: SEPTIEMBRE 2018

CALIFICACIÓN:

RESUMEN DEL PROYECTO

El presente Trabajo Fin de Grado pretende llevar a cabo el diseño e implementación de un sistema mediante la tecnología OCR (*Optical Character Recognition*), que sea capaz de detectar e interpretar la información textual incluida en las imágenes realizadas desde un dispositivo móvil Android. Dicha información será contenida en un fichero, que puede ser alojado tanto en un servidor como en el propio dispositivo móvil, para su posterior análisis y utilización.

PALABRAS CLAVE

Android, aplicación, Java, Android Studio, reconocimiento, OCR.

ABSTRACT

This final degree project aims to design and carry out the implementation of a system using OCR (Optical Character Recognition) technology that is capable of detecting and interpreting the textual information included in the images made from an Android mobile device.
The said information will be contained in a file which can be hosted either on a server or can be stored on the mobile device itself for later analysis and use.

KEY WORDS

Android, application, Java, Android Studio, Recognition, OCR.

A mi familia, por todo su apoyo, tanto en los buenos como en los malos momentos, y por hacer posible que haya llegado hasta aquí.

Índice de contenido

Índice de ilustraciones

Índice de tablas

CAPITULO 1

1. INTRODUCIÓN

En la actualidad cada vez son más las personas que tienen el sentimiento de que un teléfono inteligente o "*smartphone*" es indispensable en su vida. La dependencia hacia estos dispositivos se fundamenta en los diversos usos que se les da a los mismos, en particular, aquellos destinados al ocio, al trabajo, y a la consulta de información.

Es importante considerar el 29 de junio de 2007 como fecha clave en la cual se lanzó al mercado la primera generación de iPhone, revolucionando el área y la percepción de los *smartphones* existentes hasta el momento, cuya evolución prosiguió en los años posteriores a través de los diferentes *smartphones* que utilizaban Android.

En otras palabras, el uso de los *smartphones* ha llevado consigo un desarrollo exponencial en lo que respecta a sus características mediante la búsqueda de pantallas de gran calidad que ocupasen todo el dispositivo con la finalidad de que sirviesen no sólo para gestionar la información que disponemos sino también para mostrarla gracias a un mayor desarrollo en la versatilidad y funcionalidad de la cámara.

No obstante, la evolución de los *smartphones* no hubiera sido posible sin la ayuda de la mejora constante de las tecnologías de acceso inalámbrico y móvil, en particular, a partir del UMTS (*Universal Mobile Telecommunications System*), tecnología móvil de la llamada tercera generación (3G) que permite disponer de una mayor capacidad de voz y datos, ofreciendo una alta transmisión de datos a un bajo coste.

La necesidad de tener acceso a Internet y la inclusión de los *smartphones* en nuestro día a día se ha reflejado en que en muy poco tiempo la cantidad de datos que son y han sido consumidos y creados han aumentado progresivamente. A parte de estos factores también hay que añadir las variaciones en los precios tanto en los dispositivos como en las tarifas de acceso a voz y datos.

Millones de personas están conectadas por *smartphones*, con una facilidad sin precedentes de acceso a la información y con capacidades enormes de almacenamiento y procesamiento de datos a costes reducidos que pueden ser ilimitados. Según las cifras proporcionadas por el *International Data Corporation* (IDC), el tamaño de datos alcanzará en 2020 los 44 ZB mientras que en 2013 se disponía de un volumen de datos de 4.4 ZB [1]. Por consiguiente, el progreso del volumen de datos seguirá ampliándose no sólo con personas y *smartphones* sino también con todo el dispositivo conectado a Internet (IoT, *Internet of Things*). Así pues, esta situación ha provocado una nueva serie

de oportunidades, entre ellas el *Big Data*, pero para manejarlo técnicamente es preciso su recolección, transporte y almacenamiento para un objetivo final de análisis de datos.

En lo que respecta a la extracción de datos de texto, se utiliza el Reconocimiento Óptimo de Caracteres (en adelante, OCR, *Optical Character Recognition*) que permite convertir diferentes tipos de texto capturados mediante formato PDF *(Portable Document Format*) o imágenes captadas por una cámara en datos que puedan ser reutilizados. De manera que, el OCR ha tenido un gran avance debido a las mejoras en prestaciones en los dispositivos y a la metodología empleada para el reconocimiento de caracteres de forma más eficiente.

Hechas las consideraciones anteriores, el problema que se aborda en la aplicación objeto del presente trabajo, no es solo conseguir información sino examinar la capacidad de extraer información de una imagen con el fin de posibilitar el análisis y la extracción de información útil que proporcione un beneficio en nuestro sector. Por esta razón se otorga necesaria la creación de una herramienta que se base en la extracción de datos previamente seleccionados de la cámara del propio dispositivo *smartphone*.

1.1 OBJETIVOS

El objetivo principal del presente trabajo es el desarrollo de una aplicación que sea capaz de recabar datos de una imagen tanto en tiempo real como previamente capturada para que éstos puedan ser mostrados de una forma sencilla y concisa. Igualmente se pretende dar un posible respaldo a los datos obtenidos y reducir el error de texto mediante una captura de imagen de alta definición.

La aplicación podrá ser utilizada por cualquier tipo de usuario que posea un sistema operativo basado en Android que posea conexión a Internet para tener un completo acceso a todas las funcionalidades de la aplicación. Del mismo modo, la interfaz seguirá una serie de pautas como examinaremos posteriormente con el fin de que tenga todas las funcionalidades de cualquier aplicación que se encuentre en el mercado.

Puesto que la aplicación no debe presentar problemas en ningún dispositivo móvil, será ofrecida a partir de la versión Android 4.0.3 y contará con la funcionalidad adecuada prevista para su empleo diario.

A los objetivos anteriormente mencionados hay que añadir una serie de características y metas secundarias de la aplicación que se detallan a continuación:

En primer lugar, la aplicación debe ser precisa en cuanto a tiempo de ejecución, en vista de que los algoritmos de detección pueden conllevar un alto nivel computacional.

Además, tiene que posibilitar a los usuarios gestionar sus capturas y los datos adquiridos de una manera gráfica.

En segundo lugar, debe facilitar a los usuarios a recibir notificaciones cuando la aplicación esté dispuesta para capturar los datos que se deseen. Asimismo, debe permitir dar la opción de mantener sus datos e imágenes obtenidas para un posterior uso mediante un servidor.

En tercer lugar, debe poder definir la estructura de la imagen para obtener los datos una vez capturada.

Por último, debe ser capaz de ofrecer diferentes métodos para la captura de imágenes y así como proporcionar posteriormente su visualización de manera gráfica.

En resumidas cuentas, con vistas a alcanzar los diferentes objetivos propuestos es preciso un estudio de los diferentes algoritmos de reconocimiento actuales y del estado del arte con relación al reconocimiento y detección de texto a fin de poder elegir en un momento posterior aquel que sea capaz de cumplir las diferentes características de este apartado.

1.2 ESTRUCTURA DEL DOCUMENTO

Para una mayor comprensión de los diferentes temas que se tratan en este documento se ha decidido estructurarlo en seis capítulos que constan de los siguientes contenidos:

- El primer capítulo consta de una introducción sobre el contexto actual en el que se va a desarrollar el proyecto, los objetivos y la estructura del documento.

- El segundo capítulo se especifican las diferentes tecnologías que existen y podrían ser empleadas para el desarrollo de la aplicación. Una vez escogido el tipo tecnología a usar se describen detalla con más profundidad sobre los diferentes tipos incluyendo las ventajas e inconvenientes, finalmente se analiza cuál utilizar y por qué.

- El tercer capítulo consta de una introducción a la tecnología OCR, su historia y su funcionamiento. Después se detalla el estado del arte, las diferentes bibliotecas OCR que son utilizadas en la actualidad y, por último, la justificación sobre la biblioteca utilizada.

- El cuarto capítulo concierne al manual del programador describiendo el programa utilizado para impulsar el proyecto, las tendencias de diseño

utilizadas, las diferentes bibliotecas que han sido empleadas y las consideraciones para tener en cuenta a la hora de fomentar una aplicación con los objetivos previos.

- El quinto capítulo está dedicado a la explicación del manual de usuario a través de casos concretos de uso que puntualizan el manejo de la aplicación y su interfaz.

- El sexto y último capítulo, explica las diferentes conclusiones obtenidas tras la elaboración del proyecto, así como las líneas futuras para mejorar la aplicación.

CAPITULO 2

2 ESTUDIO DE LAS DISTINTAS TECNOLOGÍAS

Resulta necesario conocer todas las alternativas cuando se ha de iniciar un desarrollo de una aplicación móvil. A través de este capítulo se van a analizar las diferentes tecnologías operantes en el mercado, sus diferentes ventajas e inconvenientes y la justificación de las elecciones tomadas.

En el año 2016 los usuarios de móvil superaron la mitad de la población mundial. Esta popularidad creciente se ve reflejada en la creación y uso de aplicaciones móviles, tal y como se aprecia de la ilustración 1 expuesta a continuación. La misma recoge la cifra de 3,8 millones de aplicaciones en Google Play en marzo de 2018.

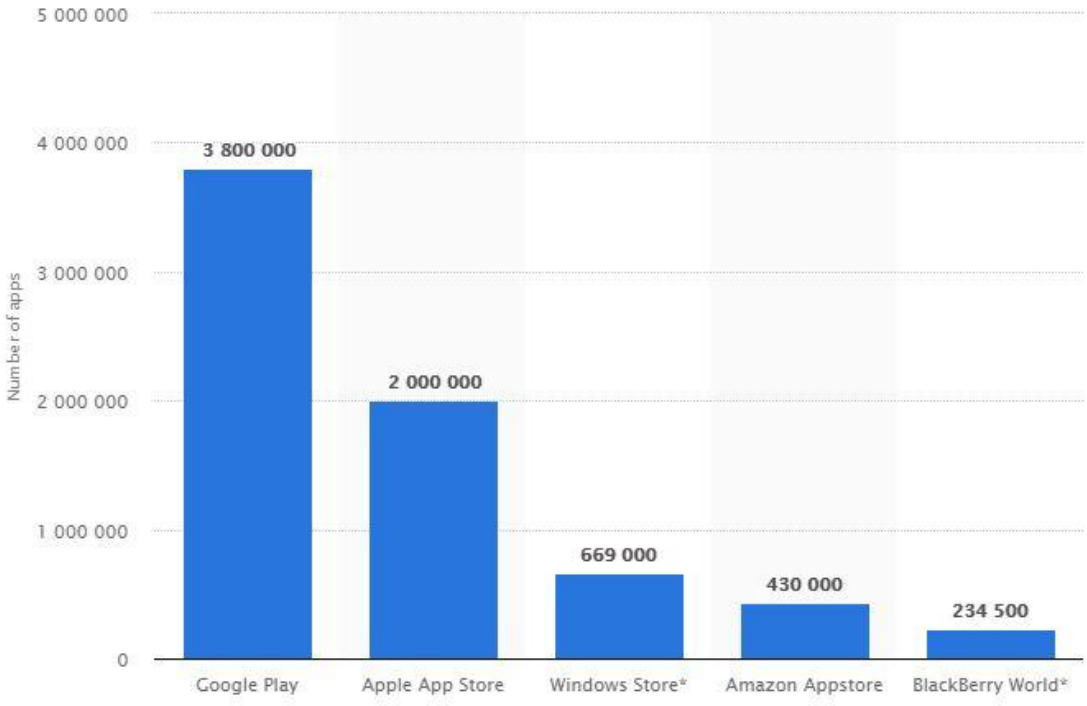

Ilustración 1 – Aplicaciones actuales en las diferentes tiendas de apps marzo 2018

Avanzando con el tema, el desarrollo de las aplicaciones para dispositivos móviles puede ser llevado a cabo ya sea bien mediante aplicaciones Web o bien mediante aplicaciones híbridas y/o nativas que otorgan diferentes soluciones para un mismo problema.

2.1 Aplicaciones Web

Entre las principales ventajas de que disponen las webs móviles, citar la fácil implementación, la sencillez de testeo y actualización, y la ausencia de exigencia de instalación. De hecho, se puede realizar gran parte del desarrollo sin necesidad de utilizar dispositivos móviles ni emuladores hasta llegar a las fases finales del desarrollo. Es más, pueden soportar múltiples dispositivos con un único código fuente [2].

En sentido opuesto, cuentan con el inconveniente de ser menos llamativas que las aplicaciones nativas, y de no tener acceso a todos los elementos del *smartphone*, razones por las cuales descartamos esta opción de desarrollo.

2.2 Aplicaciones Hibridas

Las aplicaciones móviles híbridas se desarrollan con una combinación de tecnologías web como HTML, CSS y JavaScript, por tanto, no son aplicaciones móviles verdaderamente nativas, ya que consisten en un *WebView*, y son ejecutadas dentro de un contenedor nativo, en consecuencia, tampoco están basadas en Web, porque se empaquetan como aplicaciones para distribución y tienen acceso a los diferentes elementos del dispositivo al igual que las aplicaciones nativas.

Como puntos a su favor subrayar su desarrollo unificado la hora de servir para diversas plataformas móviles y el ahorro que suponen para las empresas. Sin embargo, en comparación con las aplicaciones nativas cuentan con dificultades tales como la posibilidad de pérdida de rendimiento, el mayor espacio que requieren para el almacenamiento, la experiencia del usuario y el tiempo de respuesta al acceder a los diferentes elementos del dispositivo.

2.3 Aplicaciones nativas

El concepto de aplicaciones nativas puede entenderse como aquellas creadas a partir de un sistema operativo (SO) determinado que cada plataforma pone a disposición de la comunidad de desarrolladores. Es por eso que desarrollar una aplicación dedicada para cada plataforma donde se quiera incorporar siempre será la opción que mejores resultados dará en términos de optimización y resultado final, pero ello supone multiplicar la carga de trabajo por cada nuevo SO para el que se quiera adaptar [4].

Este tipo de aplicaciones poseen un rendimiento mucho más rápido y acceso directo a los servicios que brinda el dispositivo (acelerómetro, GPS, cámara, etc.). No es necesario el acceso al Internet y pueden notificar al usuario cuando ocurra un evento mientras

están en un segundo plano. Frente a estas ventajas existe un mayor tiempo de desarrollo ya que se necesita un lenguaje de programación diferente de cada plataforma.

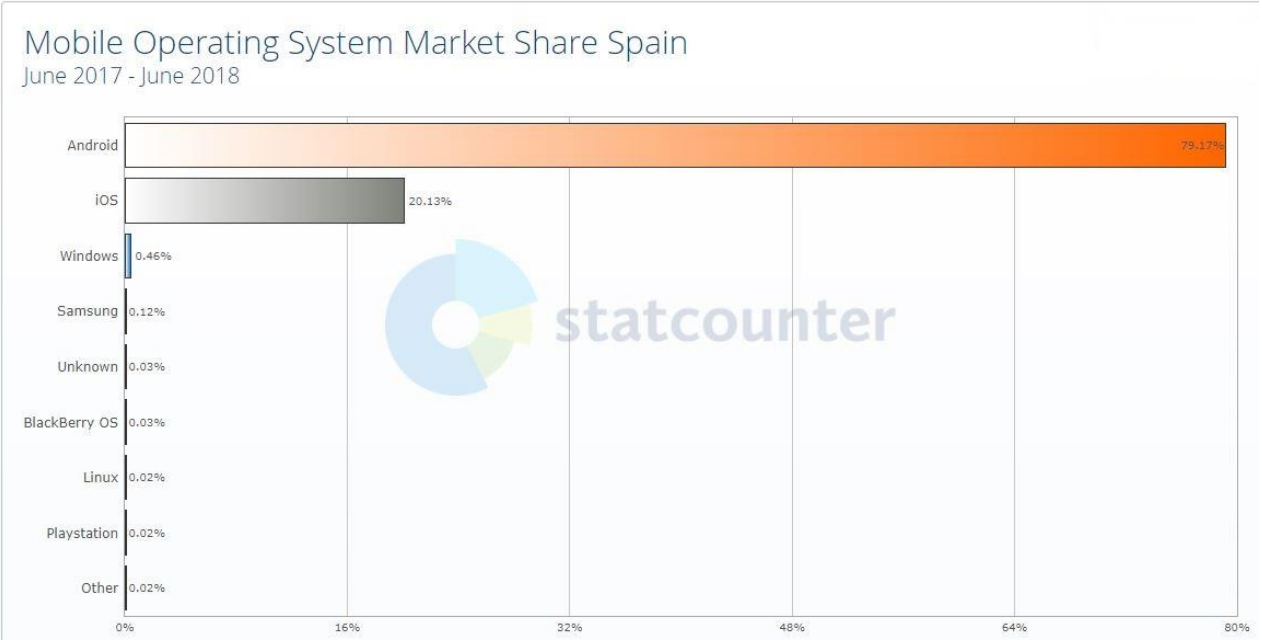

Ilustración 2: Cuota de mercado en España por Sistema Operativo desde Junio 2017 a Junio 2018 [3]

Existen diferentes sistemas operativos usados en los teléfonos actuales, pero solo hay dos que copan el mercado en los últimos tiempos: Android e iOS. Por ello, analizaremos los pros y contras que existen entre cada uno ellos.

