UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

# Desarrollo de una aplicación para medición de distancias con un dispositivo Android

Trabajo Fin de Grado

**Grado en Ingeniería Informática**

**Autor**: Francisco Maluenda Máñez

**Tutor**: Vicente Luis Atienza Vanacloig

2015/2016

# Resumen

El presente trabajo documenta el completo desarrollo de una aplicación Android para dispositivos móviles, cuya funcionalidad es la medición de distancias o tamaños de objetos. Se proponen diferentes técnicas para abarcar las distintas situaciones en las que el usuario requiera realizar una medida y no disponga de ningún instrumento a su alcance más que su propio teléfono móvil. Para conseguir este objetivo se hará uso de algunos de los sensores que incorporan la gran mayoría de los teléfonos móviles de hoy en día y muy destacadamente de su cámara. El desarrollo de la aplicación se fundamenta en el uso de OpenCV, una librería libre de visión artificial que ofrece una extensa funcionalidad para el tratamiento y análisis de imágenes.

**Palabras clave:** Android, OpenCV, medición, telémetro, regla.

# Abstract

This paper documents the complete development of an Android application for mobile devices, whose functionality is measuring distances or sizes of objects. Different techniques are proposed to cover the different situations where the user requires a measurement and does not have any instruments at its disposal than his own mobile phone. To achieve this goal, we will make use of some of the sensors that incorporate the vast majority of mobile phones today and very prominently of its camera. The application development is based on the use of OpenCV, a free library of artificial vision that provides extensive functionality.

**Keywords:** Android, OpenCV, measurement, telemeter, rule.

# Tabla de contenidos

# Tabla de ilustraciones

# 1. Introducción

En este primer capítulo nos centraremos en introducir los aspectos más básicos del proyecto, hablaremos de la motivación, justificaremos la elección de un proyecto como este y expondremos la necesidad de una aplicación de este tipo, para finalizar mencionando las metas que se han perseguido tanto a nivel profesional como personal.

## 1.1 Motivación

Hoy en día es cada vez más frecuente el uso de dispositivos móviles por una amplia gama de usuarios. Los dispositivos móviles han ido evolucionando hasta convertirse en una herramienta imprescindible para para cualquier individuo de la sociedad moderna, su uso fluctúa desde el ocio hasta el trabajo y el rango de edad de sus usuarios crece sin detenerse.

Puede parecer descabellada la elección de una aplicación para móviles como trabajo fin de grado, debido al actual estado de las tiendas de aplicaciones, que puede definirse perfectamente como abarrotado. Sin embargo, la mayoría de las aplicaciones que encontramos en dichas tiendas presentan una calidad deficiente y muchas de ellas están saturadas de publicidad hasta tal punto que se hace imposible su uso. Todo esto no hace más que acrecentar la necesidad de aplicaciones útiles y funcionales.

Debido a todo lo anteriormente mencionado, se puede justificar la elección de un proyecto como este como trabajo de final de grado, ya que las tecnologías empleadas pueden considerarse punteras en la actualidad y su aprendizaje resulta muy útil de cara a un futuro laboral.

## 1.2 Presentación del problema

La gran cantidad de funciones que pueden ser llevadas a cabo por los móviles de hoy en día puede dar una impresión errónea acerca de la complejidad de proveer a nuestro dispositivo con una nueva funcionalidad.

El problema que se desea solucionar con este proyecto consiste en dotar a nuestro dispositivo Android con la capacidad de realizar mediciones. El dispositivo deberá ser capaz de medir objetos en una escena mediante un conjunto de técnicas, descritas más adelante, que sean capaces de ajustarse al tipo de medición que se desee realizar.

Los dos grandes problemas que se hacen presentes a la hora de obtener las mediciones son: obtener una referencia en la imagen a partir de la cual poder obtener

distancias y la perspectiva de la imagen tomada, que puede hacer necesario aplicar correcciones sobre la escena para preservar la relación de las medidas.

## 1.3 Metas

Con el fin de llevar a buen término el proyecto, se ha empleado el lenguaje de programación Java en el entorno de trabajo Android Studio que utiliza como entorno de desarrollo integrado IntelliJ. Para las funciones más complejas relacionadas con el tratamiento de imágenes se ha utilizado una librería de visión por computador conocida como OpenCV, en su versión para Android.

Mediante la elaboración del proyecto se pretende incrementar la funcionalidad de dichos dispositivos mediante la implementación de una aplicación capaz de competir con las distintas herramientas típicas para la medición de distancias, desde una simple regla hasta los más modernos telémetros.

Para lograr una aplicación útil y completamente funcional se han seguido distintas técnicas o aproximaciones para la obtención de las medidas, a saber:

- La medición de objetos pequeños mediante su ubicación sobre el dispositivo o cercana a este.
- El cálculo de distancias y alturas mediante la obtención de ángulos por sensores y cálculos trigonométricos.
- Obtención de medidas de objetos en una escena mediante la inclusión de un objeto conocido.

A nivel personal se ha perseguido adquirir conocimientos de programación para dispositivos Android. Durante el transcurso del proyecto se han obtenido conocimientos de diversa índole referentes a la programación de dichos dispositivos, sin centrarse especialmente en los necesarios para el proyecto objeto de estudio, todo ello fruto del esfuerzo personal y partiendo desde cero.

También se ha querido satisfacer una curiosidad personal acerca del interesante mundo de la visión por computador o visión artificial, nuevamente se ha partido de cero y mediante el estudio de las librerías de OpenCV junto con las incontables lecturas de la interfaz de programación de aplicaciones (*application program interface* o API) de esta librería se ha conseguido adquirir una visión general de este campo.

## 1.4 Estructura de la memoria

En este punto explicaremos de una forma breve y concisa la estructura seguida para el correcto desarrollo de este documento, narraremos para cada apartado que tema o aspecto se trata en él, para que al lector le sea más fácil seguir la memoria o pueda centrarse solamente en los puntos que considere de interés.

En el siguiente capítulo (Objetivos) estableceremos todos los objetivos que deben ser cumplidos para considerar la aplicación un éxito. Los objetivos, al tratarse de una aplicación móvil, se ha decidido que principalmente hagan referencia a las tareas que debe ser capaz de llevar a cabo.

En el capítulo tres (Estado del arte), recorreremos el mercado de aplicaciones Android en busca de las aplicaciones que desempeñen labores similares a la nuestra, seleccionaremos las mejores en base a las puntuaciones de los usuarios y tras estudiarlas expondremos en qué aspectos resulta novedosa nuestra aportación.

En el cuarto capítulo (Tecnologías empleadas), nos centraremos en exponer las tecnologías y herramientas que se han usado para elaborar el proyecto, durante dicho capítulo se tratarán los aspectos más importantes de cada tecnología, que será necesario conocer para entender el funcionamiento de la aplicación. Empezaremos con una visión general de Android, veremos los paquetes empleados de OpenCV, para terminar con las herramientas que nos han permito crear los gráficos de nuestra aplicación.

Las matemáticas que se esconden tras nuestra aplicación se detallan en el quinto capítulo (Obtención de medidas), donde se explica de forma concisa los pasos a seguir para obtener mediciones, también se desglosan los cálculos que convierten los datos de entrada en medidas reales. Ejemplos reales se pueden observar en el capítulo ocho (Resultados) con diversas fotografías y capturas de pantalla que ayudan a comprender mejor el funcionamiento.

Los requisitos o funcionalidad del sistema se detallan el capítulo quinto (Funcionalidad), para conseguir mostrar la funcionalidad la forma más clara posible, se ha optado por dividir el capítulo en secciones que clasifican los requisitos, y éstos a su vez se muestran en forma de tabla, en este capítulo se muestra de forma ordenada qué capacidades y especificaciones se exigen a la aplicación.

El capítulo siete (Diseño) ofrece una amplia visión de todos los aspectos de diseño que se han seguido, veremos las tendencias seguidas para el desarrollo de la interfaz, la implementación de la persistencia de la aplicación y un esquema del flujo de ventanas.

Para cerrar la memoria, el último capítulo (Conclusiones) contiene una valoración subjetiva de todas las tecnologías empleadas, comentarios acerca de las experiencias obtenidas al trabajar con ellas y una valoración personal del proyecto y de la aplicación resultante.

# 2. Objetivos

En este apartado procederemos a enumerar cuáles son los objetivos principales del proyecto. El objetivo principal del proyecto es la elaboración de una aplicación robusta y funcional en Android para dispositivos móviles. La aplicación debe ser capaz de obtener medidas fiables de una escena mediante diferentes técnicas que se expondrán posteriormente.

Con el objetivo principal en mente, también podemos enunciar como objetivos las características funcionales que deseamos que tenga la aplicación:

· Desarrollar una aplicación capaz de gestionar imágenes, es decir, la aplicación ha de ser capaz de cargar y almacenar imágenes correctamente, así como proporcionar una correcta visualización.

· Aplicar modificaciones sobre fotografías, la aplicación permitirá aplicar transformaciones sobre las imágenes para obtener mediciones más precias, como podrían ser escalados o transformaciones perspectivas.

· Realizar mediciones sobre una fotografía tomada paralela al plano sobre el que queremos realizar la medición, mediante la inclusión en la escena de un objeto cuyas dimensiones son conocidas.

· Medir distancias sobre una fotografía en la cual incluimos un objeto del cuál conocemos sus dimensiones y previamente hemos almacenado, mediante la correcta aplicación de transformaciones sobre la imagen.

· Calcular la distancia entre el usuario y un objeto conociendo la altura del usuario, el ángulo de inclinación del dispositivo enfocando a la base de dicho objeto mediante una pre-visualización de la cámara.

· Calcular la altura de un objeto conociendo la altura del usuario, la distancia del usuario al objeto y el ángulo de inclinación del dispositivo enfocando a la parte más alta del objeto con la ayuda de la cámara.

· Obtener las dimensiones de objetos relativamente pequeños situándolos sobre el dispositivo y acotando con los dedos los extremos del objeto cuya separación deseamos conocer.

# 3. Estado del arte

En este capítulo nos centraremos en tratar aspectos referentes al desarrollo de aplicaciones móviles, trataremos diversos temas como pueden ser las tendencias actuales de desarrollo. Hablaremos también de diferentes aplicaciones que hay actualmente en el mercado, las cuales ofrecen funcionalidades similares a las que desarrollaremos en este proyecto y finalmente explicaremos qué aportamos de nuevo en el ámbito de las aplicaciones móviles para realizar mediciones.

Como ya se ha mencionado en la introducción, actualmente existe una gran cantidad de aplicaciones para dispositivos Android, y muchas de ellas capaces de realizar mediciones de una forma bastante eficiente, siendo la precisión la principal característica a mejorar, hecho que se demuestra por sí solo al observar las bajas puntuaciones que obtienen dichas aplicaciones en el mercado.

De todas las aplicaciones que podemos encontrar con una funcionalidad similar a la nuestra, destacan por la valoración de los usuarios las aplicaciones “Regla” y “Medir Y Alinear – 3D Plomada”, ambas nos ofrecen la capacidad de medir objetos usando la cámara como principal herramienta.

## 3.1 Innovación

Empezaremos marcando algunas deficiencias encontradas en las aplicaciones mencionadas anteriormente, no sólo las nombradas explícitamente, también las que realizan funciones similares. Hablaremos en términos generales porque los aspectos mejorables suelen ser comunes a todas ellas.

El principal inconveniente que se puede encontrar y también el más común, es la dificultad de uso de dichas aplicaciones, las interfaces son poco claras y confusas, y en algunos casos la curva de aprendizaje se vuelve inabordable.

La mayoría de aplicaciones se centra en una sola técnica para obtener medidas: trigonometría para calcular alturas o distancias, medir previamente algún objeto y sacar las medidas del resto a partir de ese... lo que hace que las aplicaciones pierdan funcionalidad y posean todas las desventajas y limitaciones que pueda presentar la técnica escogida.

La aplicación propuesta para este trabajo recoge varias técnicas que pueden ser usadas de manera independiente o conjunta para lograr una mayor funcionalidad y lo más importante, que el usuario pueda elegir la técnica que más se ajuste a sus necesidades.

Otra importante mejora que podemos encontrar en nuestra aplicación y que casi ninguna otra incluye, es la capacidad de tomar medias de una fotografía independientemente del ángulo desde el que haya sido tomada, es decir, la mayoría de las aplicaciones solamente permiten obtener mediciones si la foto has sido tomada

con el móvil totalmente paralelo al plano donde se sitúa el objeto de interés. Aquí proponemos una solución a este problema aplicando transformaciones sobre la imagen para corregir la perspectiva y obtener mediciones fiables.