2.3.1 iOS

En 2007 se lanzó el primer iPhone y, por consiguiente, la primera versión de lo que más tarde sería iOS. Seguidamente se lanzó el kit de desarrollo de software (SDK) en febrero de 2008 con el nombre de IPhone iOS. Es más, en junio de 2010 se renombró con el nombre de iOS que está basado en el SO MAC OS X, fundamentado a su vez en Darwin BSD, derivado de Unix. iOS no ha parado de crecer y cambiar hasta la actualidad. Con el lanzamiento de iOS 10 en junio de 2016 se liberaron sus diversas aplicaciones directas del sistema como por ejemplo Siri [4].

Las aplicaciones móviles para iOS se desarrollan Objetive-C hasta junio 2014 con la llegada de Swift, un lenguaje de programación 2,6 veces más rápido que su predecesor, integrándose a la perfección con código ya escrito.

2.3.1.1 Arquitectura de iOS

La arquitectura de iOS se divide en 4 capas como se observa en la ilustración 3.

- Cocoa Touch: es la capa superior y la más importante para el desarrollo de aplicaciones nativas en iOS, donde se encuentra la interfaz de usuario. Proporciona acceso a las funciones principales del sistema como contactos, eventos *multi-touch*, cámara etc.

- Media: Donde se encuentran los servicios gráficos y multimedia de la capa superior.

- Core Services: Da acceso a los recursos fundamentales necesarios para la aplicación.

- Core OS: Características a bajo nivel: seguridad, drivers del dispositivo, etc.

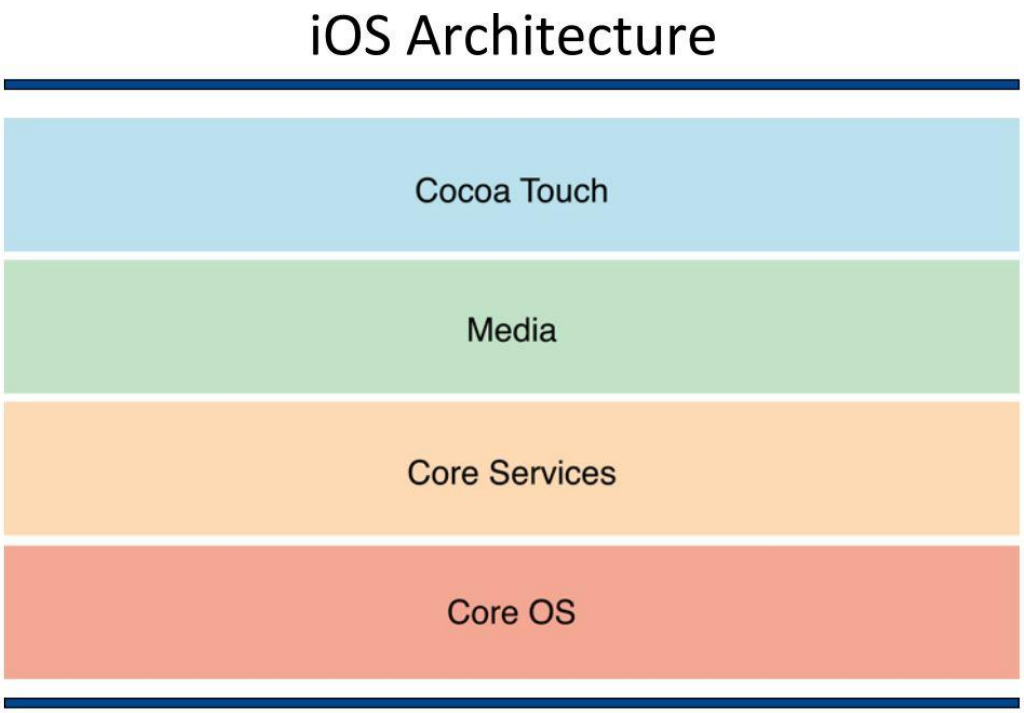

Ilustración 3: Arquitectura de iOS [5]

2.3.2 Sistema Operativo Android

En la presente sección se va a contextualizar el escenario en el que Google adquirió Android Inc. Remontándonos a los orígenes de Android, en el año 2005 dicha compañía recién creada estaba orientada a la creación de aplicaciones para terminales móviles. Seguidamente, en noviembre de 2007 se lanzó una primera versión del Android SDK y al año siguiente apareció el primer móvil Android (T-Mobile G1). Durante el año 2010, Android se consolidó en el mercado como uno de los SO más utilizados. Así pues, en 2013 con las versiones 4.4. (*Kit kat*) Android alcanzó una cuota de mercado cercada al 90% [6].

El lenguaje oficial para desarrollar aplicaciones Android era Java hasta el evento de Google I/O del año pasado, donde declararon Kotlin como lenguaje oficial en Android al mismo nivel que su predecesor. Es necesario subrayar que, a pesar de ser desconocido por la mayoría de la población, Android también se puede programar mediante C o C++, a través de unas herramientas llamadas NDK (*Native Development Kit)* que pueden incluso comunicarse con una parte de la aplicación que haya sido desarrollada en Java [7].

2.3.2.1 Arquitectura de Android

La arquitectura Android está formada por cuatro capas como se puede apreciar en la ilustración 4. Dicho esto, las capas superiores basan su funcionamiento en capas inferiores. Una de las características más importantes es que todas las capas están basadas en software libre y gratuito.

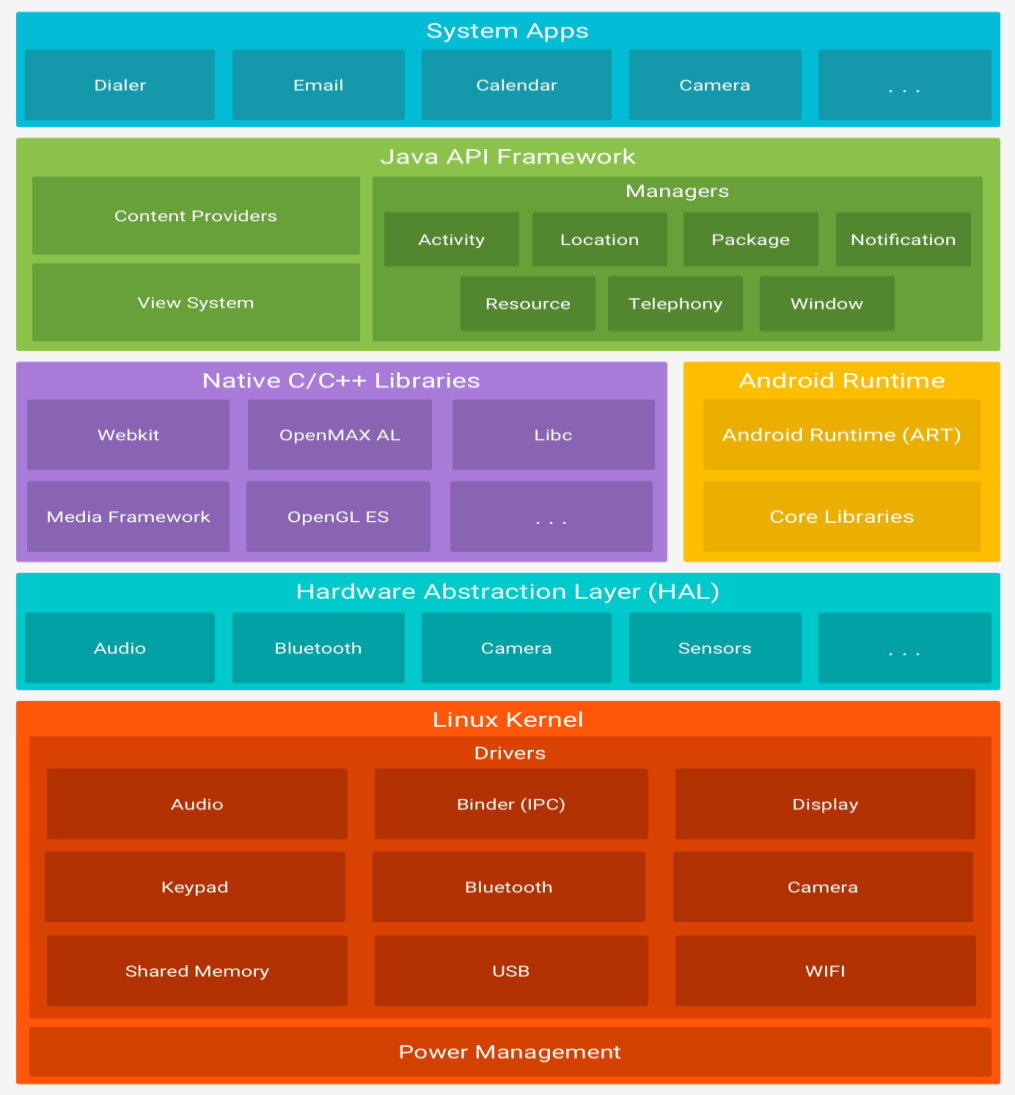

Ilustración 4: Arquitectura del sistema operativo Android [8]

- Aplicaciones el sistema: Grupo de aplicaciones que están instaladas en el sistema operativo por defecto, como aquellas que el propietario del dispositivo instale después. Las aplicaciones harán uso de los gestores que proporciona la capa inferior llamada *Application framework.*

- *Application framework:* Proporciona un conjunto de funciones desarrolladas en el lenguaje de programación Java. Sirve para facilitar la reutilización de componentes y servicios centrales y modulares. También permite la posibilidad de reemplazar los diferentes componentes y servicios por el usuario.

- Bibliotecas: Android contiene bibliotecas nativas escritas en C y C++ ya que muchos servicios centrales como ART (Android *Run Time*) y HAL (*Hardware Abstraction Layer)* los requieren.

- *Android Runtime*: En el mismo nivel de las bibliotecas encontramos Android *RunTime* compuesto por ART *virtual machine* y las bibliotecas de las que se compone. Anteriormente estaba compuesto por la máquina virtual Dalvik pero a partir de la versión Android 4.4 *KitKat* aparece ART que presenta mejoras sustanciales en cuanto al rendimiento, el tiempo de ejecución y compilación.

- Capa de abstracción de Hardware (HAL): radica en varios módulos de biblioteca y cada uno de estos implementa una interfaz de componente hardware para que cuando la capa de *Java API (Application Programming Interfaces) framework* realice una llamada a los distintos componentes hardware del dispositivo, se cargue el módulo de biblioteca especifico.

- *Kernel* de Linux: Por el último, en la capa de más bajo nivel encontramos el *kernel* (núcleo) que admite que Android se beneficie de funciones de seguridad claves y permite a fabricantes desarrollar controladores de hardware para un *kernel* ya conocido.

2.4 Android frente a iOS

El proceso de iOS para la publicación de las aplicaciones en la Tienda *App Store* permite que estas aplicaciones sean más seguras que las de Android debido a que estas tienen que pasar diferentes filtrados hasta su publicación. Es posible que debido a no pasar diferentes filtrados haya algunas aplicaciones para dispositivos Android que no funcionen. Por otro lado, los desarrolladores de aplicaciones en Android deben pagar una cuota de 25$ para publicar una aplicación, pero solamente una vez mientras que los desarrolladores de iOS deben pagar una cuota anual de 99$.

A pesar de que Android e iOS son los SO que acaparan el mercado, Android cuenta a su favor con abundantes tipos de dispositivos en el mercado actual, con características muy diferentes entre unos y otros que van desde terminales básicos hasta los terminales más costosos y complejos. Debido a esta variedad, este sistema operativo copa alrededor del 80% del mercado de todo el mundo.

El sistema operativo de iOS basado en MAC OS X, es cerrado y no existe la posibilidad de cambios por parte de ningún tipo de usuario. Esta idea se contrapone con las características de la arquitectura *kernel* de Android respaldadas por Linux, es decir, un sistema abierto y libre para todo usuario.

Mientras que IOS es un referente en cuanto a diseño desde que saliera el primer IPhone en 2007, su interfaz es pobre y no permite grandes cambios, lo cual no sucede con la personalización que identifica a Android.

Para terminar, el lenguaje de programación de Android (Java) está más extendido que los lenguajes de iOS (Objetive-c y Swift).

2.5 Conclusiones

El primer elemento que considerar es la elección del tipo de aplicación. Como se ha expuesto en los diferentes tipos de aplicaciones, el uso de elementos hardware del dispositivo (en caso de nuestro proyecto la cámara) puede desencadenar en problemas de rendimiento. Por ello, para evitar estos inconvenientes elegimos el desarrollo nativo. Un ejemplo sería el plugin de cámara de React Native que tiene 12 propiedades y 8 métodos mientras que en Android soporta decenas de clases y cientos de métodos y propiedades, además que no necesitan *plugins* que son desarrollados posteriormente tras alguna actualización por parte del sistema operativo, dando una clara ventaja a Android.

También el uso de aplicaciones nativas proporciona el desarrollo para múltiples dispositivos tanto en iOS como en Android (televisiones, coches, Apple *watch* etc.).

Dentro del desarrollo nativo, el sistema operativo elegido es Android por su característica de ser un SO libre, es decir, que a la hora de desarrollar este sistema operativo e incluirlo en un dispositivo móvil no hay que realizar ningún desembolso.

Como se ha visto al inicio del documento, Android copa la mayor parte del mercado. Este motivo es otro punto a favor a la hora de elegir Android en contra de iOS, ya que la aplicación puede llegar a mayor número de usuarios.

La versión elegida para Android será la versión *KitKat* (versión 4.4) para ser capaz de utilizar las características de diseño actuales propias de cualquier aplicación. De manera que a fin de poder ser viable en el 100% de los dispositivos se añadirá la posibilidad de que la aplicación pueda ser admitida a partir de la versión *Ice Cream Sandwich* (versión 4.0.3).

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.9%
4.3		18	0.5%
4.4	KitKat	19	9.1%
5.0	Lollipop	21	4.2%
5.1		22	16.2%
6.0	Marshmallow	23	23.5%
7.0	Nougat	24	21.2%
7.1		25	9.6%
8.0	Oreo	26	10.1%
8.1		27	2.0%

Ilustración 5: Uso de versiones Android, 23/7/2018 [8]

Finalmente, otro factor para tener en cuenta a favor de Android es la existencia de diferentes bibliotecas en cuanto a la tecnología OCR mientras que en iOS es cuantitativamente menor.

CAPITULO 3

3 OCR Optical Character Recognition

3.1 Introducción

Una definición de lo que entendemos por OCR viene aclarada por Jesús Tomás en "*Dispositivos Wearables, Visión Artificial, Google Glass y Android TV* ", entendiendo por tal*: "Los sistemas que, a partir de un texto escrito o impreso en papel o similar, crean un fichero de texto en un soporte de almacenamiento informático" [9].*

3.2 Funcionamiento OCR

El funcionamiento teórico de OCR se diferencia en tres fases generales:

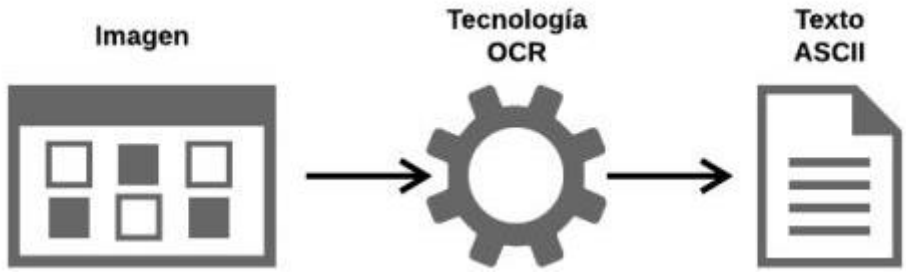

Ilustración 6: fases del funcionamiento OCR.

- Primera Fase: Se trata de la fase del pre-procesado de la imagen. En este primer momento se captura la imagen para su posterior procesamiento siendo necesario una fase de rotación si las imágenes no se encuentran en modo vertical. Se genera un Mapa de bits para la fase de detección de caracteres si existen en la imagen.

- Segunda Fase: En esta fase comienza el uso de la tecnología OCR. Existen diferentes métodos siendo los más utilizados *Pattern Matching*, que es una comparación pixel a pixel con un patrón previo, normalmente usado para textos escritos mediante fuentes conocidas no escritas por las personas y el modo *Feature detection,* que busca descomponer la imagen en caracteres para sacar unas características con la finalidad de saber de qué carácter se trata. Este proceso se utiliza en textos escritos a mano.