# 4. Tecnologías empleadas

El presente capítulo trata de explicar qué tecnologías se han empleado en la elaboración de este proyecto. Hay que destacar que la gran mayoría de herramientas y tecnologías empleadas no son de creación propia, simplemente se han mejorado algunas de las existentes o se han combinado para obtener nuevos algoritmos o funcionalidades totalmente distintas para las que fueron creadas.

Las dos principales tecnologías empleadas son Android, que proporciona todo lo necesario para el desarrollo de aplicaciones móviles y OpenCV, una librería de visión artificial que proporciona algunas de las funciones necesarias para trabajar con imágenes. En los siguientes apartados se ofrecerá una visión más completa de ambas tecnologías.

## 4.1 Android

Android puede entenderse como una plataforma *software* cuya misión consiste en abstraer el *hardware* subyacente. Inicialmente fue desarrollado para dispositivos táctiles con recursos limitados a fin de facilitar el desarrollo de aplicaciones para dichos dispositivos.

La gran diferencia de Android respecto al resto de sistemas operativos para móviles es su núcleo basado en GNU/Linux. Esto hace que Android adquiera algunas de las principales características de Linux convirtiéndose en un software libre, gratuito y multiplataforma.

### 4.1.1 Historia

Android es un sistema operativo diseñado para funcionar principalmente en dispositivos táctiles desarrollado inicialmente por Android IC, fue comprado por Google en el 2005 y hasta entonces era un sistema operativo muy poco conocido.

En el 2007, lo que todavía era un rumor, se convirtió en cierto, se fundó la Opend Handset Alliance, una agrupación de empresas de desarrollo software, hardware y telecomunicaciones con el propósito de avanzar en los estándares abiertos para el desarrollo de *software* y *hardware* para dispositivos móviles.

Inmediatamente al anuncio de la Opend Handset Alliciance, se produjo la presentación de Android por parte de Google y se liberó gran parte de su código bajo una licencia Apache.

Inicialmente el éxito tras la presentación fue escaso, debido a que el sistema operativo se presentó antes de que se comercializara ningún dispositivo que lo incluyese. En la actualidad es el sistema operativo más utilizado.

### 4.1.2 Arquitectura

La arquitectura del sistema Android puede verse como una arquitectura por capas o niveles, de forma que cada nivel puede utilizar servicios ofrecidos por los niveles anteriores y éste, a su vez, proporciona nuevas funciones a los niveles superiores. A continuación, veremos que niveles lo componen.

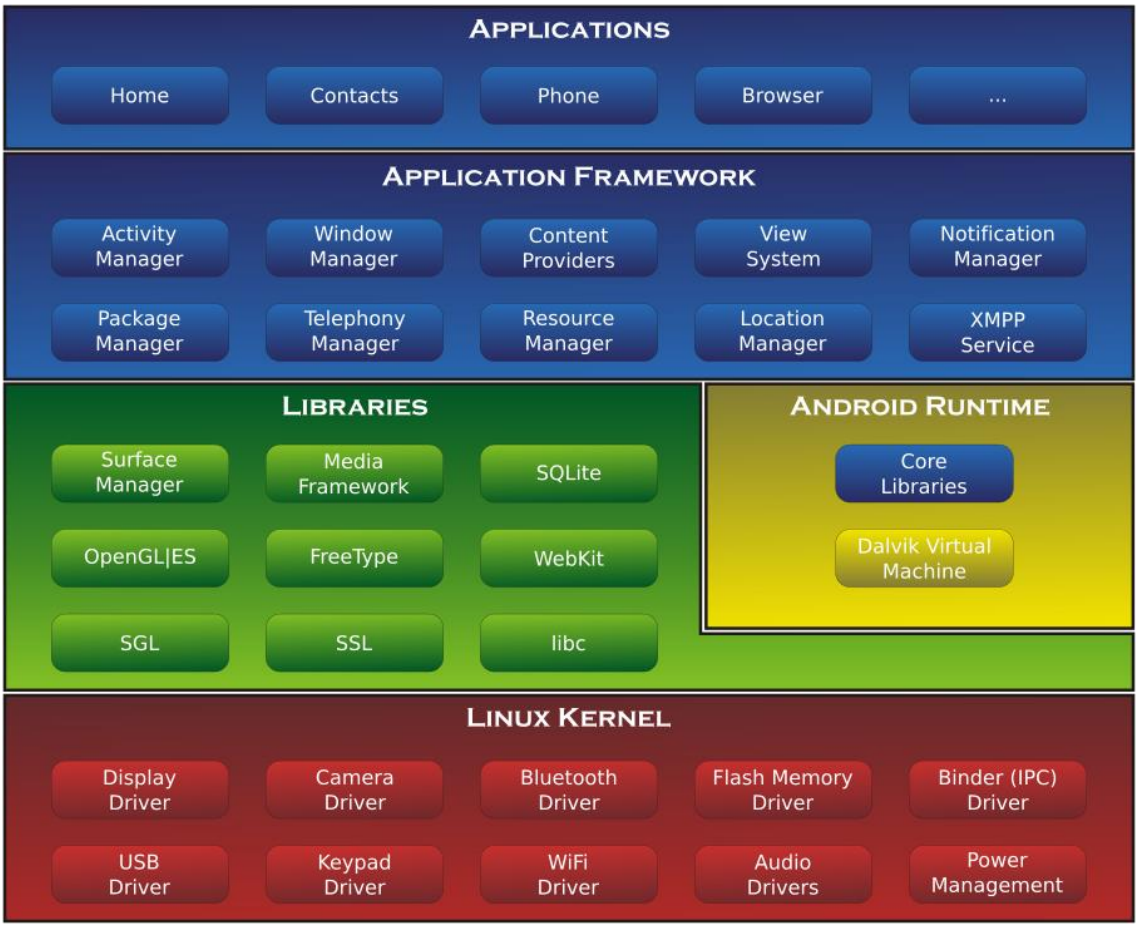



*Ilustración 1 - Arquitectura Android*

- Aplicaciones: Constituye el conjunto de aplicaciones presentes en un dispositivo, las instaladas por el usuario o por defecto, las nativas y las administradas.

- *Framework* de aplicaciones: Plataforma de desarrollo que facilita la reutilización de componentes, permite el acceso a los diversos servicios ofrecidos y al *hardware* de los dispositivos. Los más importantes son:
    - *Activity Manager:* Conjunto de APIs encargadas de gestionar el ciclo de vida de las aplicaciones.
    - *Window Manager:* Gestiona las ventanas de una aplicación mediante la librería *Surface Manager*.
    - *Telephone Manager:* Compendio de APIs que gestionan las funciones "básicas" de los teléfonos (llamadas, mensajería…).
    - *Content Provider:* Proporciona los mecanismos necesarios para la comunicación entre aplicaciones.
    - *View System:* Ofrece los elementos básicos necesarios para la construcción de interfaces.
    - *Location Manager:* Posibilita a las aplicaciones el acceso a la ubicación del dispositivo.
    - *Notification Manager:* Permite a las aplicaciones notificar al usuario información asociada a ciertos eventos que ocurran durante la ejecución de una aplicación.
    - *XMPP Service:* Colección de APIs para el uso de este protocolo de intercambio de mensajes basado en XML.
    - *Resource Manager:* Encargado de gestionar todos los elementos que forman parte una aplicación externos al código.
    - *Package Manager:* Gestor de todos los paquetes instalados en un dispositivo Android, permite la instalación de nuevos.

- Bibliotecas: Hacen referencia al conjunto de librerías presentes en Android, escritas en C/C++, proporcionan la mayoría de las características más representativas de esta plataforma. Las principales librerías que podemos encontrar son las siguientes:
    - *Surface Manager:* Gestión de la pantalla.
    - *Media Framework:* Reproducción de imágenes, vídeo y audio.
    - *SQLite:* Motor de bases de datos relacionales.
    - *WebKit:* Navegación web.

- *SGL:* Gráficos 2D.
- *Open GL/ES:* Gráficos 3D.
- *FreeType:* Renderizado de fuentes.
- *SSL:* Comunicación segura mediante *sockets*.
- *Libc:* Variante optimizada de C.

- Android *Runtime:* Al mismo nivel que las librerías encontramos el entorno de ejecución, está constituido por las librerías Java que forman el núcleo del lenguaje y la máquina virtual Dalvik (o ART para las versiones más modernas). Es necesario remarcar que Java se usa únicamente como lenguaje de programación, el código obtenido como resultado de compilar un programa Android no es compatible con el *bytecode* Java.

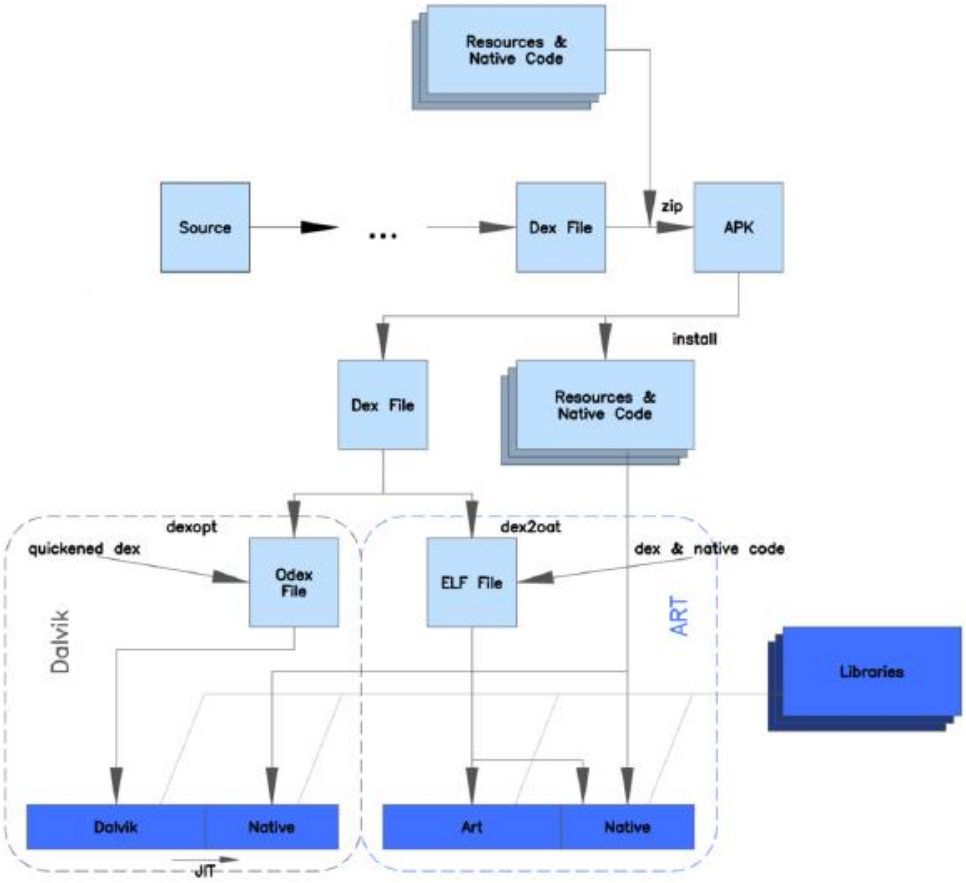



*Ilustración 2 - Android runtime*

- Núcleo Linux: Se utiliza como una capa de abstracción para el *Hardware* subyacente, contiene los *drivers* o controladores necesarios.

Por último, para cerrar nuestra visión general sobre la arquitectura Android y teniendo en mente que elementos la componen, vamos a ver a grandes rasgos como se produce la inicialización de dicha arquitectura. Los pasos a seguir se detallan a continuación.