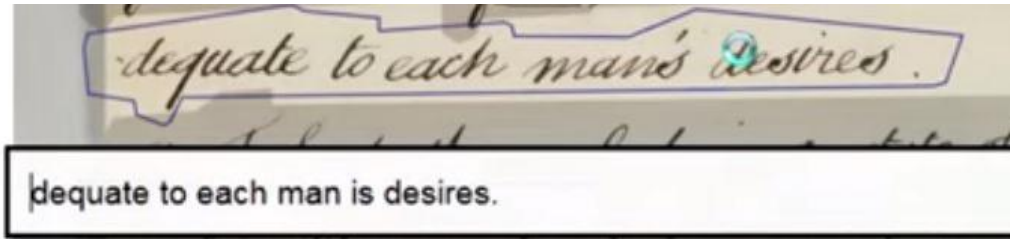

Ilustración 7: modo de detección de caracteres método Feature detection.

- Tercera fase: Depende de post-procesado según las necesidades de la aplicación, normalmente se genera un fichero texto o un fichero PDF formado por la unión de caracteres reconocidos.

En el año 1929 Gustav Tauschek [10] realizó una patente relacionada con el OCR que podía leer caracteres y números. Este algoritmo ha sido implementado desde hace años y cada día son más aplicaciones que utilizan este tipo de tecnología para mejorar su rendimiento, de ahí que hayamos seleccionado las aplicaciones que actualmente más se usan en el mercado de aplicaciones móviles.

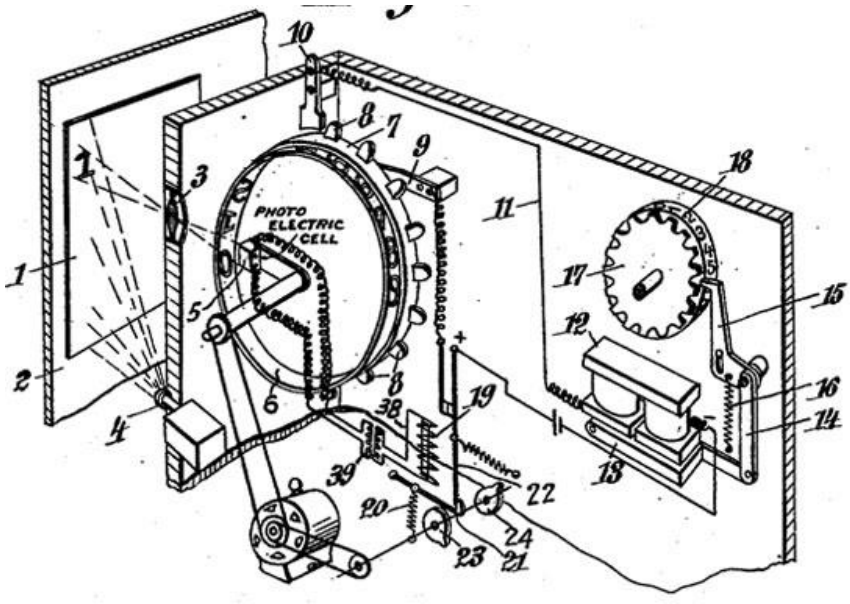

Ilustración 8: patente de Gustav Tauschek (1929) primer dispositivo OCR

3.3 Estado del arte

Han surgido varias empresas que se dedican a crear software y algunas de éstas han desarrollado aplicaciones e incluso sus propios algoritmos:

3.3.1 ABBYY FineReader

Ilustración 9: logo de Abby finereader

ABBYY Fine Reader 14 es una aplicación que tiene su propio algoritmo OCR producido por la empresa ABBYY, la cual da servicios de reconocimiento de documentos y captura de datos. Lanzó su primera versión hace más de 15 años. Su función principal es convertir imágenes en archivos editables. También soporta el formato *E-book* (E pub) utilizado en los libros electrónicos y reconoce documentos en 190 lenguas.

Su desventaja más significativa radica en que es un programa que requiere de pago previo en cuotas mensuales o anuales. Por consiguiente, según el desembolso realizado se tiene acceso a unas funcionalidades u otras, aunque se puede considerar como un precio muy elevado para este tipo de aplicaciones donde existe una gran variedad y donde no hay mucha diferencia.

Ilustración 10: Pantalla principal de ABBYY Fine Reader 14 tras captura.

3.3.2 Office Lens

Ilustración 11: logo de Office Lens

Office Lens es una aplicación desarrollada por Office bastante sencilla. En pocas palabras, facilita tomar fotos de pizarras o tarjetas de visita y convertirlas en texto editable. La principal ventaja con respecto al resto de aplicaciones es que se puede obtener beneficio de las aplicaciones de Office, como por ejemplo OneDrive para guardar la información o OneNote para editar la información. Es una aplicación sencilla y gratuita, pero en cuenta con unas opciones limitadas de edición ya que es un complemento de la cuenta Office, es decir, si no se adquiere el paquete, no se puede

acceder a sus diferentes usos. Además, no tiene soporte para otros idiomas que no sea el inglés.

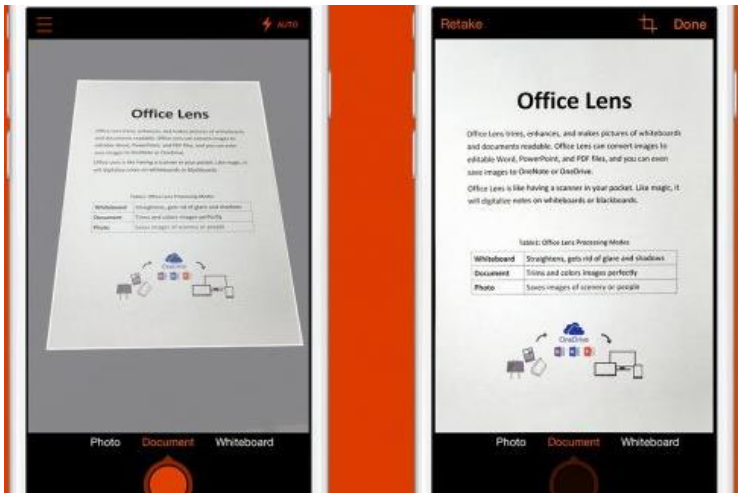

Ilustración 12: Pantalla principal de Office Lens después de realizar una Captura

3.3.3 CamScanner

lustración 13: logo de CamScanner

CamScanner es la aplicación más descargada en Play Store relacionada con técnicas de OCR desarrollado por INTSIG, un proveedor de aplicaciones móviles que ofrece aplicaciones simples pero efectivas en varios teléfonos móviles y plataformas (iOS / Android / BlackBerry / Windows Phone).

El resultado es similar a otras aplicaciones como Office Lens, pero añade un apartado de edición mucho mayor. La diferencia fundamental con Office Lens radica en la admisión de varios idiomas. Su principal contra es la presencia de publicidad incómoda que se utiliza en el modelo gratuito. En esta versión gratuita, se añade también una marca de agua (pancarta en realidad) que se coloca en la parte inferior de la página escaneada.

Además, no permite el acceso a todos los servicios que ofrecen los modelos premium de esta aplicación.

Ilustración 14: Pantalla principal de Camscanner después de realizar una Captura

3.3.4 Text Fairy

Ilustración 15: Logo de textFairy

TextFairy es una aplicación muy descargada en la *Play store* en Android, desarrollada por Renard Wellnitz, creador de aplicaciones como Zalando o Redmart Ecommerce. Es muy parecida a todas las aplicaciones de este tipo de tecnología, ya que su función principal es la conversión de imágenes escaneadas en formato PDF, pero el procesado de las mismas se realiza mediante una biblioteca en XCode. Reconoce más de 50 idiomas, pero tiene bastantes problemas a la hora de detectar gran cantidad de

caracteres, siendo este motivo por el que tiene peor valoración que las aplicaciones anteriores. Su punto a favor es que es una aplicación gratuita, sin anuncios.

Ilustración 16: Captura de una imagen y selección de texto mediante aplicación TextFairy

3.3.5 Conclusiones sobre el estado del arte

Como se ha manifestado, existen en la actualidad varias aplicaciones en el mercado que realizan las partes de pre-procesado, de forma prácticamente similar sin entrar en técnicas de edición de imagen al necesitar la tecnología OCR en su primera fase una imagen para el preprocesamiento. Además, en su última etapa todas estas aplicaciones muestran el texto reconocido en formato PDF o formato texto.

Es por ello, donde la aplicación desarrollada en este proyecto sale de los esquemas actuales del mercado e incorpora entre algunas novedades, otro formato de texto editable, como veremos posteriormente.

Cabe destacar también que estas aplicaciones procesan todos los caracteres del documento y en este proyecto también se incluye un algoritmo de pre-detección para no mostrar los caracteres no necesarios, extrayendo los datos importantes para su posterior uso.

Para desarrollar estas propiedades con respecto a otras aplicaciones necesitamos utilizar las diferentes bibliotecas que existen en torno a OCR.

3.4 Bibliotecas OCR

Existen varias bibliotecas que han sido desarrolladas para el uso de OCR, pero las más importantes son Tessereact [11] y Google Vision API.

3.4.1 Tessereact

Tessereact: La biblioteca Tesseract es probablemente el primer motor de OCR capaz de manejar texto blanco sobre negro de forma tan trivial. Fue desarrollado originalmente en Hewlett-Packard Laboratories Bristol y en Hewlett-Packard Co, en Colorado entre 1984 y 1994, [12] siendo muy popular en aquel entonces porque no había ningún algoritmo que funcionará mejor que éste. Tessereact se basa en analizar los contornos de caracteres y una vez hallados los contornos, utiliza *Blobs (conjunto de cosas, masa uniforme),* debido a que el algoritmo busca encontrarlos para juntar contornos cercanos en una misma área y luego procesarlos de manera individual.

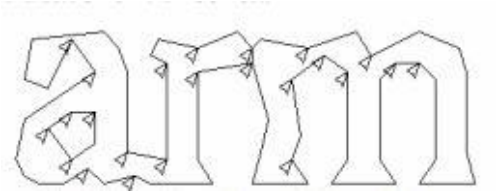

Ilustración 17: reconocimiento de bordes por el algoritmo Tessereact

Posteriormente esos *blobs* forman las líneas de texto que luego constituyen las palabras, éstas buscan ser reconocidas. Para ello, la función del algoritmo es dividir las palabras en dos conjuntos: en el primer conjunto se incluirían las palabras con caracteres monoespaciados, es decir, aquellas palabras que sus caracteres tengan idéntico espacio, mientras que el segundo conjunto estaría integrado por las palabras que no tengan los caracteres monoespaciados.

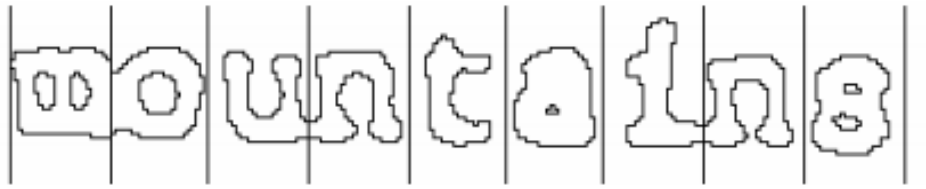

Ilustración 18: Separación de caracteres mediante la técnica de Tessereact

Por último, el proceso de reconocimiento finaliza una vez encontrados los caracteres que han sido clasificados y se extrae el archivo digital.

3.4.2 Google Mobile Vision API

Google Mobile Vision API: es una biblioteca de OCR muy potente y confiable que funciona en la mayor parte del dispositivo Android, pertenece a Google. También se puede usar para la detección de rostros en vivo y el seguimiento de rostros junto con las capacidades de escaneo de códigos de barras.

La estructura de esta biblioteca consiste en segmentar el texto reconocido en bloques, líneas y palabras, donde un bloque es un conjunto contiguo de líneas de texto ya sea un párrafo o una columna, la línea es un conjunto de palabras en el mismo eje horizontal mientras que las palabras se separan debido a los espacios.

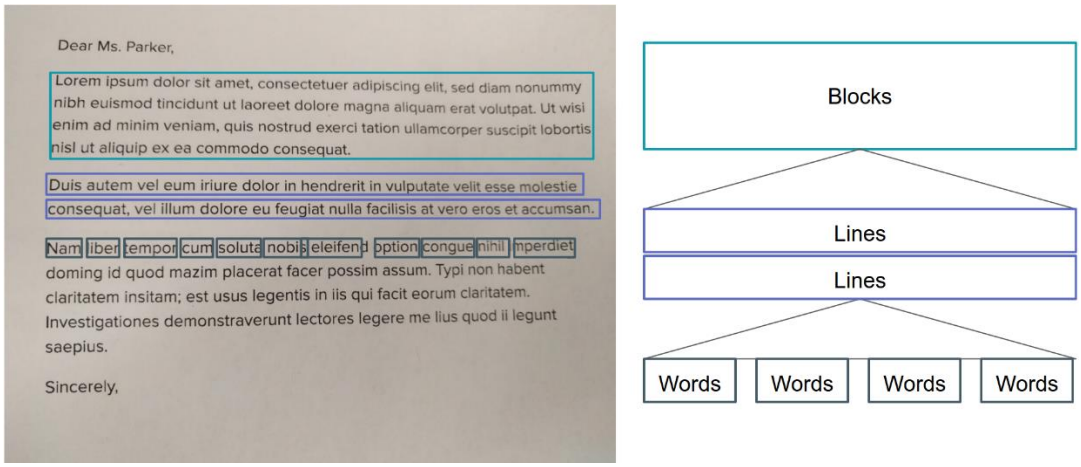

Ilustración 19: Funcionamiento de Google visión API

Una vez conocidas las dos bibliotecas más importantes para el desarrollo de OCR hay que ver los pros y contras de cada una ellas.

De un lado, la biblioteca de Google Vision pertenece a Google y está en constante actualización. Tesseract es de código abierto, gratuito y puede ejecutarse localmente.

Pero también requiere capacitación y ajustes, y en general su índice de reconocimiento es peor que el de los servicios de nube de Google. La última versión de Tesseract data de Junio de 2017 mientras que Google recientemente actualizó la biblioteca de Google Vision API (Mayo 2018). De otro, Tesseract no es apta para nuestro uso ya que el procesado es más lento y no concuerda con los objetivos propuestos. Los resultados de Google Vision son mucho mejores.

Tras haber analizado los pros y contras, la opción que estimamos más oportuna es la biblioteca Google Vision API frente a Tessereact, y a efectos informativos, tener en cuenta que la comunidad de desarrolladores de Android ofrece ejemplos prácticos para entender los conceptos de esta biblioteca. En el transcurso del proyecto, Google lanzó una actualización a esta biblioteca dejando desactualizada a Google Vision API, la misma estaría relacionada con Firebase (plataforma para el desarrollo móvil que incluye además de Android, iOS y web).

3.5 Conclusión Biblioteca OCR Elegida

Google anunció el 8 de mayo en el congreso anual Google I/O celebrado entre el 8 y 10 de mayo en Montain View, California, [13] lanzaron la beta de ML Kit, un kit de desarrollo de software optimizado a fin de desplegar inteligencia artificial para aplicaciones móviles en la plataforma de desarrollo de aplicaciones Firebase. ML Kit, disponible tanto para desarrolladores de Android como iOS, puede invocar API tanto en el dispositivo como en la nube. El kit está diseñado para ser fácil de usar tanto para principiantes como para desarrolladores avanzados. Esta API tiene funciones para reconocimiento de texto, detección de rostros, escaneo de códigos de barras, etiquetado de imágenes y reconocimiento de puntos de interés. Estará disponible de forma gratuita en la consola de Firebase. Las API en el dispositivo no requieren una conexión de red para funcionar, aunque también están disponibles en la nube, como las funciones de etiquetado de imagen y OCR para reconocer texto en un anuncio o cartelera.

Ésta es una actualización de la biblioteca Google Mobile Vision API ya que para el procesamiento de caracteres utiliza el mismo algoritmo como vemos en la siguiente ilustración.

Ilustración 20: Funcionamiento de reconocimiento de caracteres con Firebase ML kit.

El uso de Firebase facilita ciertas funcionalidades que posteriormente puedan ser utilizadas en la aplicación como la creación de una base de datos para los usuarios sin apenas complicación. Aunque esté en fase beta es una herramienta que tiene el soporte de Google y está siendo integrado en varias aplicaciones móviles.

CAPITULO 4

4. Manual del programador

En este capítulo se explica en detalle, las opciones disponibles que se han necesitado a la hora de desarrollar este proyecto. Se describe las bibliotecas externas utilizadas, el diseño del desarrollo Android con sus diferentes tendencias y el entorno de desarrollo del proyecto, y se justifica por qué se ha elegido cada uno de ellos.

4.1 Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android. Su lanzamiento inicial fue el 8 de diciembre de 2014 aunque fue anunciado en la feria internacional de Google I/O en 2013, reemplazo como IDE oficial de desarrollo para Android a Eclipse. La última versión data del 7 de junio de 2018 versión 3.1.3 bajo la licencia de Apache 2.0. Presenta todas las opciones para desarrollar una aplicación programada en Java como en Kotlin.

En cuanto a requisitos necesita:

- Windows 7/8/10 (32 o 64 bits).
- 3 GB de RAM (8 GB de RAM recomendado)
- 2 GB de espacio libre mínimo (4 GB recomendado 500 MB para el IDE + 1.5 GB para Android SDK and la emulación de la imagen de sistema).
- Resolución mínima de 1.280 x 800.
- Java 8.
- 64 bits y procesador Intel (emulador) [14].

4.2 Técnicas de Diseño. Material Design

Google publicitó Material Design en junio de 2014, en una conferencia de I/O como el nuevo lenguaje de diseño, este es un conjunto de guías que tienen como principio dar una identidad a las diferentes aplicaciones y ofrecer una experiencia óptima al usuario. Desde su lanzamiento tuvo una gran tasa de éxito. Material Design salió a la luz a partir de 2015 con un sistema operativo Android versión 5.0, *Lollipop* aunque se puede utilizar desde la versión 4 en adelante. Material Design es utilizado en tanto en aplicaciones móviles como en Gmail, Google Docs, hojas de cálculo, YouTube, Google Maps y, a un rango menor, Chrome. Hasta ahora, incluso las interfaces web de escritorio también se incorporan con Material Design como en Google Drive.

Entre los principios de Material Design se puede destacar [15]:

- Las aplicaciones deben ser sencillas y claras para que el usuario no tenga problemas a la hora del uso de aplicación y mostrar solo lo necesario en cada instante.
- Es importante que los usuarios se encuentran mejor con aplicaciones con símbolos reales en lugar de usar botones y clics.
- El usuario tiene capacidad de elegir, y en caso de elegir erróneamente, dar la opción de retroceder o enmendar el error.
- Introducir notificaciones solo cuando sea importante.
- Dar respaldo a los datos guardados en el dispositivo móvil, para en caso de pérdida no perder todo lo realizado.
- Crear animaciones o movimiento, incluso añadir comentarios a la hora de producirse una iteración.

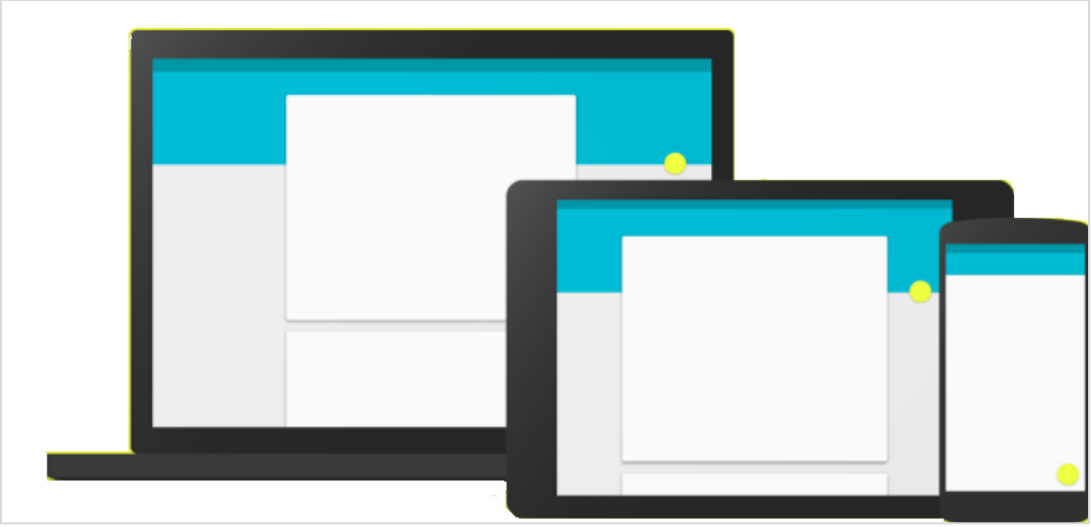

Ilustración 21: ejemplo Material Design en diferentes formatos

Para aplicar estos principios de Material Design hemos añadido varios elementos. A continuación, se describen los que se han considerado más importantes para nuestra aplicación:

4.2.1 Normalización de iconos

Android *Oreo* versión 8.0 intenta poner fin al problema de los diferentes tipos de iconos que existen por partes de los desarrolladores, ya que su diseño es un conflicto desde el lanzamiento en 2008 hasta los últimos años. Esto era debido a que no existían unas dimensiones preestablecidas y los desarrolladores solo se limitaban a seguir las dimensiones máximas y mínimas. Esto llevaba a que los diferentes iconos fueran de forma cuadrangular, con bordes cuadrados o redondeados, lo que cambiaba completamente el intento de Material Design por buscar una cierta homogeneidad entre todos.

Los desarrolladores de aplicaciones deberán crear los iconos mediante dos capas, una circular con el logo y posteriormente una segunda, para el fondo de la aplicación.

De esta manera se podrá adaptar la interfaz de usuario a los iconos y permite la posibilidad de crear animaciones cuando se produzca una iteración con ese icono.

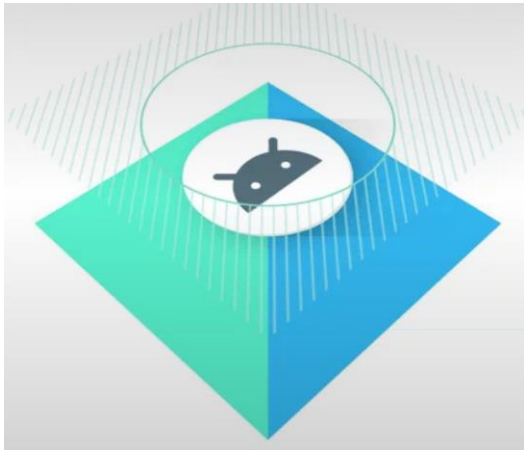

Ilustración 22: Crear un icono con Material Design

En la aplicación se ha creado un icono que cumple estas características anteriormente comentadas.

4.2.2 NavigationDrawer

NavigationDrawer apareció en la versión de Android 5.0 *Lollipop*. Consiste en un patrón de navegación que proporciona acceso a los diferentes destinos de la aplicación. Su funcionamiento consiste en un panel, que mediante una interacción de desplazamiento de derecha a izquierda surge para ofrecer al usuario las diferentes opciones que existen

de la aplicación. También permite el acceso al mismo tocando el icono que le corresponde en la barra superior de acciones.

NavigationDrawer está recomendado para aplicaciones con cinco o más destinos de alto nivel, aplicaciones con dos o más niveles de jerarquía de navegación o para una navegación rápida entre destinos no relacionados.

Este elemento se consolidó desde sus inicios, las aplicaciones más conocidas de Google lo han adoptado y se ha transformado en un elemento primordial siempre que las acciones ofrecidas por la aplicación sean las recomendadas.

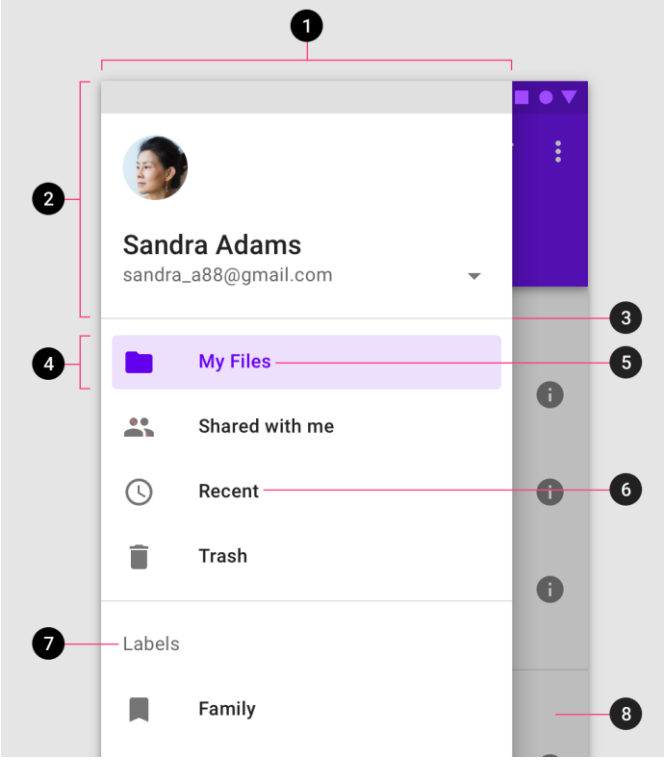

Ilustración 22: NavigationDrawer diferentes opciones que puede tener.

En el *NavigationDrawer* se puede incluir distintos elementos como se muestra en la ilustración:

1. Contenedor donde se encuentra todos los menús y cabecera.
2. Cabecera es opcional.
3. Líneas divisorias son opcionales.
4. Sombra al texto activo.
5. Texto activo.
6. Texto no activo
7. Segundo título.

8. Capa semitransparente opcional.

En el caso de la aplicación desarrollada, su contenido tiene estos elementos ya que ayuda a mejorar la navegación para el usuario.

4.2.3 RecyclerView

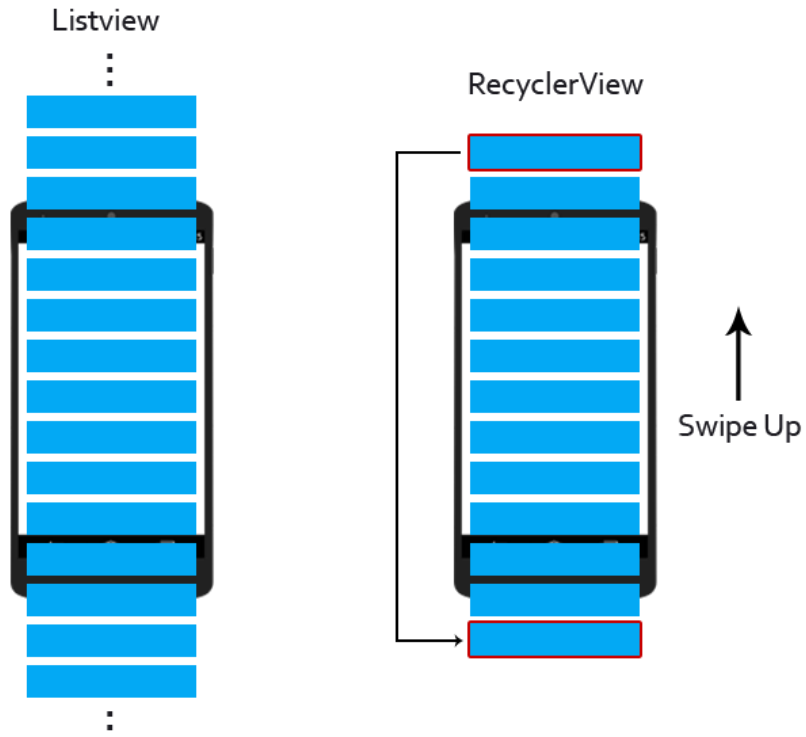

Ilustración 23: Diferencia entre ListView y RecyclerView

Una vista *ListView* visualiza una lista deslizable verticalmente de varios elementos donde cada elemento puede definirse como un *layout* [16]. Este tipo de vistas eran muy comunes hasta el inicio del Material Design. Esta vista ha sido implementada en aplicaciones Android para mostrar datos en forma de lista, sin embargo, este tipo de vista fue sustituido en la actualidad por la vista *RecyclerView*.

RecyclerView, aunque cumpla las mismas funciones que la clase *ListView*, es una versión de contenedor de elementos más avanzado que *ListView*, puesto que proporciona la capacidad de seleccionar el elemento que se desee para poder realizar otras funciones con ellos y diseñar animaciones en cuanto desplazamiento y creación de elementos que hacen que la aplicación sea más eficiente de cara al usuario.

Su funcionalidad principal es mostrar un conjunto de datos que permita al usuario una experiencia fluida, para ello, esta clase solo permite una cantidad de vistas para que se pueda visualizarlas y no cargar todas como haría el *ListView*.

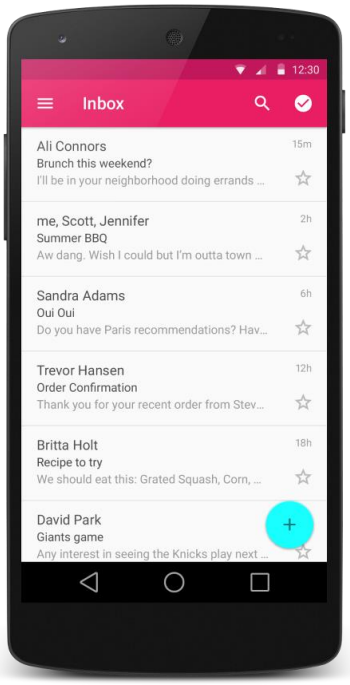

Ilustración 24: Ejemplo de RecyclerView

4.2.4 BottomNavigationView

BottomNavigationView es un patrón de navegación añadido por Google en 2016. No estuvo habilitado hasta principios de 2017. Para usarlo se necesita tener una versión de Android Studio 3.0 o superior.

Consta de una barra de navegación en el parte inferior de la vista, siendo recomendable usarla cuando existan de tres a cinco vistas diferentes. Cada vista se representa por un icono y un texto que puede ser opcional. Cuando se toca un ícono, se lleva al usuario al destino elegido, esto es una opción ideal para cambiar de vista mediante una pulsación.

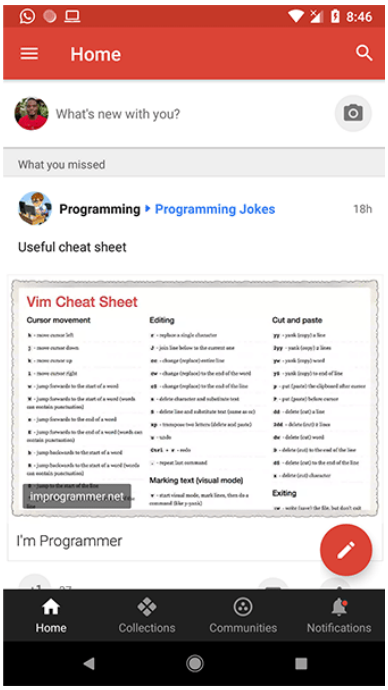

Ilustración 25: ejemplo de NavigationView en una aplicación

4.2.5 Floating Action Button (FAB)

Floating Action Button se puede considerar un estándar de *Material Design* para la interfaz de usuario. FAB es un botón para recalcar un uso en la aplicación. Se define con una forma circular, un icono en la parte interna que simboliza la acción que se va a realizar.

Tiene la capacidad de cambiar de escala, forma o color cuando el usuario hace una iteración con el FAB y es posible añadir un menú dentro de FAB para promover acciones derivadas de la acción principal.

Ilustración 26: Ejemplo de FAB con menu

4.2.6 ConstraintLayout

ConstraitLayout fue una de las novedades anunciadas en 2016 por Google, un *layout* para cambiar el diseño de interfaces. *ConstraintLayout* permite crear diseños grandes y complejos con una jerarquía de vista plana (sin grupos de vista anidados), siendo una característica fundamental que tener en consideración puesto que estas jerarquías son mucho más eficientes.

Es similar a *RelativeLayout* porque todas las vistas se presentan de acuerdo con las relaciones entre los diferentes elementos y el diseño principal, pero es más flexible que *RelativeLayout*. Su principal ventaja respecto este es que *ConstraitLayout* está disponible directamente desde las herramientas visuales del Editor de diseño, ya que la API de diseño y el Editor de diseño se crearon especialmente para este, para hacer el diseño de una manera visual y gráfica, no editando mediante XML.