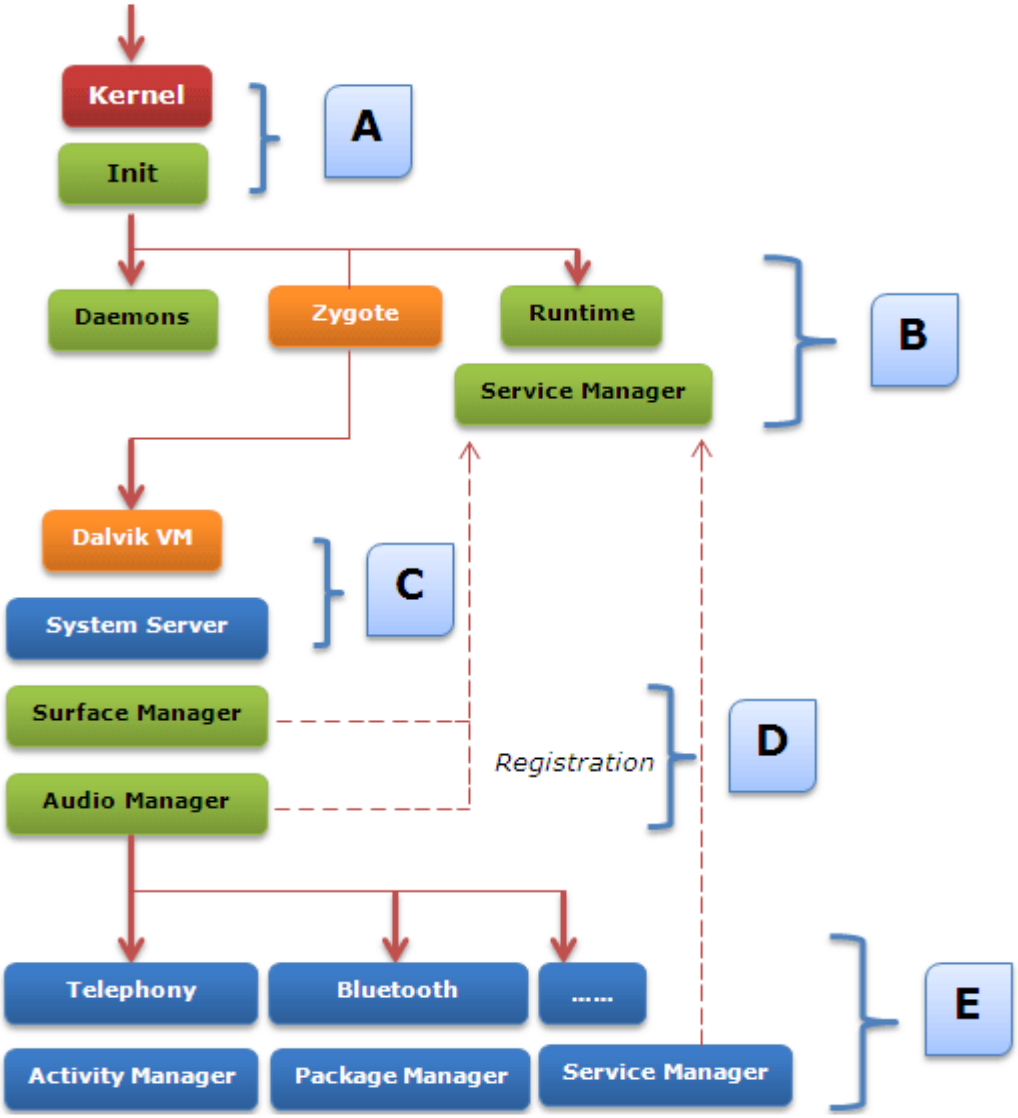



*Ilustración 3 - Inicio plataforma Android*

A. Inicialmente, el *bootloader* carga el *núcleo o kernel* y lanza el proceso *init*.

B. Se crean los demonios encargados de gestionar el *hardware*, inmediatamente después se crea el *Zygote,* que será la primera instancia de la máquina virtual y el *Runtime* inicia el gestor de servicios.

C. El proceso *Runtime* pide a *Zygote* que lance una nueva instancia de la máquina virtual para ejecutar el servidor del sistema.

D. Los dos primeros procesos ejecutados son los necesarios para gestionar el audio y la pantalla.

E. Por último, se lanzan el resto de procesos.

### 4.1.3 Aplicaciones

Este subapartado lo dedicaremos al estudio de las aplicaciones Android. Las aplicaciones Android están escritas en el lenguaje de programación Java y se implementan mediante las herramientas de desarrollo *software* (*software development kit,* SDK de ahora en adelante), que son las encargadas de recoger todo el código de la aplicación junto con todos los datos y recursos en un paquete Android o APK.

El APK se puede identificar por ser un archivo con extensión ".apk". Contiene toda la información necesaria para instalar una aplicación. Una vez instalada la aplicación su vida transcurre dentro de su propio entorno de seguridad o *sandbox*.

El sistema operativo Android puede definirse como un sistema operativo multi-usuario en el cual cada aplicación sería un usuario. Por defecto, el sistema identifica cada aplicación con un identificador único que es desconocido por la propia aplicación. Los permisos de cada aplicación se asignan por su identificador, de esta forma se garantiza que no se acceda a ficheros para los cuales no se tienen permisos.

Cada aplicación Android posee una máquina virtual propia, de forma que el código de cada aplicación se ejecuta de forma aislada del resto.

Por defecto cada aplicación se ejecuta en su propio proceso Linux. Android inicializa una aplicación cuando se requiere cualquiera de sus componentes, termina el proceso cuando ya no es necesaria o se requiere la memoria para otras aplicaciones.

De todo lo anterior, podemos deducir que Android crea un entorno de trabajo seguro, donde una aplicación solo puede acceder a la información que necesita para hacer su trabajo, este hecho se conoce como principio del mínimo privilegio.

Sin embargo, existen diversas formas mediante las cuales una aplicación puede compartir información con otra aplicación y acceder a los diferentes servicios que nos brinda el sistema. La existencia de dos aplicaciones con un mismo identificador Linux es posible, de esta forma una aplicación podrá acceder a los archivos de la otra y

viceversa. Con objeto de ahorrar recursos, las aplicaciones con mismo identificador también podrán compartir máquina virtual. Las aplicaciones también pueden acceder a recursos del dispositivo solicitando los pertinentes permisos al usuario.

#### *4.1.3.1 Componentes*

Los componentes de una aplicación son los elementos esenciales que la componen. Cada componente supone un punto de entrada mediante el cual el sistema pude acceder a la aplicación, aunque no todos suponen puntos de entrada para el usuario.

Cada componente es un bloque básico que desempeña un papel específico en el funcionamiento de nuestra aplicación. Existen cuatro tipos diferentes de componentes, cada uno de ellos con un ciclo de vida y propósito único. Los componentes son los que siguen a continuación:

- Actividades o *activities:* Cada actividad representa una pantalla de nuestra aplicación. Las actividades trabajan en conjunto para dar una visión coherente de la aplicación, sin embargo, la vida de cada actividad es independiente del resto. El ciclo de vida que sigue una actividad se muestra en la imagen siguiente.

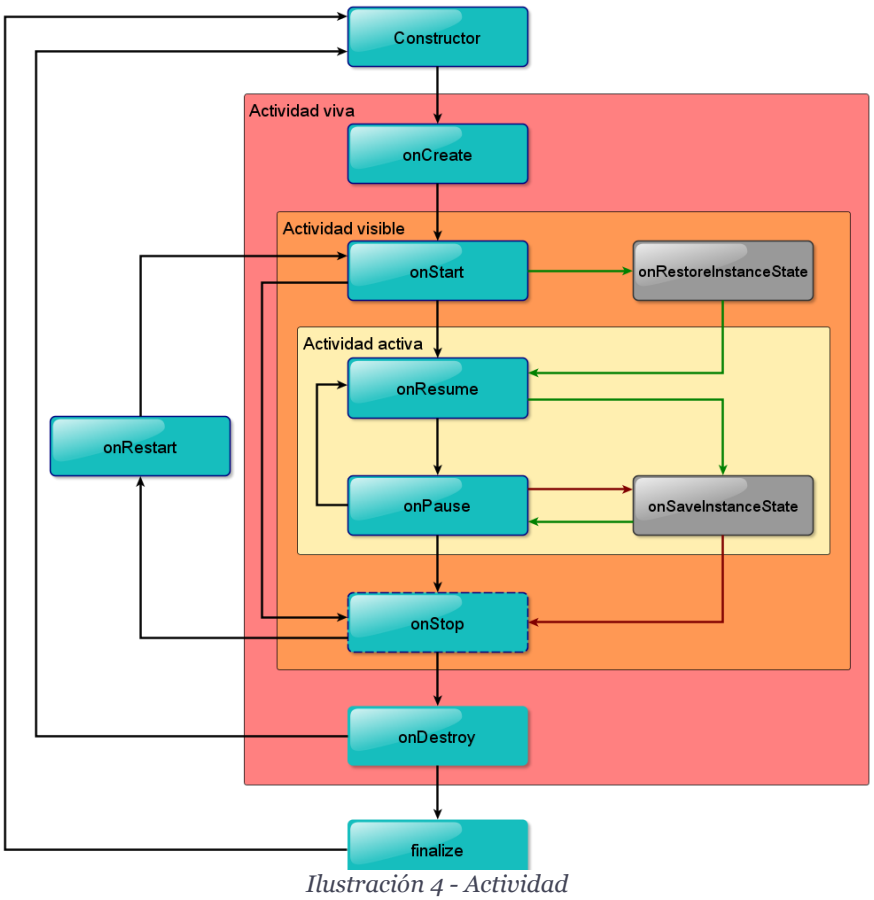



*Ilustración 4 - Actividad*

- Servicios o *services:* Son componentes que se ejecutan en segundo plano a fin de realizar operaciones de larga duración o procesos remotos. Los servicios carecen de interfaz y generalmente son llamados por las actividades para realizar tareas costosas sin bloquear la interfaz, se pueden dejar en ejecución en segundo plano o ligarlos a una actividad para interactuar con ella. En la siguiente imagen veremos el ciclo de vida de un servicio, a la izquierda actuando en segundo plano y a la derecha ligado a una actividad.

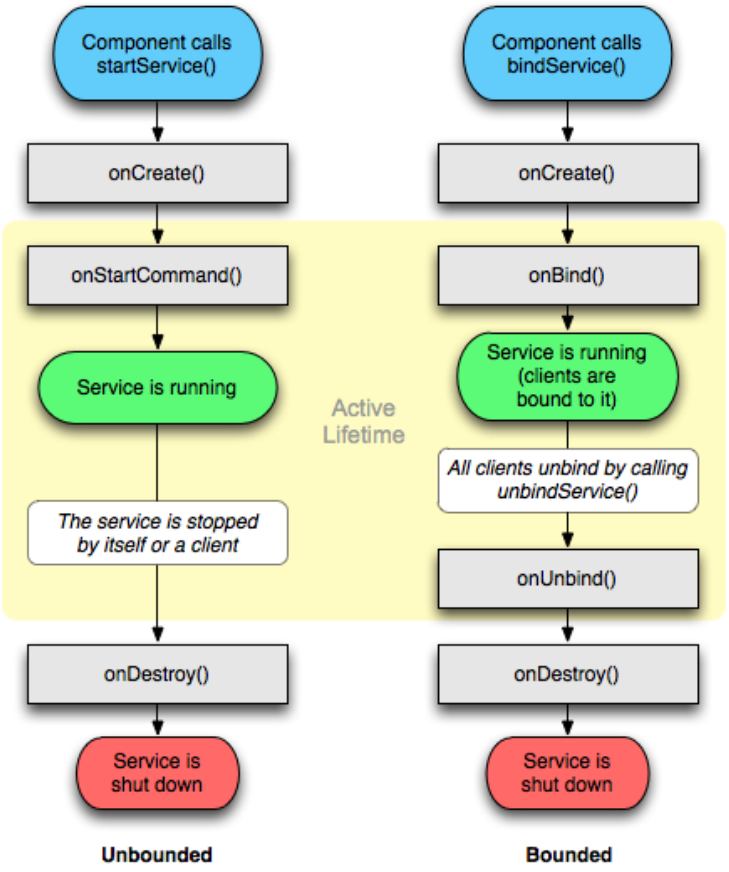



*Ilustración 5 - Servicio*

- Proveedor de contenido o *content provider:* El proveedor de contenido gestiona los datos que maneja una aplicación. Controla el acceso a archivos, bases de datos SQLite... mediante el proveedor de contenidos las aplicaciones pueden consultar y modificar datos de otras aplicaciones siempre que posean los permisos adecuados. La siguiente imagen ilustra cómo una actividad puede consultar datos de otra aplicación.