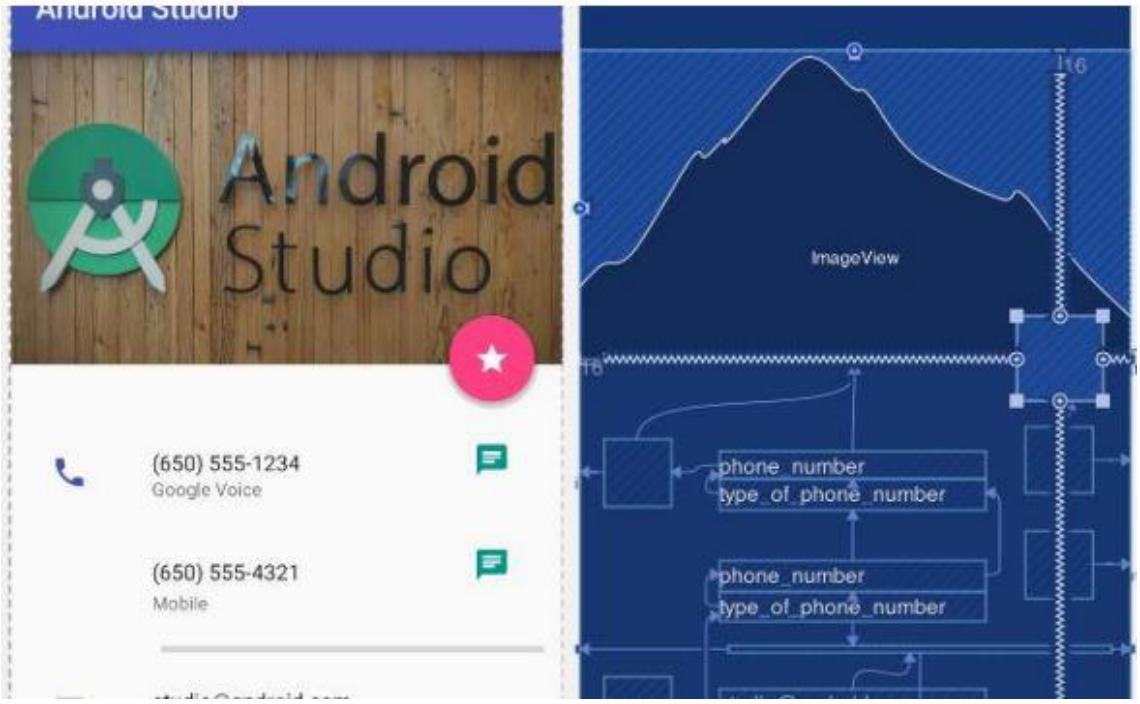

Ilustración 27: ejemplo de ConstraintLayout

4.3 Estado de las bibliotecas

El uso de bibliotecas en Android para el desarrollo de una aplicación es común ya que nos facilita el desarrollo de tareas comunes como el consumo de APIs, la edición de imágenes e incluso cuando el código se repite.

Algunas bibliotecas utilizadas en el desarrollo de la aplicación Android han sido:

4.3.1 Apache Commons

Apache Commons es un proyecto de la Apache Software Foundation, anteriormente bajo el Proyecto Jakarta. El propósito de Commons es proporcionar software Java de código abierto reutilizable [17].

Forma una parte importante de la aplicación, ya que esta biblioteca da la posibilidad de ofrecer un servicio de respaldo a los diferentes ficheros editables e imágenes capturadas en la aplicación.

La funcionalidad global se basa en un protocolo TFTP (*Trivial File Transfer Protocol*) que envía y recibe ficheros cuando sea posible, pero también da la posibilidad de poder desarrollar una implementación personalizada de las clases de paquete y los métodos de envío y recepción de paquetes TFTP.

4.3.2 Android Image Cropper

Durante el capítulo 3 se ha comprobado que existen diferentes opciones a la hora de elegir una aplicación OCR. Todas tienen vrias características comunes y una de ellas es la edición de imagen, por esto, se ha decidido usar la biblioteca para la edición de imágenes Android Image Cropper, que es una potente biblioteca para cumplimentar esta funcionalidad. Para dar la posibilidad al usuario de la edición antes de la extracción de texto de la imagen.

Esta biblioteca contiene diferentes características:

- Establece la imagen de recorte como un mapa de bits, recurso o un URI de Android.
- Rotación e inversión de la imagen.
- Auto zoom.

- Establece los limites mínimos y máximos de la imagen resultante en pixeles.
- La API mínima de Android es la 14.
- Permite hacer diferentes tipos de corte modo rectangular u oval.

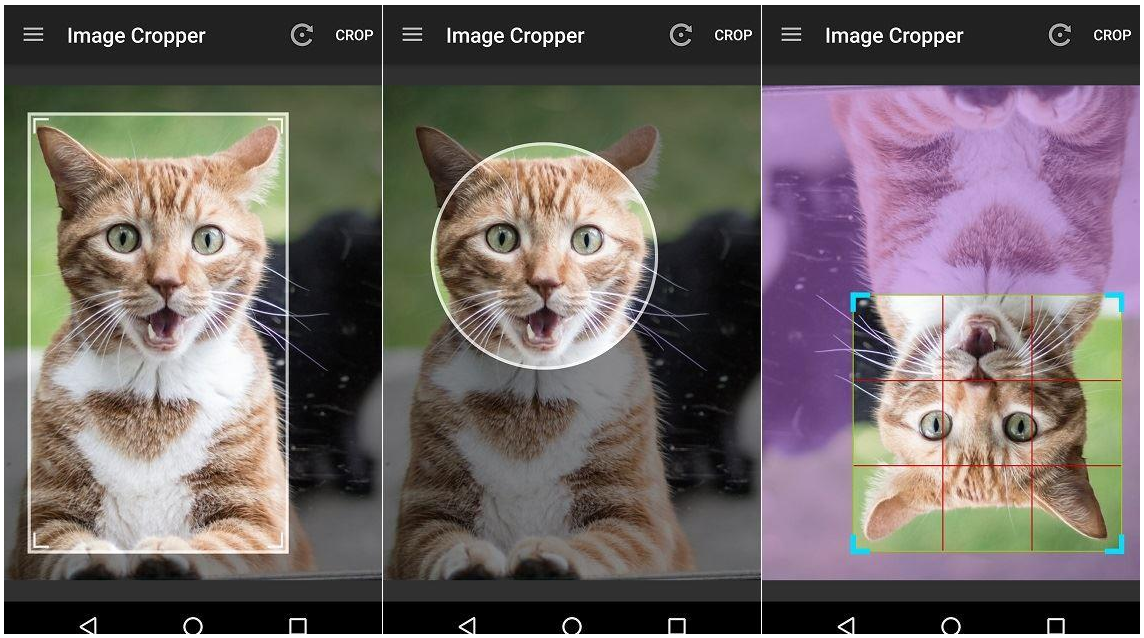

Ilustración 28: ejemplo de la biblioteca Image Cropper [18]

4.3.3 FloatingActionButton

Las técnicas de Material Design son muy importantes a la hora de desarrollar una aplicación. Se ha visto que el botón FAB es un estándar en las aplicaciones actuales como Facebook, Gmail etc.

Ilustración 29: FAB button en la app de Facebook para Android

Para el desarrollo de la aplicación se ha buscado una biblioteca que implemente un menú para las diferentes opciones de extracción de texto en imágenes. Se ha tenido en cuenta a la hora de las diferentes iteraciones con el botón FAB la inclusión de animaciones y respuesta visual para una óptima experiencia con el usuario. Es por ello que se ha añadido la biblioteca FloatingActionButton [19] aunque es una API extremadamente limitada en este momento. Soluciona algunos casos de uso propuestos de la aplicación.

La versión mínima para el uso de esta API es la 14, por lo que no habrá ningún problema.

4.3.4 Glide

Glide admite la búsqueda, la decodificación y la visualización de imágenes fijas, videos y GIF animados [20].

El objetivo principal de Glide es hacer que el desplazamiento de cualquier tipo de lista sea lo más fácil y rápido posible, pero también es efectivo para casi cualquier caso en el que se necesite buscar, cambiar el tamaño y mostrar una imagen remota.

Como en la aplicación necesitamos una galeria donde se muestren las imágenes deseadas capturadas con su texto, Glide es una biblioteca que cumple todo lo necesario para crear un tipo de galeria perfecta y rapida.

4.4 Consideraciones para tener en cuenta en el desarrollo

Existen varias características que requieren ser descritas ya son importantes a la hora de desarrollar una aplicación de este tipo:

4.4.1 Permisos

Una de las características que incluye la versión de Android 6.0 y que se mantiene durante las versiones posteriores es un control de los permisos de las aplicaciones, mientras que cuando se instalaba una aplicación en versiones anteriores se mostraban los permisos que requería al inicio de la aplicación sin poder cambiarlos una vez aceptados. Desde Android 6.0 se tiene control sobre los permisos más importantes en cuanto a proteger nuestra privacidad, como encender el micrófono, usar la cámara o leer contactos [21].

La aplicación desarrollada en este proyecto necesita permiso de cámara y de escritura en memoria de teléfono, esto se podría hacer de dos formas:

- Mediante el *smartphone* ir la entrada donde se encuentran las aplicaciones buscar el permiso de lectura y cámara y activarlo como se ve en la ilustración 30, esto es atravesado, por tanto, se realiza una comprobación mediante código.

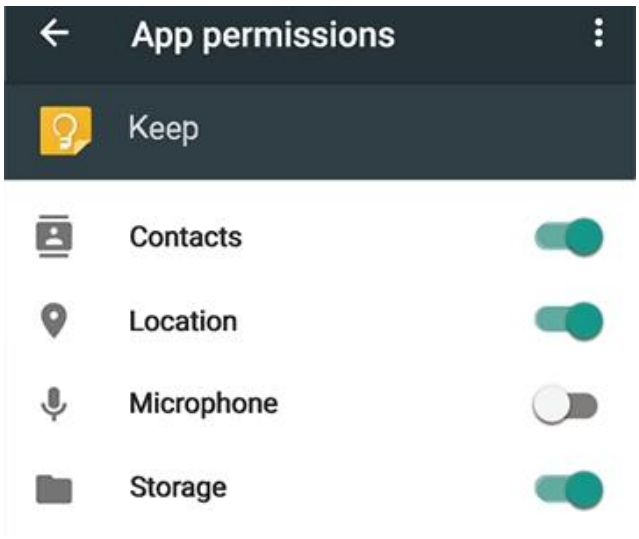

Ilustración 30: Activación de diferentes permisos en la aplicación Keep

- Mediante el código: se solicita de una manera explícita al usuario si tiene otorgados los permisos o no. La petición se hace durante el uso de la aplicación pudiendo aceptar o denegar permisos sin que esto pueda afectar a la aplicación, aunque si se rechaza el permiso y se vuelve a hacer un uso de ese permiso se

volverá a pedir el permiso en cuestión. Existe una ventana donde se indica la opción “No volver a preguntar” para rechazarlo siempre como se observa en la ilustración 31.

Ilustración 31: Cuadro de dialogo peticion de permiso

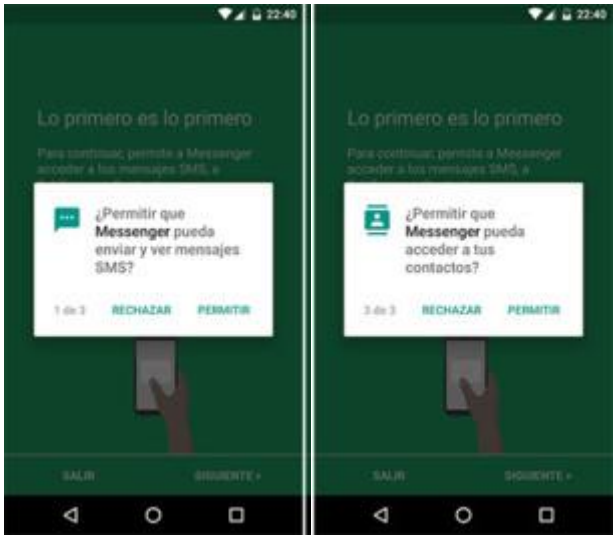

Ilustración 32: Petición de acceso a los contactos y al envío de SMS

4.4.2 SurfaceView

En el momento en el que Android comenzó a soportar la grabación de video o la captura de fotos muchos desarrolladores fueron incapaces de implementar la funcionalidad. El motivo principal fue la necesidad de proporcionar un "objeto superficie" para los diferentes modos de captura [22].

SurfaceView consiste en una superficie de renderizado dedicado a animaciones con un escenario exigente a diferencia del uso vista normal. Tiene la necesidad del uso de un hilo (*Thread)* secundario para actualizar los contenidos, lo que hace que este en constante actualización y necesita procesos diferentes para conseguir un redibujado. Esto consiste en crear una superficie sobre la que representar el texto o formas dentro del *SurfaceView* que será la pantalla del *smartphone*. Una vez creada se inicia el hilo que se encarga de dibujar sobre la superficie, este método se llama *SurfaceHoder*. Posteriormente, se deberá llamar al método *onDraw* que su función es ir editando la superficie si ha ocurrido nuevas animaciones. Estos métodos se encontrarán en constante actualización en el hilo como se observa en ejemplo de la ilustración 33 en el movimiento de un circulo.

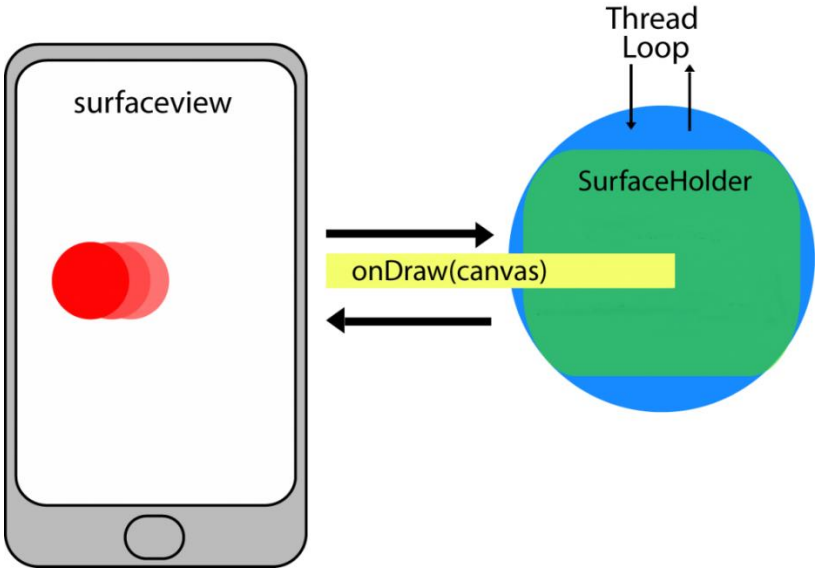

Ilustración 33: Ejemplo esquematico de SurfaceView

4.4.3 AsyncTask

En la aplicación se ha propuesto la posibilidad de crear una copia de respaldo a los diferentes ficheros que se crean en la aplicación. La decisión ha sido crear un servidor FTP (*File Transfer Protocol*) para ese respaldo si el usuario lo considera oportuno.

Esta Tarea se realiza en segundo plano ya que no se quiere bloquear la interfaz de usuario mientras se realiza el envío al servidor, se ejecuta de forma síncrona, utilizando la clase AsyncTask, que, además de ejecutar el código en hilo independiente, proporciona un método que se activa al terminar su ejecución que sirve para crear notificaciones al usuario y crear un dialogo del proceso de subida.

Ilustración 34: ejemplo de cuadro de dialogo.

El ciclo de vida de esta clase, en nuestra aplicación, se crea un hilo secundario usando AsyncTask una vez pulsado el botón enviar para crear la copia de respaldo.

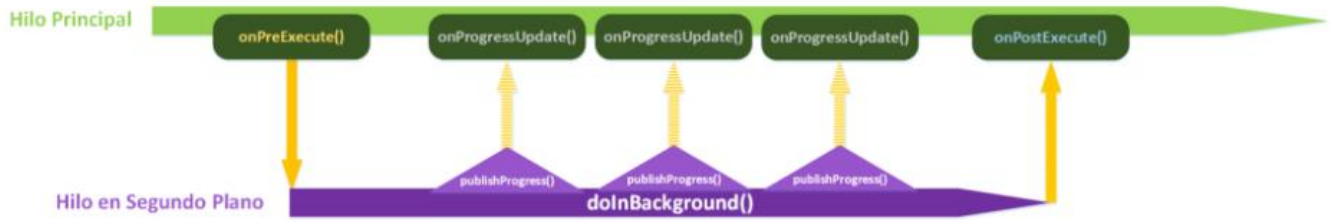

Ilustración 35: Ciclo de vida de la clase AsyncTask

Como es una subida de varios ficheros a un servidor FTP se debe avisar al usuario con un cuadro de dialogo, el método usado será *OnPrexecute()* que se ejecuta sobre el hilo principal, mientras que la subida de ficheros se realiza en segundo plano mediante el método *doinBackground*().

Durante la subida de los ficheros se realiza una actualización al hilo principal con la ayudo de *publishProgress*(), que sobrescribe un método llamado *onProgessUpdate*(). Por último, cuando se termina la subida de ficheros se llama al método *onPostExecute*() y se da por finalizado el hilo en segundo plano.

CAPÍTULO 5

5. Manual de Usuario

En este capítulo se describe todo respectivo al uso de la aplicación desarrollada y cómo se debe utilizar para que los usuarios puedan emplear todas las funcionalidades que ofrece la aplicación.

5.1 Funcionalidades

Las funcionalidades consisten en pruebas para comprobar todos los posibles caminos lógicos que se puede tomar en la aplicación. Éstas se detallan a continuación y posteriormente se describen más en profundidad mediante casos de uso a través de tablas.

- Modo Captura Normal
- Edición de Imagen
- Mostrar texto en pantalla
- Modo Captura Texto en cámara
- Configurar Modo Automático
- Modo Automático
- Galería
- Detalle de fichero capturado
- Enviar fichero a servidor FTP
- Ayuda
- Acerca De

Identificador	**Modo Captura Normal**	
Autores	Alfonso Jiménez Hernández	
Descripción	El usuario podrá capturar una imagen desde la cámara	
Actores o personal involucrado	Usuario	
Precondición	El usuario debe haber aceptado los permisos de uso de cámara y almacenamiento.	
	Paso	**Acción**
	1	El usuario pulsa el botón FAB
	2	El sistema presenta los diferentes modos de reconocimiento que existen
	3	El usuario elige Modo Normal

Secuencia Normal de sucesos	4a	El sistema muestra donde se guarda la imagen capturada.
	4b	El sistema abre la cámara propia del sistema.
	5	El usuario captura la foto que desea
	6a	El sistema cierra la aplicación de cámara.
	6b	El sistema muestra la foto capturada en la pantalla principal de la aplicación.
Postcondición	Se cumple correctamente a la funcionalidad deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-4	Si el usuario pulsa el botón *Back*, el caso de uso retrocede una secuencia de uso
Requisitos especiales		Ninguno
Comentarios		Ninguno

Tabla 1: Caso de uso Modo Captura Normal

Identificador	**Edición de Imagen**	
Autores	Alfonso Jiménez Hernández	
Descripción	El usuario podrá editar una imagen desde la aplicación para su posterior uso.	
Actores o personal involucrado	Usuario	
Precondición	El usuario debe haber aceptado los permisos de uso de cámara y almacenamiento.	
Secuencia Normal de sucesos	**Paso**	**Acción**
	1	El usuario pulsa el botón FAB con el icono de edición.
	2	Si hay una imagen capturada mediante el modo Normal el sistema usa esa captura
	3	El sistema abre una nueva ventana donde se muestran diferentes opciones para la edición de imagen
	4	El usuario interacciona con las diferentes herramientas de edición de imagen y cuando está listo da en la *Toolbar* donde indica “listo”.
	5	El Sistema carga la foto en la pantalla principal de la aplicación una vez editada.

Postcondición	Se cumple con la función deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	2b	Si no existe ninguna imagen capturada anteriormente se muestran diferentes opciones de donde cargar la imagen para su posterior uso
	2c	El usuario elige la opción seleccionada.
	2d	El sistema abre la aplicación correspondiente donde se encuentran las imágenes para capturar
	2e	El usuario elige la imagen para su posterior edición en la secuencia siguiente del caso de uso.
	1-4	Si el usuario pulsa el botón *Back*, el caso de uso retrocede una secuencia de uso
Requisitos especiales		Ninguno
Comentarios		Ninguno

Tabla 2: Caso de uso Edición de imagen

Identificador	**Mostrar texto en pantalla**	
Autores	Alfonso Jiménez Hernández	
Descripción	El sistema muestra el texto reconocido de la imagen previamente seleccionada	
Actores o personal involucrado	Usuario	
Precondición	El usuario debe haber seleccionado una imagen previamente.	
Secuencia Normal de sucesos	**Paso**	**Acción**
	1	El usuario pulsa el botón FAB con el icono de *Imagen to Text*
	2	El sistema muestra los caracteres que han sido reconocidos de la imagen previamente seleccionada en la pantalla principal.
	3	El sistema guarda el texto reconocido en un fichero texto y la imagen correspondiente para su posterior uso
Postcondición	El resultado es el esperado	

Alternativas o flujo alternativo	**Paso**	**Acción**
	1-1	Si el usuario no tiene una imagen previamente seleccionada no contina con la secuencia del caso de uso
Requisitos especiales		Ninguno
Comentarios		Ninguno

Tabla 3: Caso de uso Mostrar texto en Pantalla

Identificador	**Modo Captura Texto en cámara**	
Autores	Alfonso Jiménez Hernández	
Descripción	El usuario podrá visualizar a tiempo real el texto reconocido de la imagen que se muestre por cámara	
Actores o personal involucrado	Usuario	
Precondición	El usuario debe haber aceptado los permisos de uso de cámara y almacenamiento.	
Secuencia Normal de sucesos	**Paso**	**Acción**
	1	El usuario pulsa el botón FAB
	2	El sistema presenta los diferentes modos de reconocimiento que existen
	3	El usuario elige Modo Texto en cámara
	4	El sistema muestra una vista *SurfaceView* de cámara dentro de la propia aplicación
	5	El usuario desplaza la cámara hacia donde existe imagen con texto.
	6	El sistema muestra el texto reconocido en la misma vista donde se está viendo la imagen a tiempo real
Postcondición	El resultado es el esperado	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-4	Si el usuario pulsa el botón Back, el caso de uso retrocede una secuencia de uso.
	1-6	Si los caracteres exceden de la pantalla del móvil se representa con puntos suspensivos.
Requisitos especiales		Ninguno

Comentarios		Ninguno

Tabla 4: Caso de uso Mostrar Captura en Texto en Cámara

Identificador	**Configuración Modo Automático**	
Autores	Alfonso Jiménez Hernández	
Descripción	El usuario configura el modo automático para capturar los diferentes datos que él quiera obtener un fichero XML que posteriormente se visualizará por pantalla.	
Actores o personal involucrado	Usuario	
Precondición	El usuario debe haber abierto el menú de las diferentes opciones que existen	
Secuencia Normal de sucesos	**Paso**	**Acción**
	1	El usuario pulsa Configuración Modo Automático
	2	El sistema muestra las diferentes opciones que se necesitan para comenzar el reconocimiento de texto.
	3	El usuario escribe las diferentes opciones que se necesitan para el reconocimiento.
	4	El usuario pulsa el botón Siguiente
	5	El Sistema muestra las diferentes opciones para crear el documento XML.
	6	El usuario añade las diferentes etiquetas que luego el sistema utilizará para todo el algoritmo de reconocimiento.
	7	El usuario da al botón FAB siguiente
	8	El Sistema muestra da al usuario la posibilidad de volver a inicio o continuar al modo automático
	9	Independientemente de lo que el usuario elija, se ha configurado el modo automático
Postcondición	El resultado es el esperado	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-4	Si el usuario pulsa el botón Back, el caso de uso retrocede una secuencia de uso
	1-4b	El usuario no escribe las opciones, el sistema informa de que se

		necesitan escribir esas opciones para continuar
	1-5	Si el usuario se equivoca al introducir una etiqueta o los caracteres a reconocer se podrá eliminar la etiqueta.
Requisitos especiales		Ninguno
Comentarios		Lo normal es que el usuario una vez configurado el modo automático prosiga con el modo Automático. No es posible introducir caracteres no válidos, ya que la aplicación en cada momento tiene un teclado diferente.

Tabla 5: Caso de uso Configuración Modo Automático

Identificador	**Modo Automático**	
Autores	Alfonso Jiménez Hernández	
Descripción	El usuario podrá capturar datos de la cámara, previamente establecidos por el usuario, esos datos se mostrarán en un fichero XML	
Actores o personal involucrado	Usuario	
Precondición	El usuario debe haber configurado el modo automático.	
Secuencia Normal de sucesos	**Paso**	**Acción**
	1	El usuario pulsa el botón FAB
	2	El sistema presenta los diferentes modos de reconocimiento que existen
	3	El usuario elige Modo Automático
	4	El sistema muestra una vista de cámara dentro de la propia aplicación.
	4b	El usuario desplaza la cámara hasta que el sistema notifica para parar el desplazamiento.
	5	El sistema envía una notificación al usuario de que ha encontrado el inicio de reconocimiento
	6	El sistema guarda la imagen y crea un fichero XML con los datos capturados en la secuencia 4
	7	El Sistema da al usuario la posibilidad de dar un nombre al

		fichero XML creado
	8	El usuario escribe el nombre del fichero o pulsa aleatorio.
	9	El sistema muestra el XML creado y una serie de opciones para el usuario mediante un *BottomNavigationView*.
Postcondición	Se completado correctamente la función deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-4	Si el usuario pulsa el botón *Back*, el caso de uso retrocede una secuencia de uso
	1-7	Si el usuario pulsa el botón Inicio, el sistema vuelve a la pantalla principal.
	1-7	Si el usuario pulsa el botón Galería, el sistema muestra la galería de la aplicación
	1-7	Si el usuario pulsa el botón Nueva Captura, vuelve al caso de uso Configurar Modo Automático.
Requisitos especiales		Ninguno
Comentarios		Ninguno

Tabla 6: Caso de uso Modo Automático

Identificador	**Galería**	
Autores	Alfonso Jiménez Hernández	
Descripción	El usuario podrá ver todas las capturas y textos reconocidos en modo lista sin entran en detalle.	
Actores o personal involucrado	Usuario	
Precondición	El usuario debe haber capturado y reconocido texto con anterioridad.	
Secuencia Normal de sucesos	**Paso**	**Acción**
	1	El usuario accede al menú lateral
	2	El usuario pulsa la opción Galería
	3	El sistema carga todas las imágenes y textos almacenados anteriormente en un *RecyclerView*

	4	El usuario navega por la galería viendo las capturas
Postcondición	Se ha accedido correctamente a la funcionalidad deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-4	Si el usuario pulsa el botón *Back*, el caso de uso retrocede una secuencia de uso
Requisitos especiales		Ninguno
Comentarios		Paso previo al caso de uso Detalle de fichero capturado

Tabla 7: Caso de uso Galería

Identificador	**Detalle de fichero capturado**	
Autores	Alfonso Jiménez Hernández	
Descripción	El usuario podrá visualizar en detalle las capturas y texto capturados.	
Actores o personal involucrado	Usuario	
Precondición	El usuario debe estar en el caso de uso Galería	
Secuencia Normal de sucesos	**Paso**	**Acción**
	1	El usuario pulsa en el *RecyclerView* la imagen que quiera visualizar
	2	El sistema muestra la foto capturada de la cual se creó un fichero tanto XML, como texto y muestra el nombre de este.
	3	El usuario pulsa el botón de Texto
	4	El sistema muestra una vista donde se visualiza el contenido del fichero. Se muestran también varias opciones mediante un *ButtonNavigationView*
Postcondición	El resultado es el esperado	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-2	Si el usuario pulsa el botón Back, el caso de uso retrocede una secuencia de uso.
	1-4	Si el usuario pulsa el botón Atrás, el caso de uso retrocede una secuencia de uso.
	1-7	Si el usuario pulsa el botón Enviar, se envía la imagen y el fichero

		mediante FTP entrando el caso de uso de Enviar archivo a servidor FTP
	1-7	Si el usuario pulsa el botón Galería, vuelve al caso de uso Galería.
Requisitos especiales		Ninguno
Comentarios		Ninguno

Tabla 8: Caso de uso Detalle de fichero Capturado

Identificador	**Enviar fichero a servidor FTP**	
Autores	Alfonso Jiménez Hernández	
Descripción	El usuario podrá enviar los ficheros seleccionados a un servidor FTP para crear una copia de respaldo	
Actores o personal involucrado	Usuario	
Precondición	El usuario debe encontrarse en la pantalla de caso de uso Detalle de fichero capturado El sistema debe poseer acceso a Internet	
Secuencia Normal de sucesos	**Paso**	**Acción**
	1	El usuario pulsa el botón enviar.
	2	El sistema comprueba si tiene acceso a Internet
	3	El sistema crea un dialogo para notificar al usuario la subida de los ficheros.
	4	El sistema crea un hilo secundario para la subida de los ficheros.
	5	El sistema una vez subido el fichero completo notifica al usuario el éxito de la subida.
Postcondición	El resultado es el esperado	
	Paso	**Acción**

Alternativas o flujo alternativo	1-2	Si el sistema no tiene Internet se notifica al usuario la necesidad de su uso
Requisitos especiales		Ninguno
Comentarios		Ninguno

Tabla 9: Caso de uso Enviar fichero a servidor FTP

Identificador	**Acerca de**	
Autores	Alfonso Jiménez Hernández	
Descripción	El usuario se informará acerca de la aplicación y su desarrollo	
Actores o personal involucrado	Usuario	
Precondición	Ninguna.	
Secuencia Normal de sucesos	**Paso**	**Acción**
	1	El usuario elige la opción *Acerca de* del menú lateral.
	2	El sistema muestra la información acerca de la aplicación en la pantalla
Postcondición	Se ha accedido correctamente a la funcionalidad deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-2	Si el usuario pulsa el botón Back, el caso de uso retrocede a la pantalla principal.
Requisitos especiales		Ninguno
Comentarios		Ninguno

Tabla 10: Caso de uso Acerca de

Identificador	**Ayuda**
Autores	Alfonso Jiménez Hernández
Descripción	El usuario busca información relacionada con el uso de la aplicación

Actores o personal involucrado	Usuario	
Precondición	Ninguna.	
Secuencia Normal de sucesos	**Paso**	**Acción**
	1	El usuario elige del menú la opción *Ayuda.*
	2	El sistema muestra la información acerca de la aplicación en la pantalla
Postcondición	Se ha accedido correctamente a la funcionalidad deseada.	
Alternativas o flujo alternativo	**Paso**	**Acción**
	1-2	Si el usuario pulsa el botón *Back*, el caso de uso retrocede a la pantalla principal.
Requisitos especiales		Ninguno
Comentarios		Ninguno

Tabla 11: Caso de uso Ayuda

5.2 Interfaz de usuario

Una vez visto las diferentes funcionalidades de la aplicación, se presenta una interfaz clara e intuitiva al usuario, a lo largo de la sección se muestran las pantallas más características de la aplicación o que pueda ver algún tipo de duda con ellas.

5.2.1 Pantalla de inicio e icono

Una vez instalada la aplicación, aparece en el dispositivo móvil mediante el icono de la ilustración 37, una vez pulsado el *Launcher* se muestra una pantalla inicial durante 3 segundos, la ilustración 36, que indica que se acaba de entrar en nuestra aplicación OCR, para posteriormente avanzar a la pantalla principal.

Ilustración 36: Pantalla inicial de carga de la aplicación

Para iniciar la aplicación no es obligatorio ningún patrón de registro y las funcionalidades están disponibles para cualquier usuario, aunque en líneas futuras se prevé que se implemente esta funcionalidad.

Ilustración 37: Icono de la aplicación

5.2.2 Pantalla principal

Es la pantalla inicial de la aplicación. Primero, la aplicación pide los diferentes permisos necesarios para utilizarla, como se puede ver en la ilustración 38. En el caso de no aceptarlos, se pedirán cuando se realicen las funcionalidades y se requiera de estos. En la pantalla principal cómo se puede ver en la ilustración 39 (número 1) se muestra una

instrucción en modo imagen si es la primera vez que usas la aplicación, es decir, no hay datos almacenados de la aplicación en el dispositivo móvil.

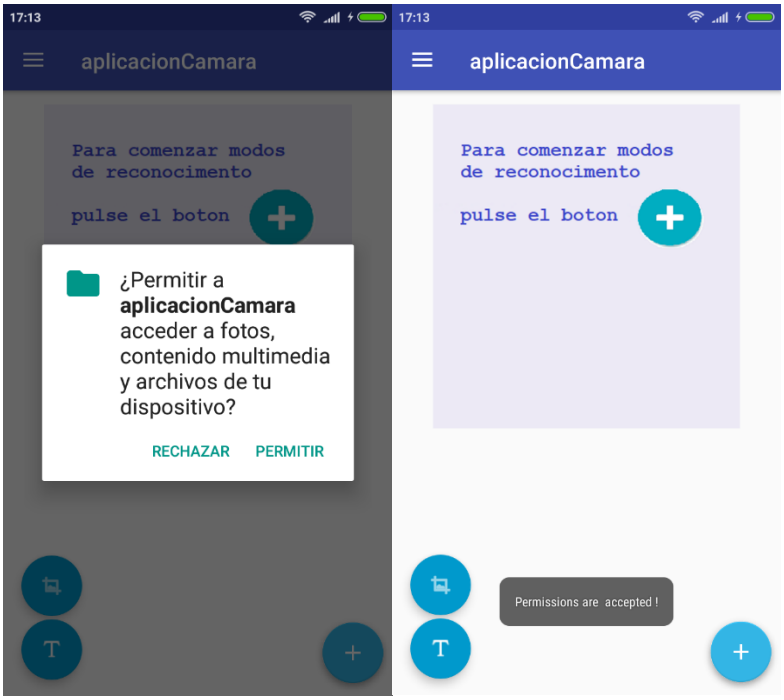

Ilustración 38: Permisos de la aplicación

Esta instrucción una vez hecha la primera captura por el Modo Normal será transformada por la imagen capturada que posteriormente se utilizará para el reconocimiento de texto.