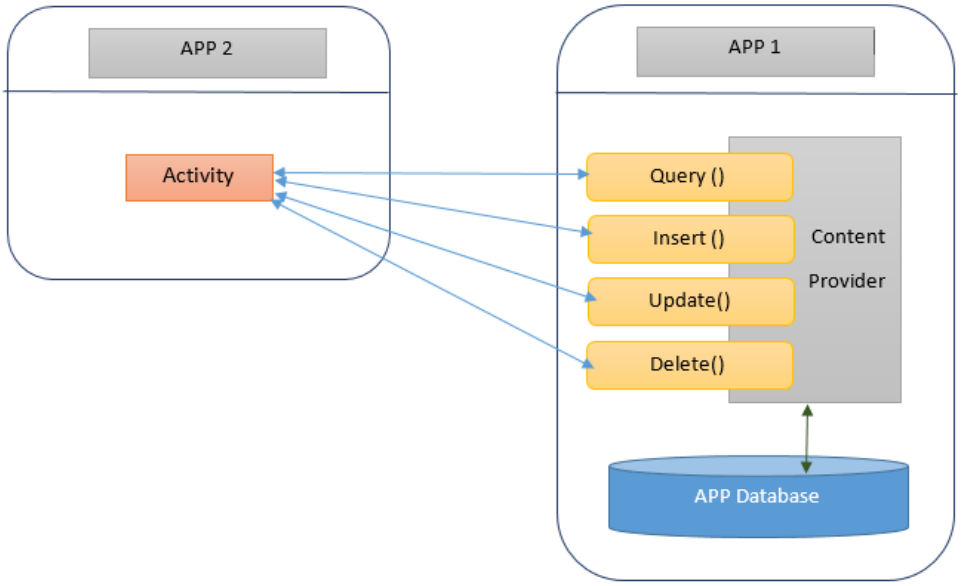



*Ilustración 6 - Proveedor de contenido*

- Receptor de difusiones o *broadcast receiver:* Es el componente encargado de responder a las difusiones o anuncios del sistema (por ejemplo, un anuncio del sistema que indica que queda poca batería). Carece de interfaz, pero debe crear una barra de estado para notificar al usuario cuando detecta una difusión. Una aplicación debe registrar un *broadcast receiver* para indicar que difusiones le interesan.

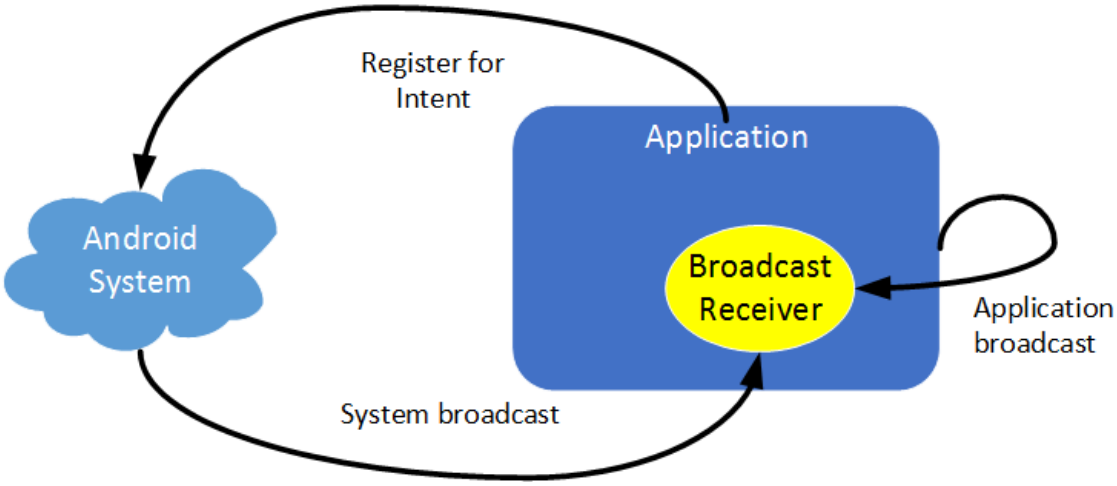



*Ilustración 7 - Receptor de difusiones*

#### *4.1.3.2 Recursos*

Los recursos de una aplicación corresponden con todos los ficheros, imágenes, cadenas de texto... que nuestra aplicación utiliza. Cuando trabajamos desarrollando

aplicaciones para Android, las buenas prácticas nos dicen que este tipo de archivos debe mantenerse independiente al código de la Aplicación.

Este hecho, conocido como externalización de recursos, permite adaptar un mismo código a dispositivos con diferentes configuraciones y características físicas de una forma prácticamente automática. En la siguiente imagen podemos comprobar cómo quedaría una pantalla que no ha sido adaptada para diferentes dispositivos (parte superior) y una que sí (parte inferior).

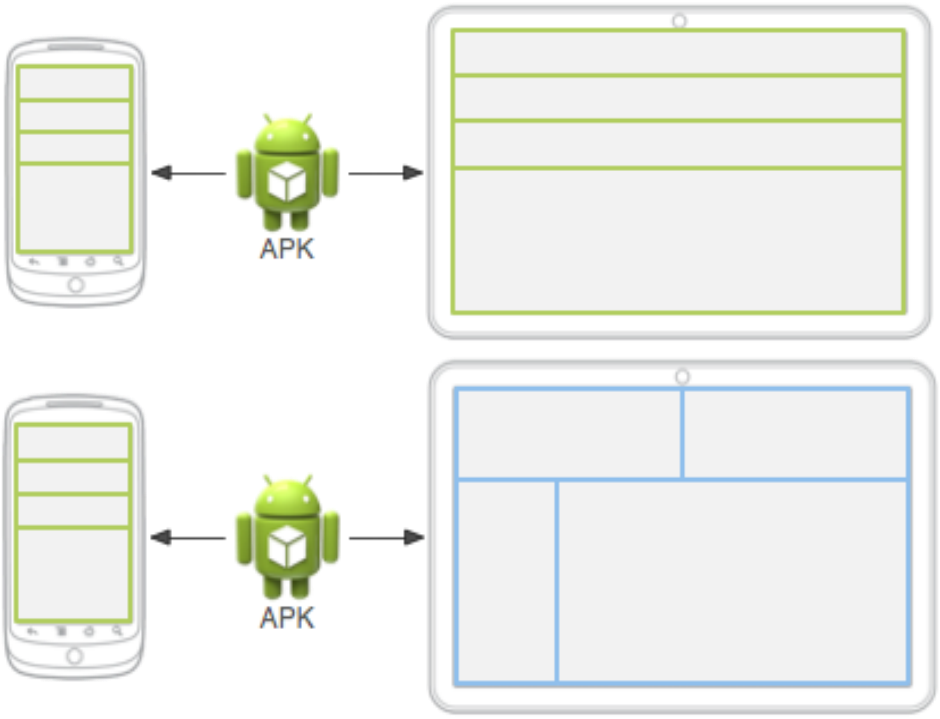



*Ilustración 8 - Externalización de contenido*

#### *4.1.3.3 Manifiesto*

El Manifiesto o *Manifest* es un archivo XML que contiene información esencial para Android acerca de la aplicación. Todas las aplicaciones deben tener un archivo AndroidManifest.xml en su directorio raíz.

Entre todas las funciones que realiza el manifiesto, las más significativas son las siguientes:

- Da nombre al paquete Java de la aplicación.
- Describe los componentes y qué proceso los hospeda.
- Contiene la declaración de permisos.
- Lista las librerías necesarias.
- Declara el nivel mínimo de Android API requerido.

### 4.1.4 Android Studio

Android Studio es el entorno de desarrollo integrado para el desarrollo de aplicaciones Android basado en ItelliJ IDEA. Puede considerarse un de los mejores editores de código basados en ItelliJ.

Entre las muchas ventajas que nos ofrece podemos contar con una vista ordenada y modular de los archivos que componen nuestro proyecto. La vista mostrada no se corresponde con la forma original de los archivos en disco, está pensada para optimizar el trabajo.

Integrado en nuestro entorno de trabajo, podemos encontrar un sistema para el control de versiones que incluye soporte para los repositorios más conocidos (Git, Subversion, Mercurial...).

La interfaz gráfica está divida de forma inteligente para mostrar una gran cantidad de información de forma eficiente, es totalmente personalizable permitiendo al usuario mostrar y ocultar información según se requiera. Android estudio sigue el contexto de trabajo del usuario mostrando las ventanas con las herramientas que considera oportunas en cada instante. A continuación, se muestra un ejemplo de interfaz gráfica:

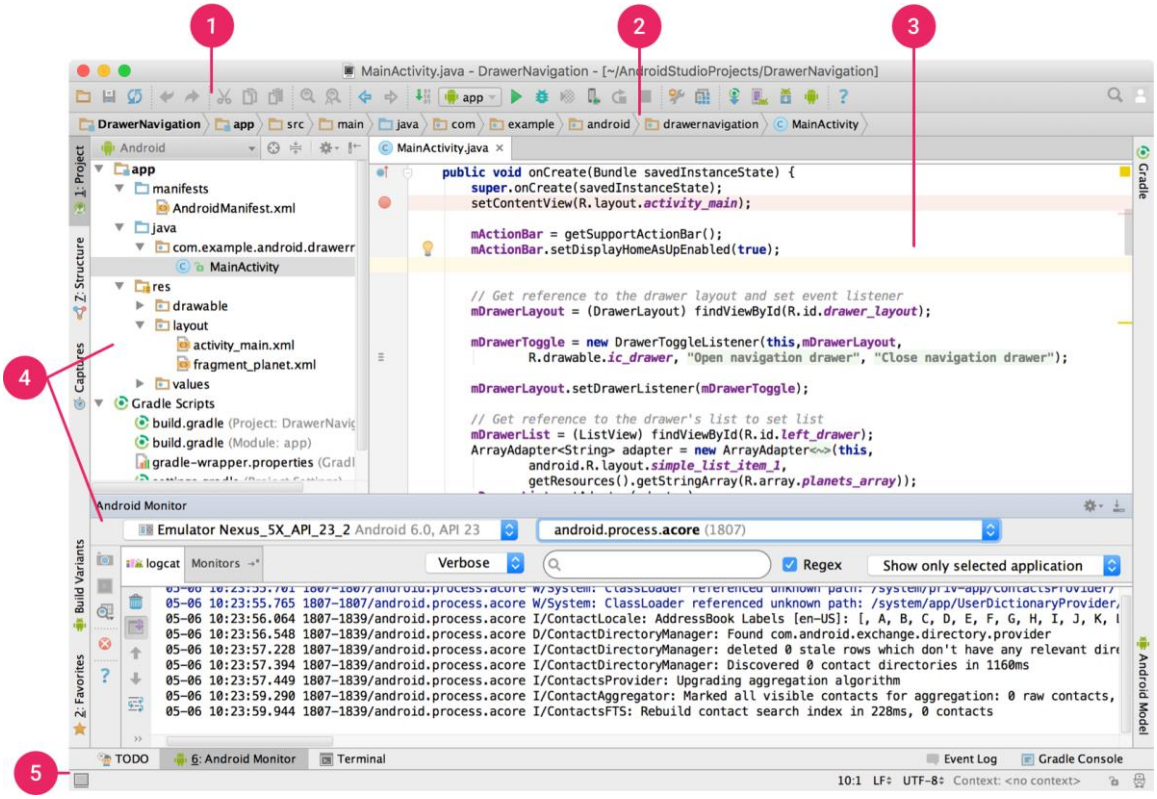



*Ilustración 9 - Interfaz gráfica de Android Studio*

1. Barra de herramientas o *toolbar:* Permite realizar una amplia gama de acciones.
2. Barra de navegación o *navigation bar*: Permite navegar entre los archivos abiertos para facilitar su edición.

3. Ventana de edición o *editor window:* Permite modificar los diferentes archivos.
4. Ventana de herramientas o *tool window*: Proporciona acceso a la gestión del proyecto, control de versiones…
5. Barra de estado o *status bar*: Muestra el estado del proyecto, así como mensajes y avisos.

Para finalizar nuestra visión de Android Studio, hablaremos de Gradle. Gradle es un sistema abierto que automatiza la compilación del código, utiliza un grafo dirigido acíclico para determinar en qué orden se pueden ejecutar las tareas. Android Studio incorpora esta tecnología para facilitar la reutilización de código permitiendo generar APKs con distintas funcionalidades partiendo de un mismo proyecto.