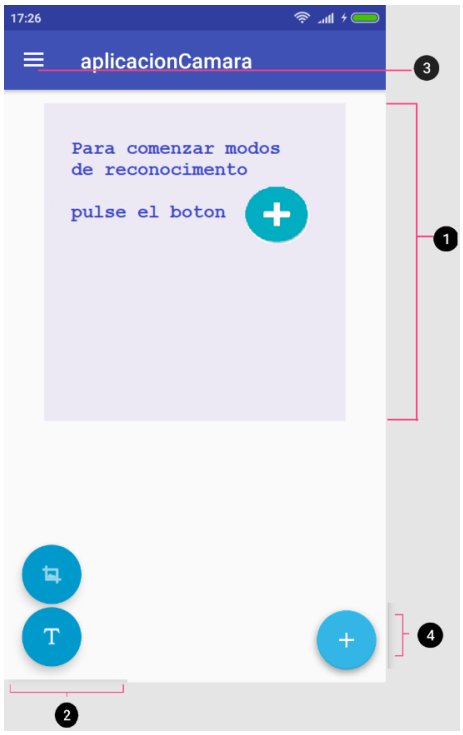

Ilustración 39: Descripción de la pantalla principal

La pantalla principal está formada por tres botones:

- El botón FAB (número 4) que al ser pulsado despliega los diferentes modos de reconocimiento que tiene la aplicación: Modo Normal, Modo Automático y Texto en Cámara (ilustración 40).

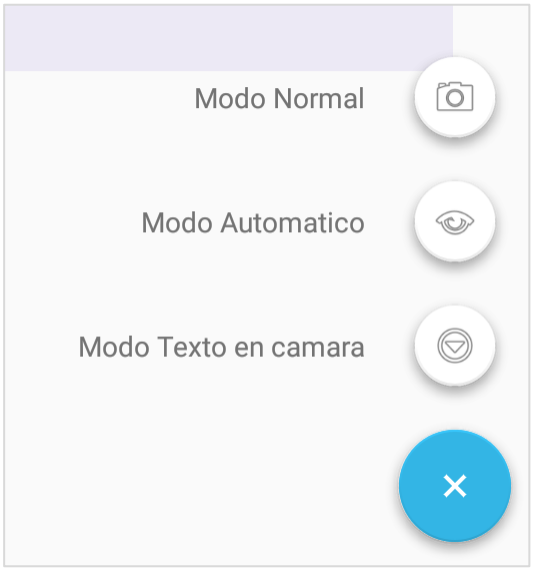

Ilustración 40: FAB menú

- El botón FAB con el icono de recortar, utilizado para la edición de imagen. Si no hay ninguna imagen capturada dará la opción de cargar imagen desde diferentes sitios como Google Drive, galería etc. (ilustración 41).

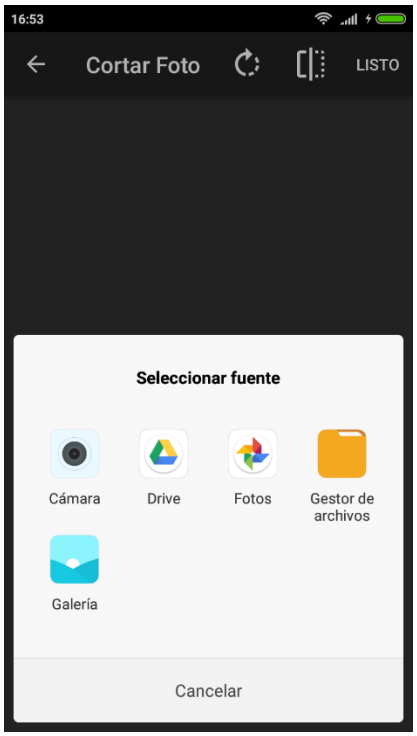

Ilustración 41: Modo edición sin Captura

- El botón FAB con el icono de Texto, utilizado para obtener el texto de la imagen mostrada en la pantalla principal. Si no hay cargada ninguna imagen se alertará mediante un dialogo que no hay ninguna imagen cargada. El reconocimiento de texto, aunque es rápido puede durar más de un segundo y se añade una alerta para no pensar que la aplicación se haya pausado. (ilustración 42).

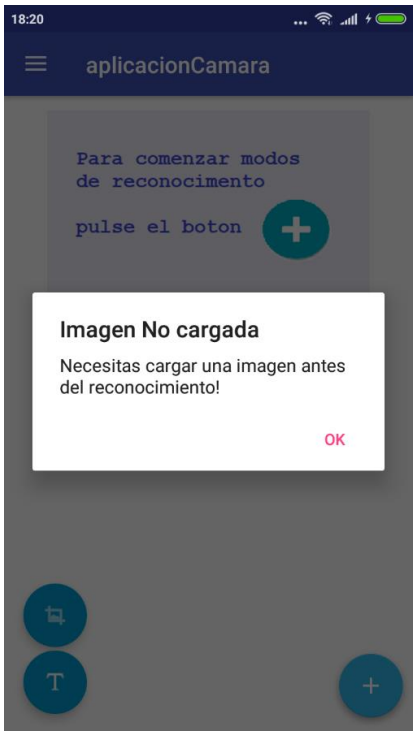

Ilustración 42: Dialogo Reconocimiento de Texto

Si el usuario realiza un desplazamiento a derechas desde la pantalla principal (*swipe right)* aparece el menú de opciones que tiene la aplicación como se observa en la ilustración 43. También se puede pulsar el icono número 3 de la ilustración 39. Desde este menú se puede acceder las diferentes funcionalidades de la aplicación.

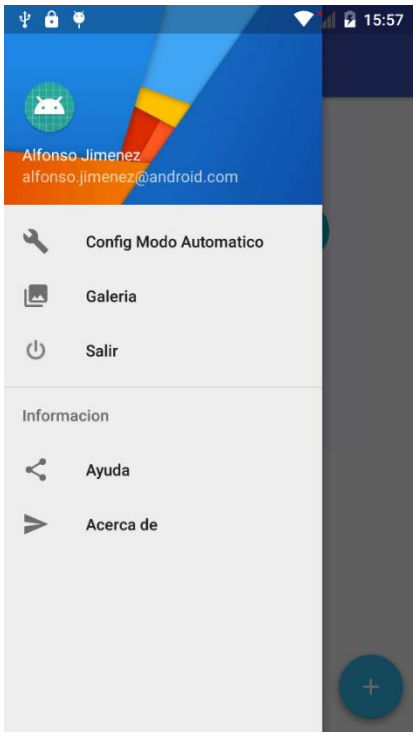

Ilustración 43: Menú de la aplicación

5.2.3 Pantallas Configuración Modo Automático

Consisten en las dos pantallas que se utilizan para la creación de las etiquetas del fichero XML y donde comenzar el reconocimiento.

La pantalla inicial de la configuración ofrece un pequeño formulario con una imagen de ayuda para saber por dónde comenzar el reconocimiento, como se muestra en la ilustración 43. En que el primer formulario se debe elegir un número del uno al diez siendo reflejados en la imagen superior de la pantalla para saber en qué lugar se encontrará la palabra clave. Esta será la segunda parte del formulario. En la primera línea, solo es un teclado numérico mientras que la segunda parte el teclado es alfanumérico, en caso de dejar algún hueco sin rellenar la aplicación mostrará un dialogo para indicar que hemos dejado algo sin escribir.

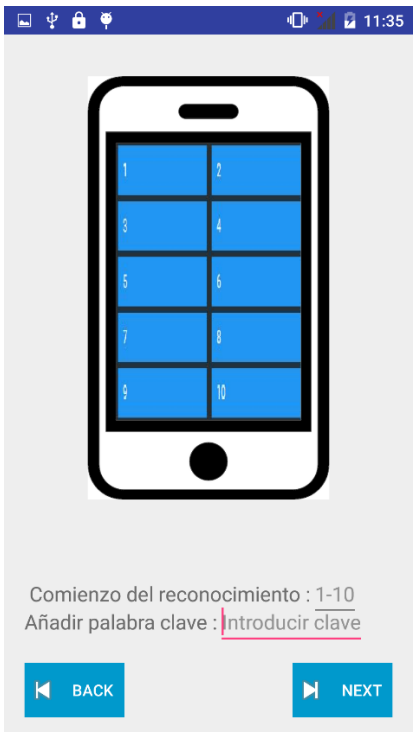

Ilustración 44: Pantalla inicio Configuración modo automático

Posteriormente, como se ha indicado en el caso de uso Configuración Modo Automático de la sección anterior se añadirá las diferentes etiquetas para su inclusión en el documento XML, para generar este documento y crear las etiquetas, el usuario añade las diferentes etiquetas escribiendo en el formulario el nombre de la etiqueta y el

carácter que ha de reconocer, una vez escritos se añade mediante la barra superior (*Toolbar). Esta Toolbar* se utiliza tanto para añadir la etiqueta como para eliminar las posibles etiquetas que el usuario haya creado erróneamente. Se describe en la ilustración 45 a continuación:

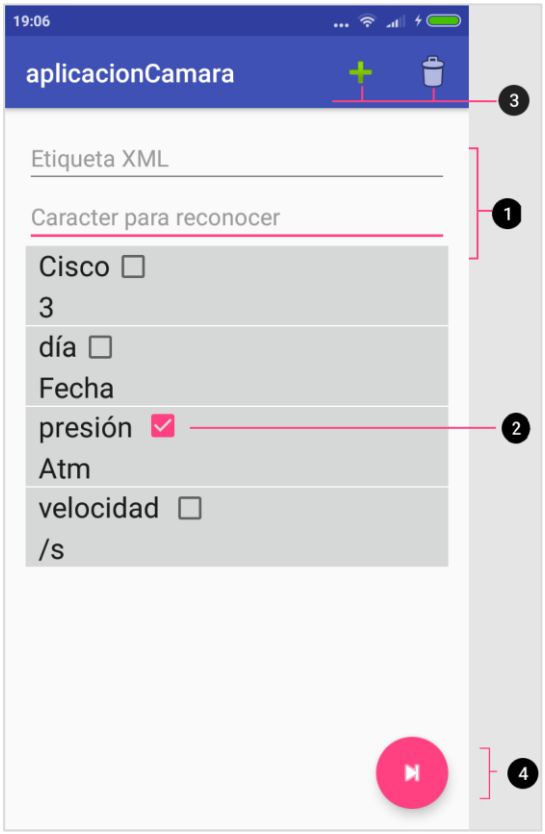

Ilustración 45: Ejemplo pantalla dos Configuración Modo Automático

- El número uno, el formulario que crea las etiquetas, para añadir una etiqueta pulsa en el icono del símbolo "+" (número 3) y se incorpora a la lista para su posterior uso en el modo Automático.
- El número dos, sirve si se ha producido un error o varios, a la hora de añadir uno o varios elementos, solo con pulsar el recuadro de los elementos como se indica en este número y pulsando en el símbolo de papelera de la *Toolbar (*número 3) se eliminan esas etiquetas.
- Mediante el número 4 se da por finalizado la configuración del modo Automático al pulsar tenemos las opciones de volver a la pantalla principal o acceder al modo Automático.

Destacar que la primera etiqueta corresponde a la palabra clave reconocida siendo una etiqueta común en todos los ficheros XML generados.

5.2.4 Modo Normal

El modo Normal es el método más sencillo de los tres modos de reconocimiento. En este modo se utiliza una imagen previa para el reconocimiento. Para comenzar con este modo se selecciona el modo Normal de los tres modos de reconocimiento que existen, como se muestra en la ilustración 40.

Seguidamente se entra en la API de la cámara del dispositivo. Este *framework* incluye compatibilidad con varias cámaras y funciones de cámara disponibles de cada dispositivo, lo que le permite capturar imágenes y videos en la aplicación. Cuando se captura la imagen se muestra en la pantalla principal, eliminando la instrucción del inicio y colocando en pantalla principal la imagen capturada mediante la cámara (ilustración 46).

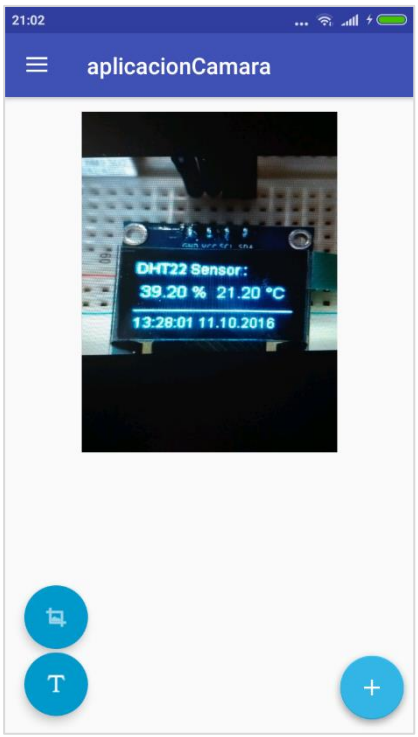

Ilustración 46: Captura antes de reconocimiento

Una vez mostrada la imagen, existe la posibilidad de editarla por si puede haber lugar a dudas en cuanto el reconocimiento de texto o proceder al reconocimiento pulsando el botón inferior de la pantalla principal, una vez pulsado se procede al reconocimiento y se muestra en pantalla como en la ilustración 47.

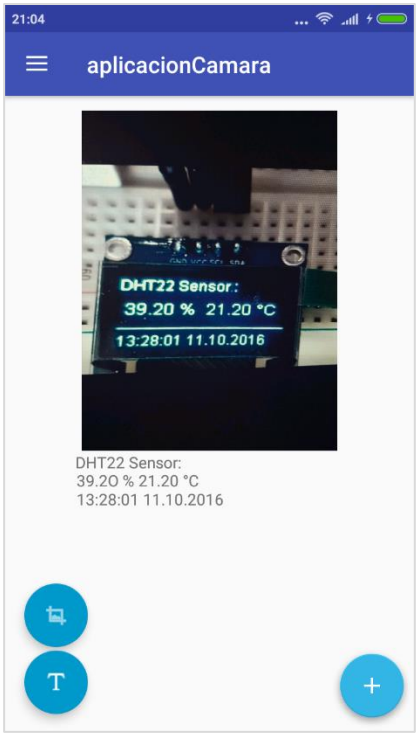

Ilustración 47: Modo Normal Captura de Caracteres

5.2.5 Modo Automático

Es el modo de reconocimiento más importante, necesita la funcionalidad de configuración de este modo ya que si no sería imposible continuar. Este modo es capaz de generar ficheros XML una vez sea configurado el modo Automático. Por tanto, partimos de la configuración de este método como se observa en la ilustración 48.

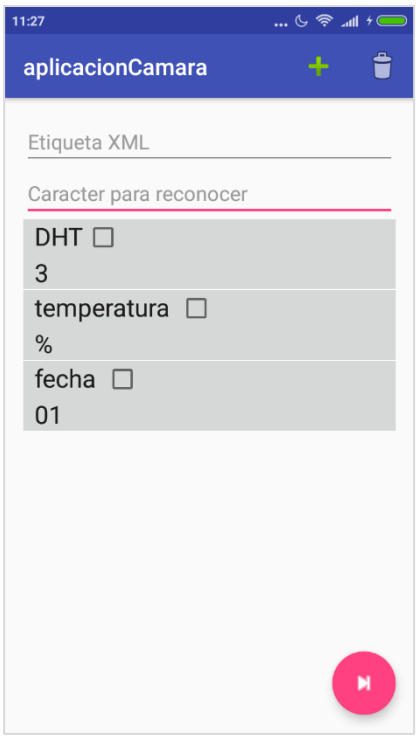

Ilustración 48: Inicio de configuración para Modo Automático

Una vez pulsado el botón *Next* se inicia una pantalla *SurfaceView* pero especifica de la biblioteca llamada *CameraSourcePreview*. En ella se recoge la información en tiempo real mostrada por la cámara y una vez que se encuentre la palabra o carácter clave se produce una notificación sonora y visual mediante un cuadrado rojo que indica que el reconocimiento es exitoso. La notificación visual se observa en la ilustración 49.

Ilustración 49: Notificación Visual Modo Automático

Cuando se notifica la creación del fichero XML generado se muestra al usuario como en se ve en la ilustración 50. Al finalizar este modo de reconocimiento se da la posibilidad mediante un *BottonNavigationView* de elegir si necesita un nuevo reconocimiento, visitar la galería o volver a la pantalla principal.

Ilustración 50: Creación de XML por Modo Automático

5.2.6 Modo Texto en cámara

El tercer modo de reconocimiento es en la propia cámara del dispositivo, una mezcla entre los dos anteriores, por un lado, tiene el reconocimiento en tiempo real por medio de un *CameraSourcePreview* y por otro, reconoce el texto en la propia pantalla. Es utilizado para tener una idea previa de la captura que se va a realizar para los dos anteriores métodos. En este modo al ser mostrado en pantalla el texto reconocido, si se obtiene un número de caracteres mayor del que la pantalla puede soportar se añaden puntos suspensivos al final del texto como se observa en la ilustración 51.

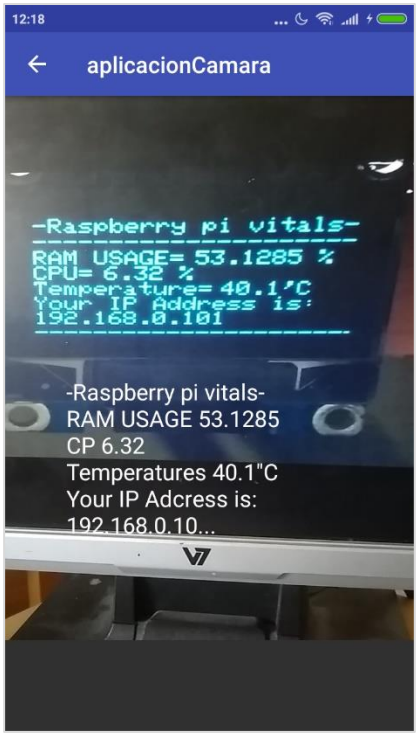

Ilustración 51: Modo Texto en Cámara

5.2.7 Galería y Detalle de ficheros

Desde estas pantallas se puede ver tanto las imágenes que han sido utilizadas para un reconocimiento de caracteres como los ficheros creados.

La galería muestra mediante el *RecyclerView* los diferentes grupos de ficheros sin entrar en detalle como se ve en la ilustración 52 de forma esquematizada. Desde la galería se ven las diferentes imágenes que han sido utilizadas para el reconocimiento que están almacenadas en el teléfono, han sido asociadas a su documento XML o de texto según el modo de reconocimiento utilizado.

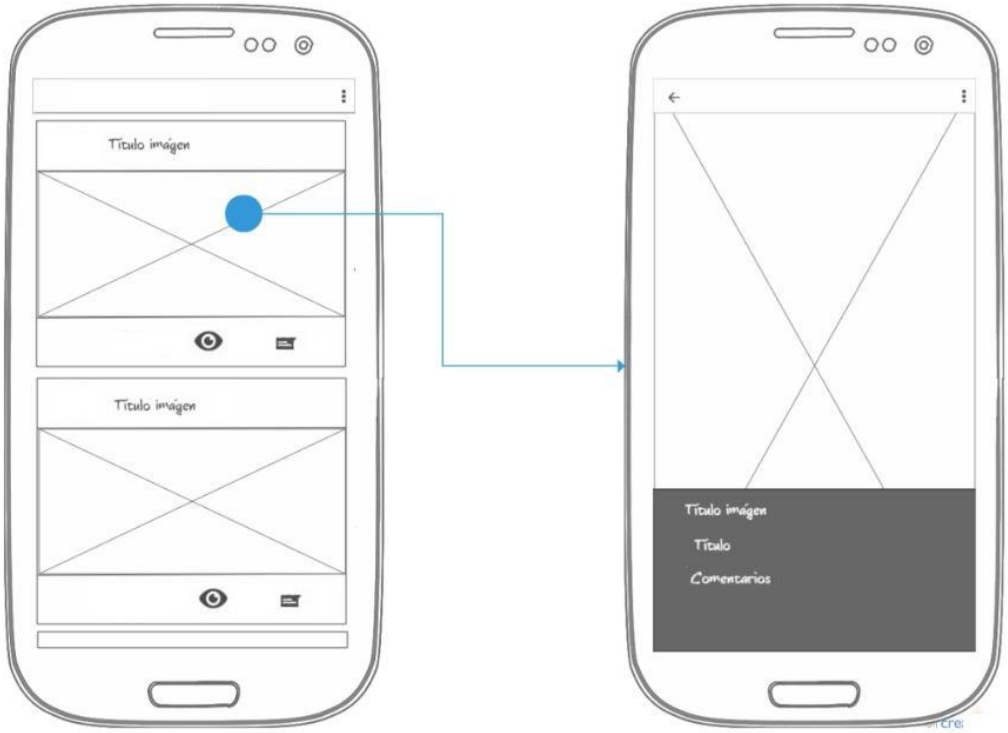

Ilustración 52: Ejemplo esquemático de Galería

Posteriormente al pulsar sobre un elemento, la aplicación muestra en diferentes pantallas la imagen a tamaño real (ilustración 53), con su nombre y el nombre del documento con la posibilidad de al pulsar un botón a la derecha del título del documento.

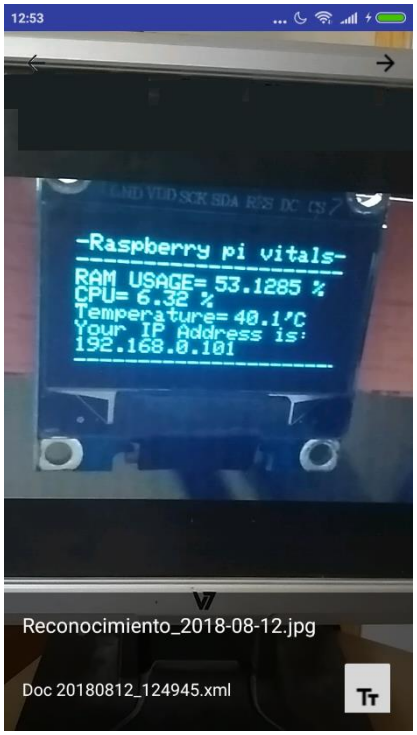

Ilustración 53: Imagen a Tamaño Real Galería

El inicio de otra pantalla, llamada pantalla detalle, donde se muestra el contenido del documento siguiendo el esquema de la pantalla anterior. Además, se incluye un *BottonNavigationView* para dar la posibilidad de enviar el documento y la imagen asociada a él a un servidor FTP. Como se observa en la ilustración 54.

Ilustración 54: Pantalla detalle XML

Si el usuario decide enviar un archivo mediante la opción del Enviar del *BottonNavigationView,* se comprueba la conexión a Internet y se procede la subida si la conexión es correcta (ilustración 55).

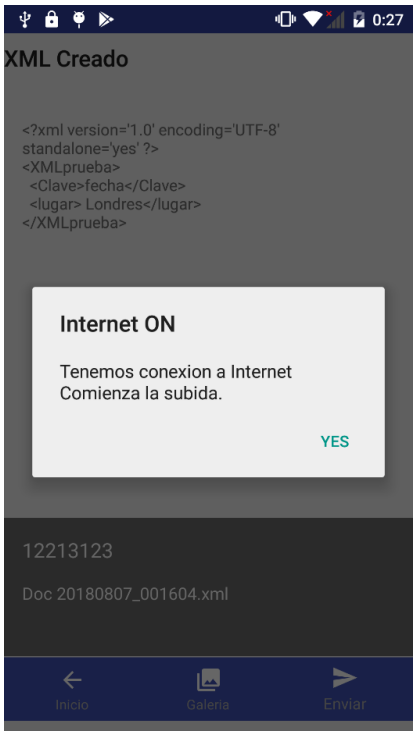

Ilustración 55: Pantalla de detalle conexión Internet

Como son dos ficheros, y la subida no es instantánea se procede a la aparición de una barra de porcentaje para esa subida y que el usuario pueda estar al corriente si se produjo un error o si la subida fue un éxito (ilustración 56).

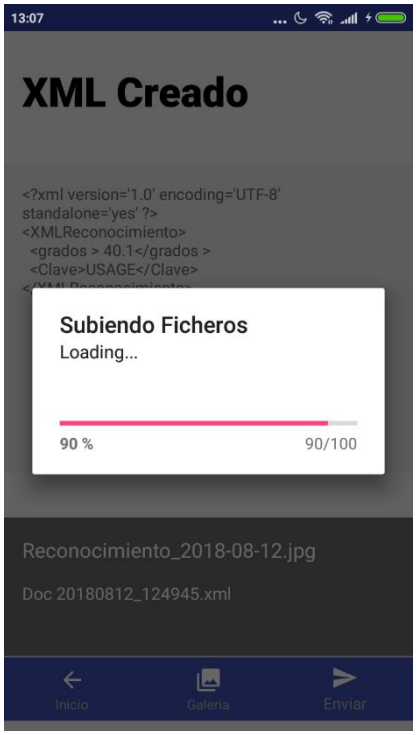

Ilustración 56: Subida de Ficheros a servidor FTP

5.2.8 Pantalla Acerca de y Ayuda

Son pantallas meramente informativas, la pantalla Acerca de donde se indica la versión desarrollada de la aplicación y la ventana Ayuda, se muestra diferente información respecto a las funcionalidades.

CAPÍTULO 6

6. Conclusiones y Líneas futuras

En este sexto capítulo se detallan las conclusiones obtenidas de la realización de este trabajo de fin de grado desde un punto de vista personal y profesional. Se señala la consecución de los objetivos propuestos en el primer capítulo y las líneas futuras que debe seguir este proyecto.

6.1 Conclusiones

A nivel profesional para lograr los objetivos del capítulo 1 se han seguido los siguientes pasos que se muestran a continuación:

En primer lugar, para el desarrollo de una aplicación móvil se procede a realizar un estudio previo de las diferentes opciones de desarrollo que se encuentran en la actualidad. Seguidamente viendo los requisitos generales que necesitaba la aplicación y estudiando los pros y contras de cada tecnología se eligió el desarrollo nativo, en concreto, el sistema operativo Android, entre otras cosas por su amplia presencia en el mercado. Seguidamente se busca un equilibrio entre funcionalidad y diseño siendo elegida la versión de Android *Kit Kat,* pero funcionando desde la versión 4.0.3

En segundo lugar, se busca información acerca OCR, se elabora un análisis de esta tecnología, viendo las diferentes bibliotecas y algoritmos de detección y reconocimiento para lograr las metas que se han exigido. Se elige la biblioteca ML Kit Firebase, siendo precisa en cuanto al nivel de ejecución y que puede facilitar la creación de notificaciones una vez la aplicación esté dispuesta para proceder a reconocer el texto que se encuentra en la imagen.

Con el estudio del estado del arte en el tercer capítulo se tiene una idea global de las aplicaciones existentes, añadiendo herramientas de edición para definir la estructura de la imagen y añadiendo diferentes métodos de captura de imágenes para el reconocimiento de texto viendo que estas aplicaciones actualmente no poseen.

En tercer lugar, una vez logrado las metas de recabar datos en tiempo real, se busca desarrollar una aplicación intuitiva y simple siguiendo las normas establecidas por Material Design de Android descritas en el capítulo del manual del programador. Para lograr una aplicación de calidad se crea una galería para mostrar las diferentes imágenes

con sus documentos creados y la posibilidad de una copia de respaldo mediante un servidor FTP.

A nivel personal y académico, la realización de esta aplicación ha sido muy efectiva en diversos aspectos.

La elaboración de esta aplicación afianzó los conocimientos que había adquirido al cursar la asignatura optativa *Desarrollo de aplicaciones para dispositivos móviles,* además de aprender más en profundidad del lenguaje de programación Java que tanto se demanda hoy en día a la hora de incorporarse en el mundo laboral. También recalcar la oportunidad de estudiar e investigar sobre la tecnología Android creando una inquietud de seguir aprendiendo sobre este lenguaje.

La preparación de este proyecto me ha enseñado a ser más autodidacta, ya que me he encontrado en situaciones con las que no había tenido la oportunidad de enfrentarme, para posteriormente confeccionar un análisis técnico de algún imprevisto que pudiese surgir durante el desarrollo de esta aplicación, destacando la implementación de la biblioteca *ML Kit de Firebase* de la cual, no se conoce mucha información debido a que fue lanzada hace apenas 2 meses.

Destacar también que se han aplicado los conceptos adquiridos en los años cursados en la carrera, especificando entre otros los conocimientos técnicos desarrollados en las asignaturas de *Sistemas en Tiempo Real* para los conceptos de los diferentes hilos de una aplicación, *Tecnologías para aplicaciones Web* en conceptos de técnicos como XML y hábitos de desarrollo de un proyecto de larga duración. En definitiva, este trabajo de fin de grado me ha ayudado a aumentar en gran medida mis conocimientos y perfeccionar aquellos hábitos y habilidades que ya poseía gracias a los diferentes trabajos elaborados durante la carrera.

6.2 Líneas Futuras

Lo visto a lo largo de este trabajo, se puede resumir en dos partes principales. Una parte es la comprensión del funcionamiento del algoritmo de reconocimiento óptico de caracteres y la otra es el desarrollo de una aplicación para *smartphones* que utilice esas funcionalidades de dicho algoritmo para crear ficheros editables. Una vez finalizado este trabajo se tiene una versión de la aplicación con diferentes funcionalidades de captura, procesado y reconocimiento.

No obstante, han surgido mejoras e ideas y se espera mejorar la calidad de las características que ya posee y la creación de otras nuevas para crear una interfaz más agradable a la hora de usar esta aplicación ya que la intención que el trabajo

desarrollado durante estos meses no se detenga aquí. Para versiones siguientes se pretende añadir las siguientes características:

- Para el uso de la aplicación de usuarios que no sean hispanohablantes la inclusión de diferentes idiomas en la aplicación sería una forma óptima de llegar a un mayor número de ellos.
- Implementar una base de datos en la nube, con un breve registro, para que el usuario pueda administrar los diferentes documentos que tiene aparte del almacenamiento en el propio dispositivo, ya que aporta mayor persistencia. Es una idea atractiva, buscando una base de datos en la nube que ofrezca asistencia sin conexión permanente a Internet y sin importar la latencia. Para adaptarse a estas necesidades la opción que aporta Google con Cloud Firestore sería una magnifica idea adaptada a los requisitos que necesita el usuario. Gracias a esta base datos se podrían crear consultas para comparar diferentes datos y crear gráficos sobre el texto analizado.
- En cuanto al diseño surgen nuevas opciones para que la aplicación sea más amigable como la inclusión de Material Desing 2 que se está introduciendo actualmente en las diferentes aplicaciones de Google para mejorar la experiencia del usuario y ser una aplicación de aspecto convincente mejorando aspectos básicos como la tipográfica o colores.
- Dar la posibilidad al usuario de compartir sus ficheros como opción en la galería vía Google Drive o Dropbox para su rápida comunicación con otros usuarios daría otro salto de calidad a la aplicación.
- En cuanto a la biblioteca utilizada de OCR de Google cabe la posibilidad de añadir las tipografías que aparecen en textos escritos, para ello habría que entrenar un motor OCR mediante redes neuronales de *Machine Learning* que sean capaces de reconocer las posibles tipografías.

Bibliografía

[1] Statista – The portal for statistics, https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/, junio de 2018.

[2] Ramíez Vique, Robert. *Métodos para el Desarrollo de Aplicaciones Móviles.* FUOC. Fundavio per a la Universitat Oberta de Catalunya. p. http://docplayer.es/3103691-Metodos-para-el-desarrollo-de-aplicaciones-moviles.html, junio 2018

[3] statcounter- GlobalStats, http://gs.statcounter.com Julio 2018

[4] Universidad Internacional de Valencia, *Desarrollo de aplicaciones multiplataforma*, https://www.universidadviu.es/desarrollo-de-aplicaciones-multiplataforma/, Julio 2018.

[5] –Computer Science at the University of Virginia, *Mobile Application Development,iOs Architecture*, Julio 2018

[6]- Jesús Tomas Girones*, El gran libro de Android, 4ª Edicion,2015*, Julio 2018

[7]- Joan Ribas Lequerica, *Desarrollo de aplicaciones para Android. Edición 2018*, Julio 2018

[8] Google Developers., https://developer.android.com, Julio 2018

[9] Jesús Tomas, Antonio Albiol, Mohamed Falhi, Vicente Carbonell, *Dispositivos Wearables, Visión Artificial, Google Glass y Android TV*., Julio 2018

[10] Georgi Dalakov, *history of computers*, https://history-computer.com/ModernComputer/Basis/OCR.html, Julio 2018

[11] Algoritmo tesseract, *github,* https://github.com/tesseract-ocr

[12]IEEE, *An Overview of the Tesseract OCR Engine*, https://ieeexplore.ieee.org/document/4376991/, Julio 2018

[13] Google I/O 2018, https://codelabs.developers.google.com/io2018, Julio 2018

[14] Android Studio,https://developer.android.com/studio/, Julio 2018

[15] Material Design, https://material.io, Julio 2018

[16]Jesús Tomas Girones, *página 155. El gran libro de Android 4º edición,* Julio 2018

[17] Biblioteca TFTP, http://commons.apache.org*,* Julio 2018

[18] ArthurHub, *github,* https://github.com/ArthurHub/Android-Image-Cropper, Julio 2018

[19] Biblioteca FAB button, *github*,https://github.com/futuresimple/android-floating-action-button, Julio 2018

[20] Biblioteca Glide, *github*, https://github.com/bumptech/glide*,* Julio 2018

[21] - Joan Ribas Lequerica, *Desarrollo de aplicaciones para Android, Página 310. Edición 2018*, Julio 2018

[22] W.Frank Ableson Robi Sen Chris King, *Android Guía para desarroladores,2ª edición*, Julio 2018