### 4.1.5 SDK - Software development kit

Android SDK o *software development kit* es un conjunto de herramientas para desarrollar, compilar y depurar aplicaciones para el sistema operativo Android, en definitiva, es una API que además de herramientas para el desarrollo, proporciona soporte técnico, ejemplos y una buena documentación.

El SDK incluye un emulador de dispositivos Android, lo que permite probar las aplicaciones de forma rápida y eficiente. Los parámetros de nuestro emulador pueden configurarse de forma que podemos elegir las características que tendrá nuestro dispositivo, como la memoria principal, el tamaño del dispositivo o la versión de Android, lo que permite al programador probar su código en una infinidad de dispositivos sin necesidad de adquirirlos.

### 4.1.6 NDK - Native development kit

Android NDK o *native development kit* provee las herramientas necesarias para desarrollar código Android en C y C++, frente al SDK que se programa usando el lenguaje Java.

Inicialmente el NDK no está recomendado para programadores sin experiencia en Android ya que incrementa notablemente la complejidad del código y tiene poca utilidad para la mayoría de aplicaciones.

El NKD puede utilizarse para crear nuevo código o reutilizar el ya existente, y pese a lo anterior dicho, existen dos situaciones que justifican su uso: exprimir al máximo la capacidad de cómputo de los dispositivos o utilizar bibliotecas de otros desarrolladores escritas C/C++, ambas situaciones se presentan en este proyecto mediante el uso de la librería OpenCV.

## 4.2 OpenCV

OpenCV es una librería que proporciona una gran cantidad de funciones relacionadas con la visión por computador y aprendizaje automático. Inicialmente fue desarrollada por Intel y actualmente podemos decir que se trata de una biblioteca libre bajo licencia BSD (*Berkeley Software distribution*).

Esta librería es multiplataforma, existiendo versiones para los sistemas operativos más conocidos: GNU/Linux, Mac OS y Windows. En este trabajo utilizaremos su versión para desarrollo Android conocida como OpcenCV4Android.

OpenCV cuenta con más de 2500 algoritmos implementados de forma muy eficiente, sin embargo, para el presente proyecto solamente se utilizarán aquellos relacionados con el tratamiento de imágenes. Los aspectos más tratados con esta librería son los referentes al reconocimiento de formas y objetos en una imagen, y la manipulación espacial de imágenes. Los módulos empleados para tales fines son:

- Core: Proporciona las estructuras de datos básicas usadas por el resto de módulos.
- Improc: Módulo de procesamiento de imágenes; permite el calcular y aplicar transformaciones perspectivas sobre imágenes.
- Utils: Contiene las herramientas necesarias para convertir los tipos de datos propios de Android en los que utiliza OpenCV y viceversa.
- Features2D: Implementa los algoritmos utilizados para el reconocimiento de objetos en una imagen.

## 4.3 Tecnologías para elementos gráficos

Este apartado recogerá aplicaciones de terceros que se han empleado para elaborar algunos elementos gráficos que encontramos en nuestra aplicación; los iconos del menú principal, el icono de la aplicación y algunos esquemas con cálculos trigonométricos presentes en la memoria.

La primera aplicación a tratar es Paint Net, una versión más completa y novedosa del clásico Paint que encontramos instalado por defecto en todos los sistemas Windows. Principalmente se ha usado para diseñar los elementos gráficos anteriormente mencionados.

La siguiente herramienta es Online Image Editor, un editor de fotografías online muy sencillo de usar y de acceso gratuito. Este editor se ha usado para escalar las imágenes y añadir transparencias donde fuese necesario.

# 5. Obtención de medidas

Ya se ha mencionado en repetidas ocasiones que la finalidad de esta aplicación es la de realizar mediciones, en este capítulo trataremos de explicar las diferentes técnicas empleadas para conseguirlo, es necesario señalar que una gran mayoría de dispositivos móviles carecen de los sensores adecuados para este tipo de cálculo, por lo tanto, se ha optado por simularlos con los sensores que sí incorporan muchos de los dispositivos actuales.

Para cada técnica implementada, explicaremos qué sensores intervienen, de qué forma el usuario las puede utilizar y mostraremos los cálculos necesarios para convertir los datos de entrada en una distancia.

## 5.1 Pantalla como referencia

La primera técnica propuesta, consiste en valernos de la pantalla de nuestro dispositivo para hacer mediciones, para ello lo primero que debemos hacer es ubicar un objeto sobre la pantalla o muy cercano a un lateral del dispositivo, para realizar la medición solamente tendremos que acotar con los dedos los dos extremos cuya distancia deseamos conocer.

Para logar este comportamiento serán necesarios varios elementos, para poder dibujar el trazo que servirá de cota, se ha implementado una vista personalizada cuya que extiende la clase *View*, en esta clase se han redefinido los métodos *onDraw(Canvas canvas)* y *onTouchEvent(Event event)* y los constructores. Finalmente se crea una instancia y se añade al *fragment* correspondiente para que se muestre de fondo.

El método *onTouchEvent* se ha modificado para que capture los toques sobre la pantalla solamente si se producen dos a la vez, de esta forma el usuario puede valerse de dos dedos para establecer los extremos a medir, al recibir los dos toques, se guardan las coordenadas de ambos puntos.

El método *onDraw* se ha modificado para que dibuje una línea vertical entre los dos puntos, y dos líneas perpendiculares en los extremos para que ayuden a la obtención de las medidas.

La siguiente imagen ilustrará la idea básica que se ha perseguido y servirá de orientación al resultado final esperado:

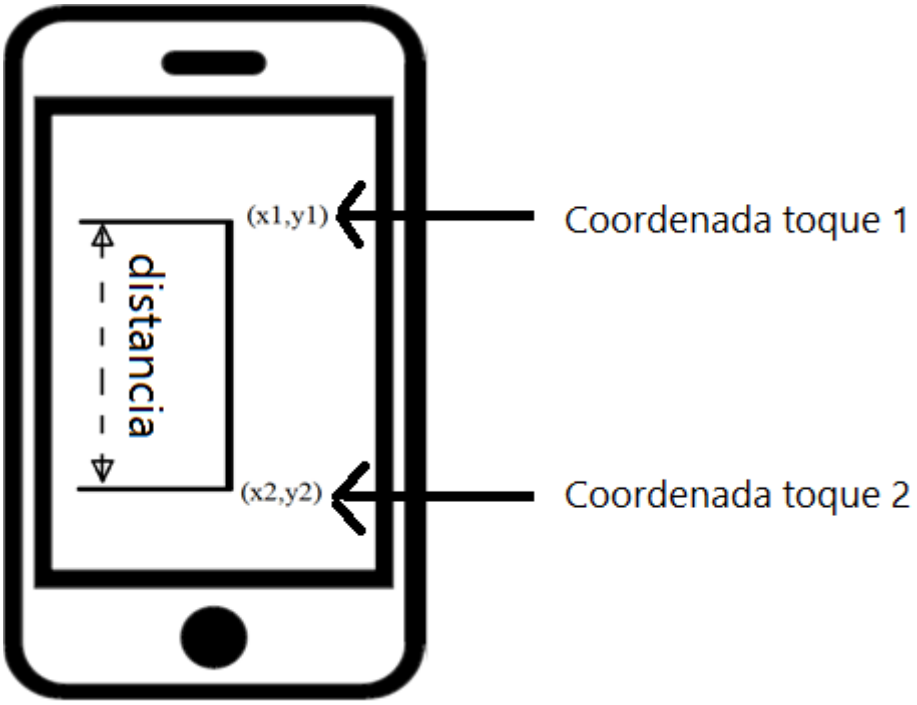



*Ilustración 10 - Mockup regla*

Por último, para obtener la distancia real, es necesario obtener la distancia en píxeles de los dos extremos:

$$distancia_{pixel} = \sqrt[2]{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Con el objetivo de reducir los cálculos se ha optado por una simplificación, la cota de medición es forzada a ser una línea vertical, esta simplificación no modifica en absoluto la funcionalidad de la aplicación y permite realizar los cálculos de manera más eficiente. Se ha escogido que sea vertical simplemente por ser la dimensión más grande de los dispositivos móviles. Para evitar errores de cálculo, se ha bloqueado la orientación del dispositivo, de esta forma, el usuario puede girar el móvil y obtener una medición exacta. La nueva distancia se calcularía:

$$distancia_{pixel} = |\, y_1 - y_2 \,|$$

Ahora solo es necesario convertir la distancia en píxeles a centímetros, para ello obtenemos la densidad de píxeles por pulgada en el eje vertical de nuestro dispositivo, a continuación, mostramos una forma de hacerlo:

```
DisplayMetrics metrics = new DisplayMetrics();
getActivity().getWindowManager().getDefaultDisplay().getMetrics(metrics);
float ydpi = metrics.ydpi;
```

Una vez conocemos la densidad de píxeles en el eje vertical, aplicamos una transformación para convertir la distancia en píxeles en una distancia en centímetros:

$$distancia_{cm} = \frac{distancia_{pixel} * 2{,}54}{ydpi}$$

Llegados a este punto solamente falta mostrar la distancia en centímetros y para ello nos valdremos de un *TextView*.

La medida máxima que podremos alcanzar con esta técnica vendrá dada por el largo de nuestro teléfono móvil, y será la principal limitación que podremos achacarle. Si el objeto que deseamos medir excede a nuestro dispositivo, será necesario usar algunas de las técnicas que se describen a continuación.

## 2.2 Medición basada en una fotografía

Esta técnica consiste en obtener mediciones de objetos que aparecen en una fotografía. Sobre la fotografía marcaremos referencias, objetos cuyas medidas reales conocemos, y que posteriormente servirán para para calcular las medidas del objeto de interés. Para elaborar esta aproximación seguiremos dos variantes que pueden parecer muy similares entre sí, pero realmente la complejidad detrás de ellas es muy diferente y las prestaciones que nos ofrecen también.

### 5.2.1 Fotografía en paralelo

Los elementos necesarios para llevar a cabo esta técnica pueden considerarse mínimos. En primer lugar, necesitaremos una imagen en la que aparezcan en un mismo plano tanto el objeto a medir como el elemento de referencia con medidas reales conocidas. Pondremos esta imagen de fondo de pantalla, para ello obtendremos la ruta de acceso a la imagen mediante la galería y la visualizamos con la ayuda de un *ImageView*. Por último, necesitaremos una clase personalizada que herede de la clase View, le pondremos un fondo transparente y la ubicaremos sobre el *ImageView* que contiene la foto, de esta nueva clase sobrescribimos los constructores y los métodos *onDraw(Canvas canvas)* y *onTouchEvent(Event event)*.

El método onTouchEvent nos permitirá almacenar los puntos sobre los que el usuario pulse y almacenarlos, permitiendo al usuario arrastrar el punto sobre la imagen para conseguir una mayor precisión.

El método *onDraw* se sobrescribe para dibujar líneas, los dos primeros puntos almacenados servirán de referencia y se pintará una línea que los una para mejorar el *feedback*, lo mismo sucederá con los dos siguientes puntos, pero esta vez marcarán la distancia que deseamos medir.

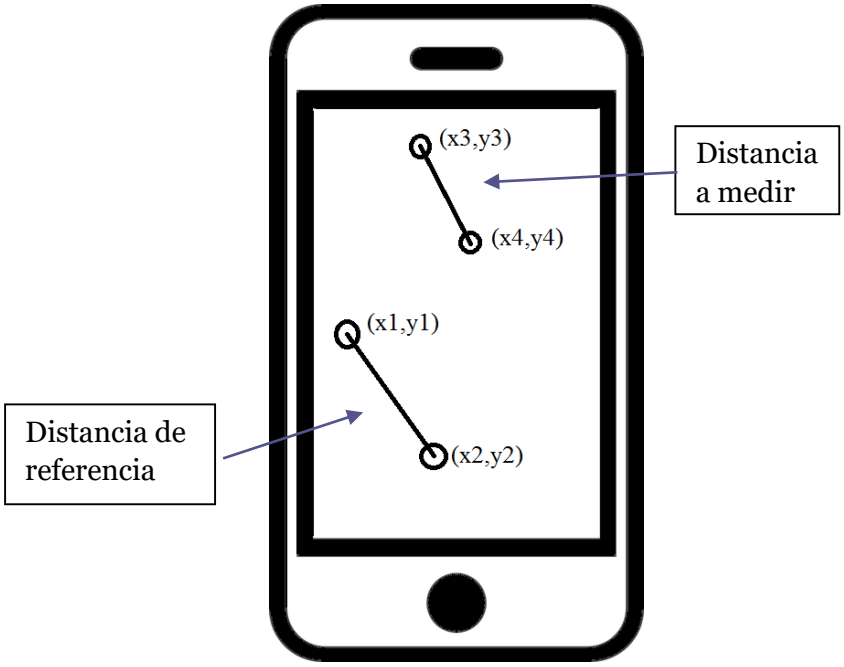



*Ilustración 11 - Mockup Fotografía en paralelo*

Una vez tenemos la referencia y el objeto a medir señalado, se preguntará al usuario por la medida real de la referencia. Para obtener la media real del objeto simplemente hay que seguir dos simples pasos, obtener la distancia de referencia y la del objeto en píxeles, y obtener la medida real del objeto. Las unidades resultantes coincidirán con las introducidas por el usuario (x1, x2, y1, y2 hacen referencia a las coordenadas de los extremos de la referencia y x3, x4, y3, y4 a las cotas del objeto)

$$referencia_{pixel} = \sqrt[2]{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$$objeto_{pixel} = \sqrt[2]{(x_3 - x_4)^2 + (y_3 - y_4)^2}$$

$$objeto_{uds} = \frac{objeto_{pixel} * medidaReal}{referencia_{pixel}}$$

Una vez obtenida nuestra medida podemos tomar varias decisiones, obtener otra medida con la misma referencia, o empezar desde el principio redibujando la referencia para intentar obtener una medida mejor.

Esta aproximación cuenta con dos limitaciones, en primer lugar, la fotografía debe ser toma con el móvil situado de forma paralela al plano donde se encuentran

los objetos para evitar que la perspectiva deforme dichos objetos; en segundo lugar, tanto la referencia como el objeto deben estar situados en el mismo plano, para que las medidas mantengan la proporción.

### 5.2.2 Fotografía sin perspectiva

La siguiente técnica es muy similar a la anterior, pero en esta haremos uso de más herramientas a fin de eliminar alguna de sus limitaciones. En primer lugar, cargaremos la imagen de la galería y reduciremos su tamaño hasta que quepa en la pantalla. La imagen se visualizará sobre un *ImageView,* y sobre este añadiremos otra vista personalizada con los métodos *onDraw(Canvas canvas)* y *onTouchEvent(Event event)* sobrescritos. Nuevamente el método *onTouchEvent* servirá para gestionar los toques sobre la pantalla y el método *onDraw* para marcar los toques y mejorar el *feedback*.

Para realizar los cálculos lo primero que necesitamos es tener un objeto cuyas medidas conozcamos; serán necesarias varias medidas, preferiblemente el ancho y el largo e incluir dicho objeto en la escena.

Una vez realizada la fotografía, el usuario tendrá que localizar en ella el objeto marcando los vértices de un polígono que la incluya (el polígono debe estar los más ajustado posible, se almacenan las coordenadas de sus vértices), mediante OpenCV (haciendo uso de la función *getPerspectiveTransform()*), obtendremos una matriz de transformación (M) que convertirá el objeto referencia en, en ese mismo objeto, pero con las medidas reales (previamente almacenadas en los ajustes). De esta forma la matriz obtenida es una matriz que, al multiplicarla por el objeto en la foto, obtendríamos el objeto con las dimensiones reales. Análogamente, al multiplicar

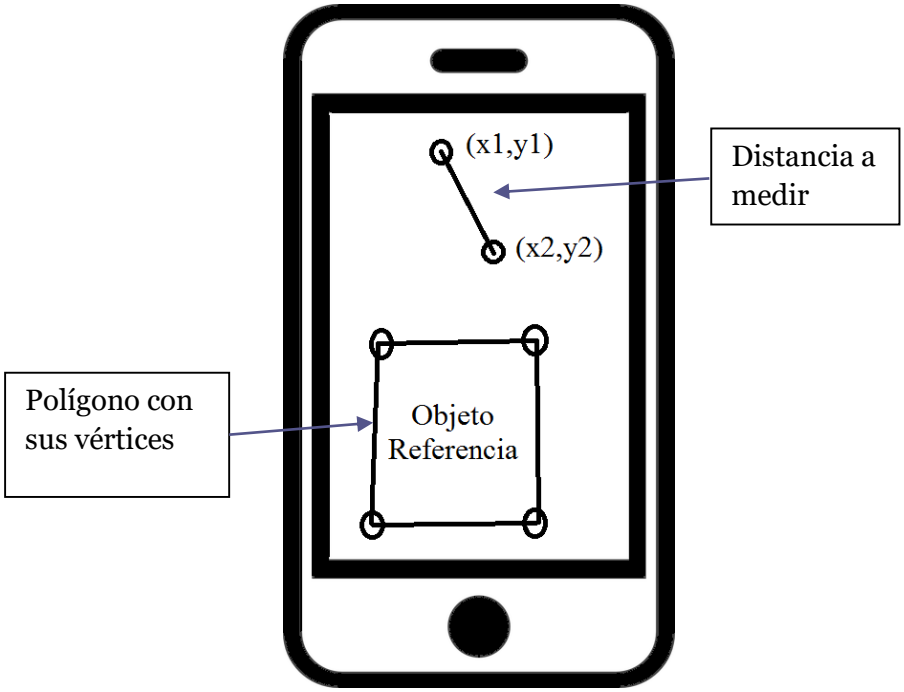



*Ilustración 12 - Mockup fotografía sin perspectiva*

cualquier par de píxeles de la imagen, estos quedarían separados por una distancia equivalente a la distancia real que los separa.

$$Objeto_{tamaño-real} = M * Objeto_{referencia}$$

Con la matriz de transformación calculada, sólo hay que marcar en la foto los extremos cuya distancia se desea medir. Se obtienen las coordenadas reales de dichos extremos multiplicándolos por la matriz de transformación, y solo resta calcular la distancia de punto a punto para obtener la medida real.

En este caso eliminamos la restricción que nos obliga a tomar la foto con el móvil perpendicular al plano, pero la referencia y el objeto deben seguir siendo coplanarios.

Como alternativa a tener que localizar el objeto manualmente dibujando un polígono, se estudia la posibilidad de localizar el objeto automáticamente en la escena. Para ello la librería OpenCV nos proporciona algunos mecanismos que facilitan la tarea; el detector/descriptor de características ORB (Oriented BRIEF) y un *matcher*.

El descriptor ORB nos servirá para extraer características o "puntos de interés", tanto de la imagen de referencia (objeto a encontrar en la escena), como de la imagen donde deseamos encontrar el objeto. El descriptor nos proporcionará las coordenadas de las características encontradas, así como información (una descripción) acerca de estas. La descripción de las características permitirá comparar características entre sí, y establecer relaciones entre características encontradas en imágenes distintas.

El *matcher* es un objeto que proporciona la librería OpenCV. Permitirá relacionar las características encontradas en ambas imágenes (referencia y escena) mediante sus descriptores. Un atributo de interés es la distancia, para cada *match* encontrado se almacena la distancia de Hamming de sus descriptores. Haciendo uso de la distancia podremos eliminar todas aquellas "relaciones" o *matches* cuya distancia sobrepase un umbral. El umbral escogido es tres veces la distancia mínima, de esta forma eliminaremos prácticamente todos los *matches* que sean incorrectos.

Con los *maches* incorrectos ya eliminados, solo nos falta encontrar una matriz que relacione los puntos en la referencia con los puntos en la escena. La forma de hacerlo es mediante la función findHomography. Dicha función nos proporcionará una matriz de transformación perspectiva que, al multiplicarla por cualquier píxel de la referencia, lo convierte en su correspondiente píxel en la escena.

Si creamos un rectángulo cuya diagonal vaya del punto (0,0) al (ancho, largo) de la referencia y lo multiplicamos por la matriz de homografía, obtendremos un polígono que engloba al objeto en la escena. En definitiva, tendremos el objeto localizado.

Los vértices del nuevo rectángulo obtenido reemplazan a los del polígono de inclusión que el usuario dibujaba a mano. A partir de este punto la forma de obtener medidas es idéntica.

## 5.3 Obtención de distancias y alturas.

La medición de distancias y alturas se llevan a cabo mediante la cámara y el giroscopio incorporados en el dispositivo. Para poder incluir una previsualización de la cámara en nuestra aplicación tendremos que valernos de un *SurfaceHolder* donde se mostrarán las imágenes obtenidas por la cámara.

Sobre la previsualización de la cámara añadimos una imagen que representa un puntero, la imagen se coloca centrada, de esta forma servirá para poder enfocar la base de los objetos cuya distancia queremos conocer o la parte superior para obtener la altura.

Ahora toca inicializar el sensor, para ello necesitaremos un *sensorManager,* al que solicitaremos un el sensor de orientación por defecto del dispositivo. Para capturar los datos del sensor es necesario crear una pequeña clase que implemente la interfaz *SensorEventListener,* de la cual sobrescribiremos el método *onSensorChanged(SensorEvent event),* y lo modificamos para obtener los datos del sensor. Una vez tengamos la clase y una instancia de ella simplemente hay que registrarla como oyente o *listener* del sensor, y desde el método sobrescrito podremos obtener los valores.

Para calcular una distancia, el usuario deberá colocar el móvil frente a él a la altura (H) introducida previamente y apuntar a la base del objeto, lo siguiente es obtener del giroscopio el ángulo de inclinación (α) y mediante los cálculos que se muestran a continuación obtendremos la distancia (D).

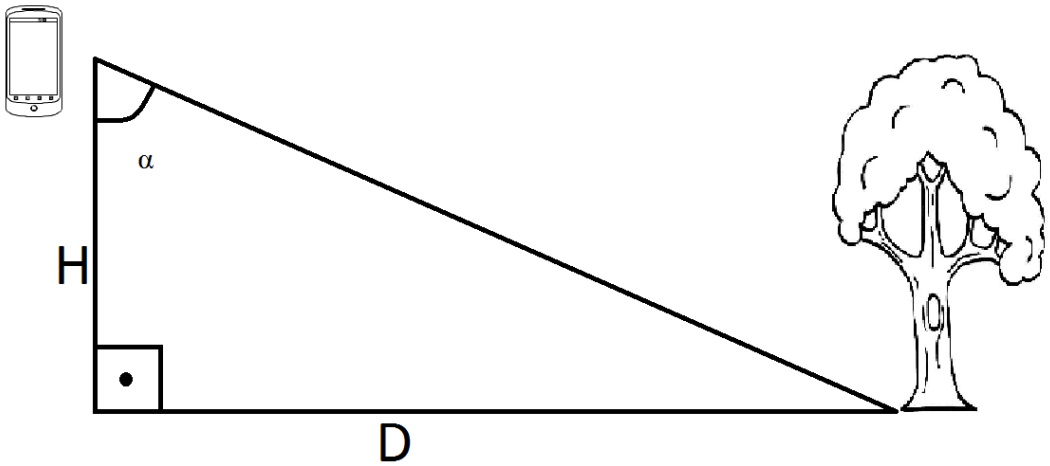



*Ilustración 13 - Medición de distancias*

$$D = \tan(\alpha) * H$$

Para calcular la altura, será un requisito necesario obtener primero la distancia, seguidamente, apuntaremos con el dispositivo a la parte superior del objeto y con el nuevo ángulo de inclinación(β) obtendremos la altura (A) del objeto. Los cálculos necesarios se detallan a continuación:

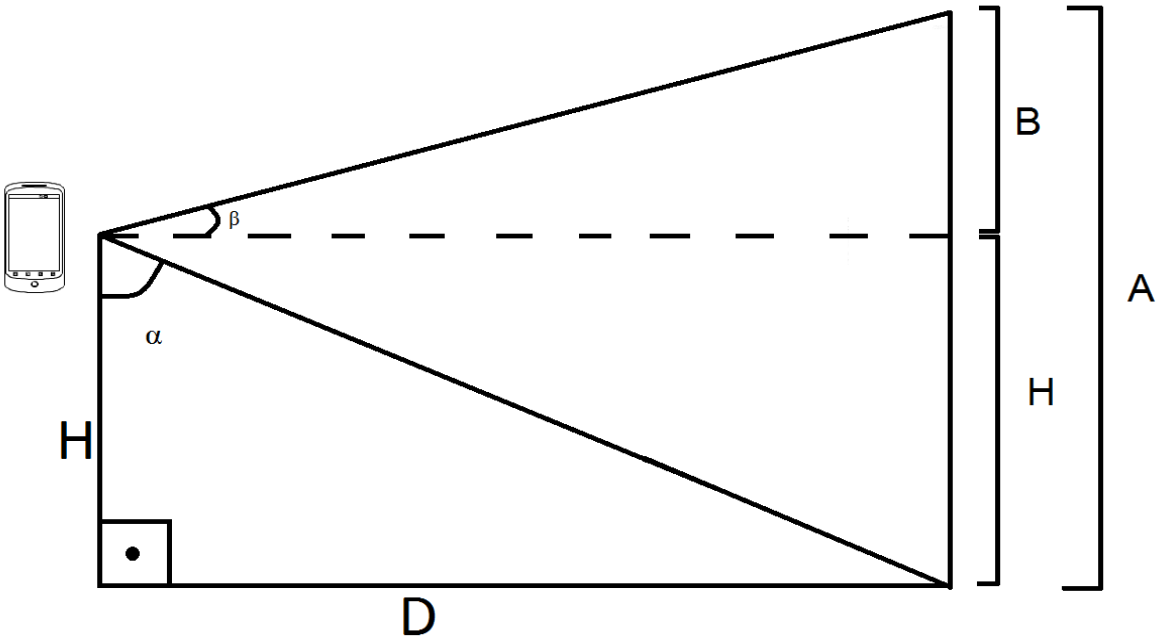



*Ilustración 14 - Medición de alturas 1*

$$B = D * \tan(\beta)$$

$$A = B + H$$

En el caso en el que el objeto sea menor que el usuario, aunque el procedimiento para obtener su altura sea el mismo, los cálculos anteriores no servirven, para obtener la altura (A) será necesario calcular dos distancias, la primera apuntando a la base del objeto (D) y la segunda apuntando a la parte superior (T), con estas dos distancias y conservando el ángulo usado para obtener la segunda (γ), se puede obtener la altura de la siguiente forma:

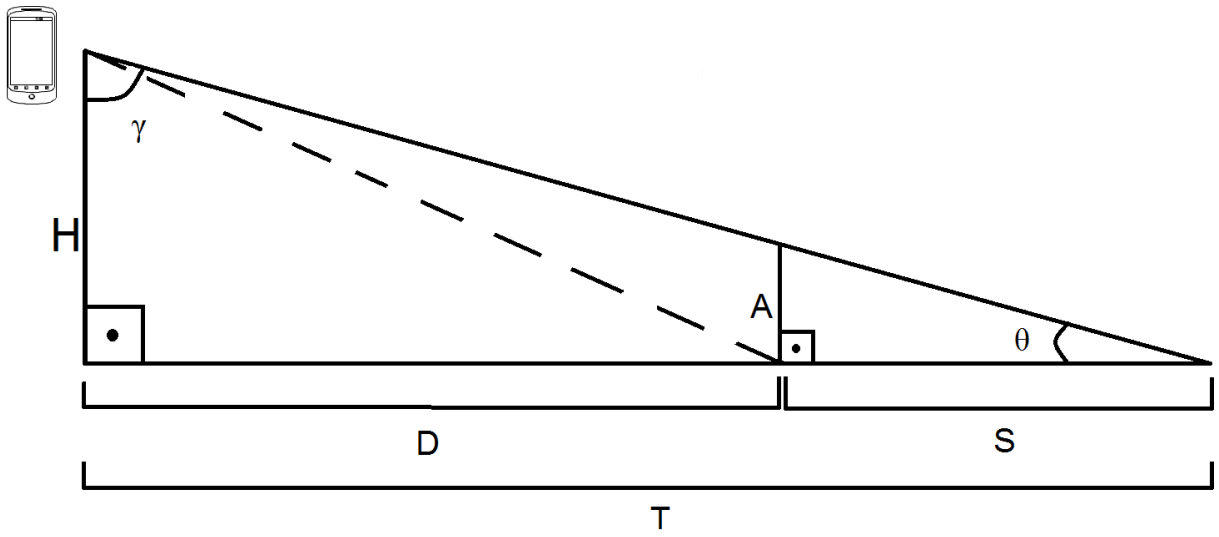



*Ilustración 15 - Medición de alturas 2*

$$S = T - D$$

$$\theta = 180 - 90 - \gamma$$

$$A = \tan(\theta) * S$$

# 6. Funcionalidad

En el presente capítulo abordaremos los temas referentes a la funcionalidad de la aplicación, trataremos de explicar detalladamente los requisitos (funcionales y no funcionales) que nuestra aplicación posee.

Es importante dejar clara desde un inicio la funcionalidad de una aplicación para evitar confusiones. El objetivo de dedicar un capítulo a este tema es establecer de forma concisa las capacidades de las que dispone nuestra aplicación, de esta forma podremos establecer los límites al funcionamiento y las prestaciones qué el usuario debería esperar.

## 6.1 Requisitos funcionales

Se entiende por requisitos funcionales del sistema aquellos que expresan las actividades que nuestro sistema desempeña, en nuestro caso describirán los servicios proporcionados por la aplicación, la forma correcta de interactuar con ella, el formato de los datos de entrada y qué resultados debemos esperar en cada caso.

### 6.1.1 Del usuario

Para tener una visión más completa de los requisitos funcionales de usuarios, podrían definirse de una forma más amigable como la reacción esperada del sistema ante las acciones del usuario. Procederemos a organizarlos en una tabla.

| ID | DESCRIPCIÓN |
|---|---|
| **1** | La aplicación permitirá al usuario hacer uso de la cámara cuando este lo requiera. |
| **2** | Se permitirá al usuario dibujar las líneas necesarias para medir con los colores escogidos en las opciones. |
| **3** | El usuario podrá configurar su altura para el correcto funcionamiento del telémetro. |
| **4** | La aplicación detectará correctamente el movimiento de los dedos del usuario al acotar un objeto mientras realiza una medición sobre la pantalla. |
| **5** | El usuario podrá introducir a mano las medidas de un objeto para usarlo de referencia en una escena. |
| **6** | La aplicación reaccionará perfectamente a los toques del usuario sobre la pantalla para salvar la distancia y la altura en la función telémetro |
| **7** | Se ofrece la posibilidad de realizar distintos tipos de medición: sobre una fotografía, sobre la pantalla del móvil, o usando la cámara como telémetro. |

### 6.1.2 Del sistema

En este punto estableceremos los requisitos funcionales a nivel de sistema, es decir, mostraremos en forma de tabla una lista con las funciones que nuestra aplicación ofrecerá al usuario.

| ID | DESCRIPCIÓN |
|---|---|
| 1 | El sistema obtiene los datos del teléfono necesarios, como la densidad de píxeles, para los cálculos. |
| 2 | Permite mediante un menú lateral acceder a todas las opciones de la aplicación. |
| 3 | Se ofrece la posibilidad de realizar distintos tipos de medición: sobre una fotografía, sobre la pantalla del móvil, o usando la cámara como telémetro. |
| 4 | La aplicación ofrece al usuario una guía práctica y sencilla para aprender su correcta utilización. |
| 5 | La detección automática de objetos ofrece resultados satisfactorios en la mayoría de los casos, no obstante, podría fallar y sería necesario intentarlo de nuevo o repetir la fotografía. |

## 6.2 Requisitos no funcionales

Los requisitos no funcionales de un sistema generalmente hacen referencia restricciones sobre el propio sistema tales como tiempo de ejecución, fiabilidad... la siguiente tabla ofrece los identificados para nuestra aplicación.

| ID | DESCRIPCIÓN |
|---|---|
| 1 | La aplicación es compatible con las versiones Android desde la 4.0 hasta las 5.0. |
| 2 | La aplicación se adapta correctamente a todos los tipos de pantalla. |
| 3 | Los datos de ajustes se gestionan correctamente. |
| 4 | Las imágenes se cargan correctamente, aplicando escalados cuando sea necesario. |
| 5 | Los textos se muestran en el idioma correspondiente según los ajustes del dispositivo (español e inglés) |

# 7. Diseño

En este capítulo nos centraremos en explicar los aspectos más relevantes del diseño de nuestra aplicación. Describiremos las herramientas utilizadas y la metodología seguida para conseguir una aplicación completamente operativa y funcional, siguiendo los patrones y las tendencias de diseño más modernas a la fecha de realización de este proyecto.

La aplicación se ha realizado siguiendo un modelo de desarrollo incremental, desde el comienzo se estimaron las principales funciones deseadas en la aplicación y se dividieron en bloques de trabajo. Los bloques se han ido implementado de forma independiente para finalmente integrarlos y formar la aplicación.

En la siguiente imagen veremos un esquema de la metodología seguida, donde cada incremento quedará reflejado en la aplicación como una funcionalidad a la que podemos acceder desde el menú principal.

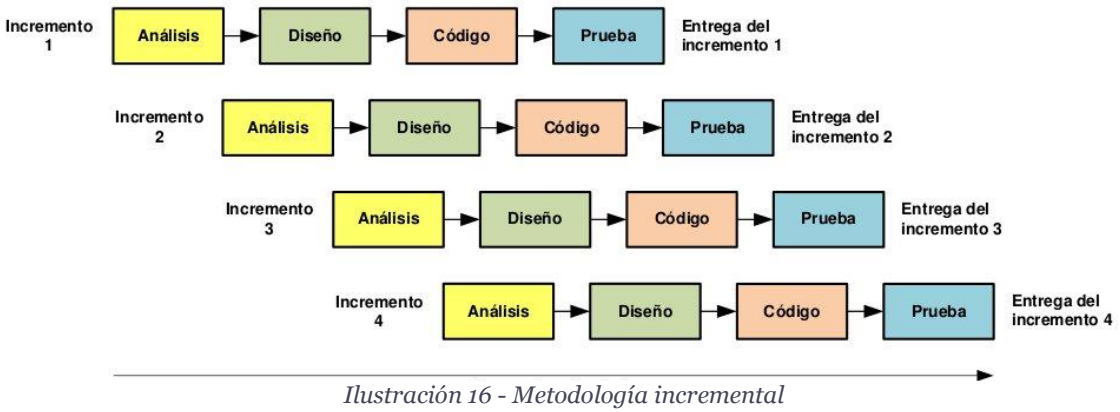



*Ilustración 16 - Metodología incremental*

## 7.1 Interfaz

Para el desarrollo de la interfaz de usuario, se ha seguido una tendencia actual bastante moderna que además de resultar muy vistosa, consigue que los usuarios se adapten fácilmente al uso de nuevas aplicaciones. Para ello se ha abandonado el uso de los típicos patrones de diseño conocidos como *dashboard* y se ha optado por apostar por el menú lateral ocultable.

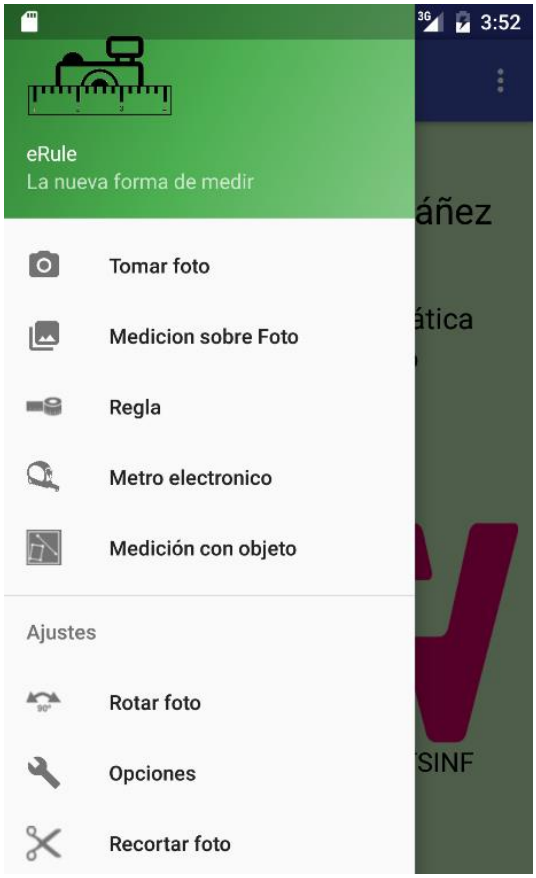



*Ilustración 17 - Menú lateral*

En la imagen previa podemos observar cómo quedaría el menú lateral en nuestra aplicación, en la cabecera de dicho menú encontramos el icono que se ha diseñado para la aplicación y el nombre que se ha escogido "eRule". Para cada entrada del menú se ha escogido un icono distinto, algunos de ellos los proporciona Android mientras que otros han sido diseñados y escalados.

El desarrollo de las distintas vistas que ofrece la aplicación se ha realizado mediante *fragments,* que pueden definirse como un conjunto de vistas o *widgets* (elementos básicos que formal la interfaz.), en sustitución a las tendencias de desarrollo más clásicas que proponen una actividad por pantalla. Esto facilita la modulización y reutilización del código (Ejemplos visuales de los *fragments* se pueden ver en el capítulo 8).

Con todo lo anterior podemos decir que nuestra aplicación cuenta con una única actividad sobre la que vamos añadiendo, eliminando o reemplazando *fragments*, otra forma de desarrollo bastante novedosa.

## 7.2 Flujo de la aplicación

Con el patrón de diseño seguido en mente, pasaremos a explicar de forma breve y concisa el flujo de ventanas de nuestra aplicación, para ello nos apoyaremos en la siguiente imagen.

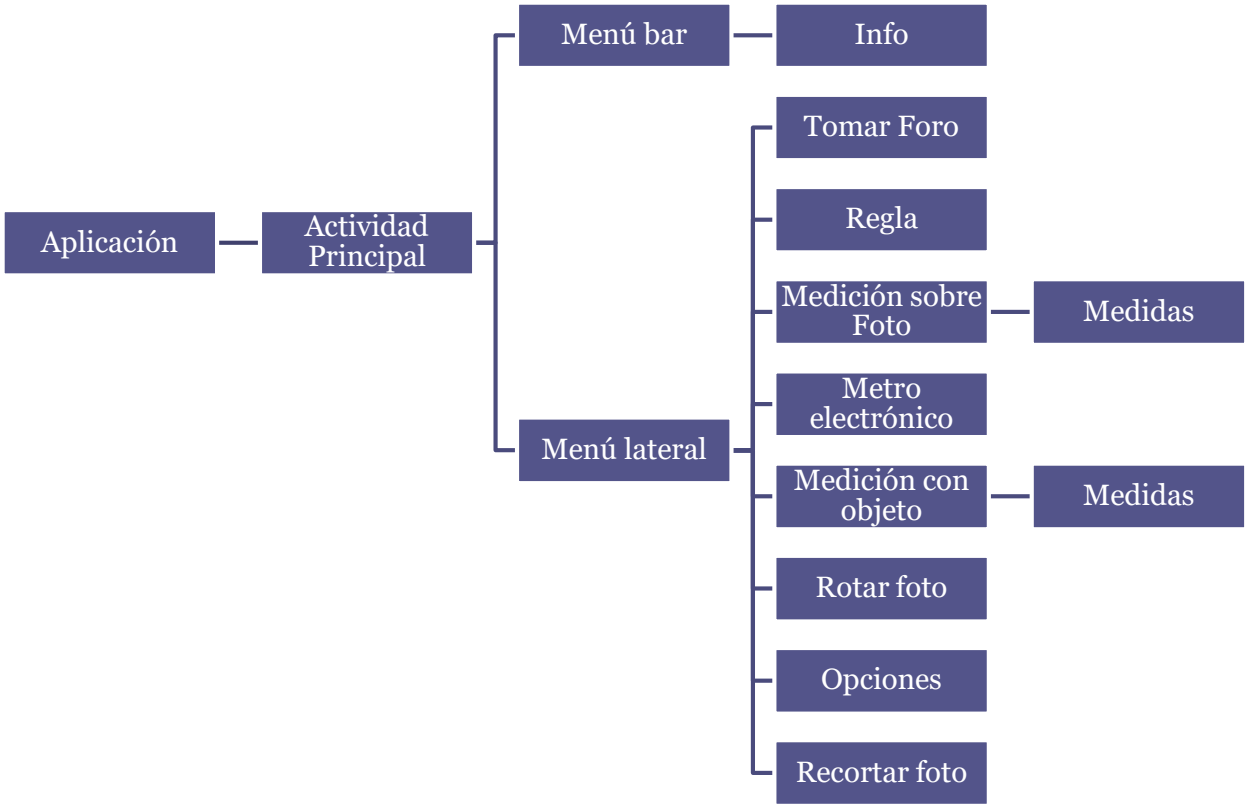



*Ilustración 18 - Flujo de ventanas*

Seguir el flujo de la aplicación es bastante sencillo, al iniciarse la aplicación se inicia de forma automática la actividad principal junto con el menú lateral y el superior, mediante los cuales podemos acceder a todas las funciones de nuestra aplicación.

Una vez dentro de una funcionalidad, si queremos cambiar a otra no tenemos más que volver a desplegar el menú lateral y escoger. Si pulsásemos la tecla retroceso o *back* saldríamos de nuestra aplicación.

### 7.2.1 Tomar fotografía

Las acciones requeridas para tomar una fotografía son muy sencillas, al seleccionar esta opción del menú lateral, se lanza un *intent* a la cámara instalada por defecto en el dispositivo, visualizamos la foto y si consideramos que sirve para realizar la medición deseada, la almacenamos. El diagrama de flujo correspondiente puede verse a continuación:

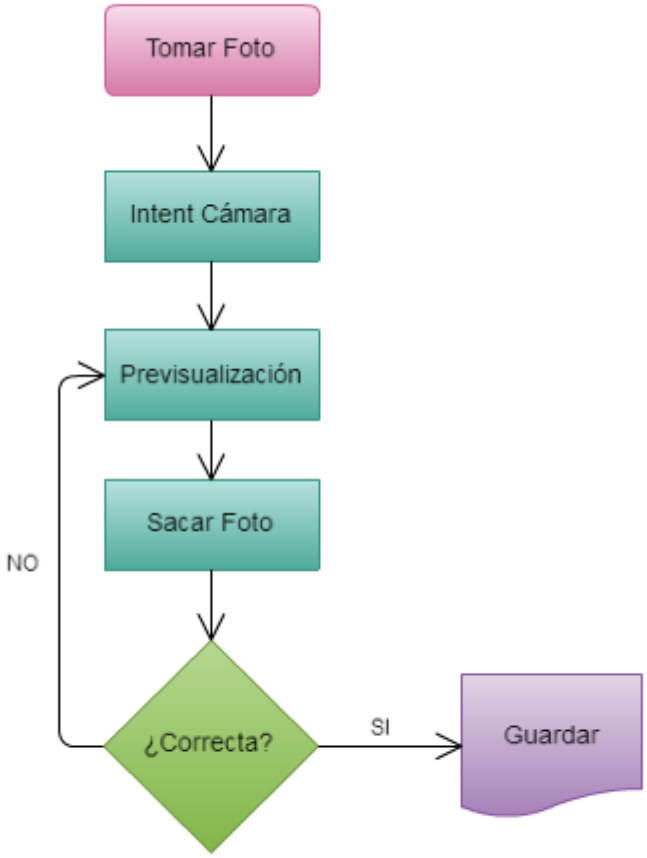



*Ilustración 19 - Flujo de toma fotografía*

### 7.2.2 Medir sobre fotografía

Para tomar una medición sobre una fotografía, la primera acción a realizar será cargar de la galería la imagen donde se encuentra el objeto a medir, marcamos dentro de la imagen una referencia con medidas conocidas, marcamos el objeto a medir y repetimos si queremos obtener varias distancias.

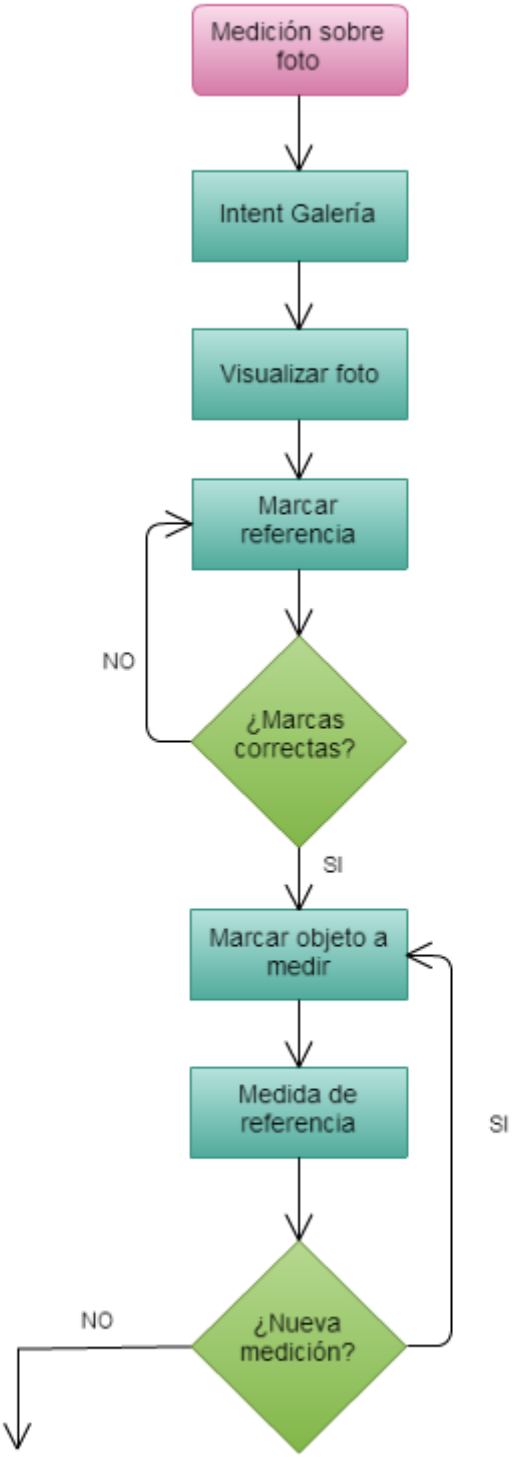



*Ilustración 20 - Flujo medir sobre fotografía*

### 7.2.3 Telémetro

Si deseamos hacer uso de la función telémetro, en primer lugar, se lanza una previsualización de la cámara, seguidamente se inician los sensores y con los valores obtenidos se calcula la distancia, el usuario puede salvar la distancia cuando desee para posteriormente calcular la altura.

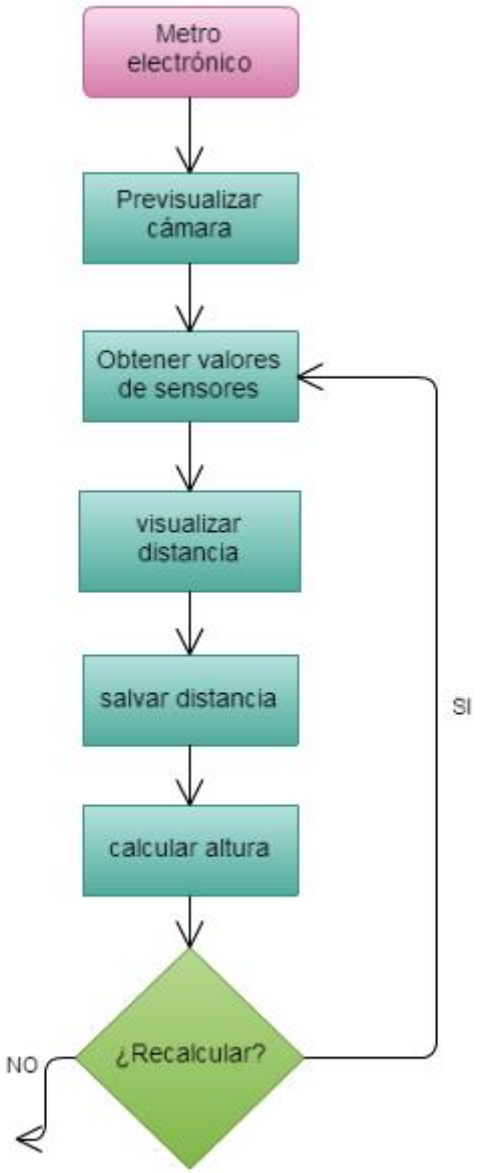



*Ilustración 21 - Flujo telémetro*

### 7.2.4 Medición con objeto conocido

Para realizar este tipo de medición inicialmente tenemos que tener las medidas de un objeto e incluirlo en la fotografía sobre la que deseemos calcular distancias, una vez que tengamos las medidas y la fotografía procedemos a cargarla, seguidamente localizamos el objeto conocido y lo usamos de referencia para tomar tantas medidas como deseemos.

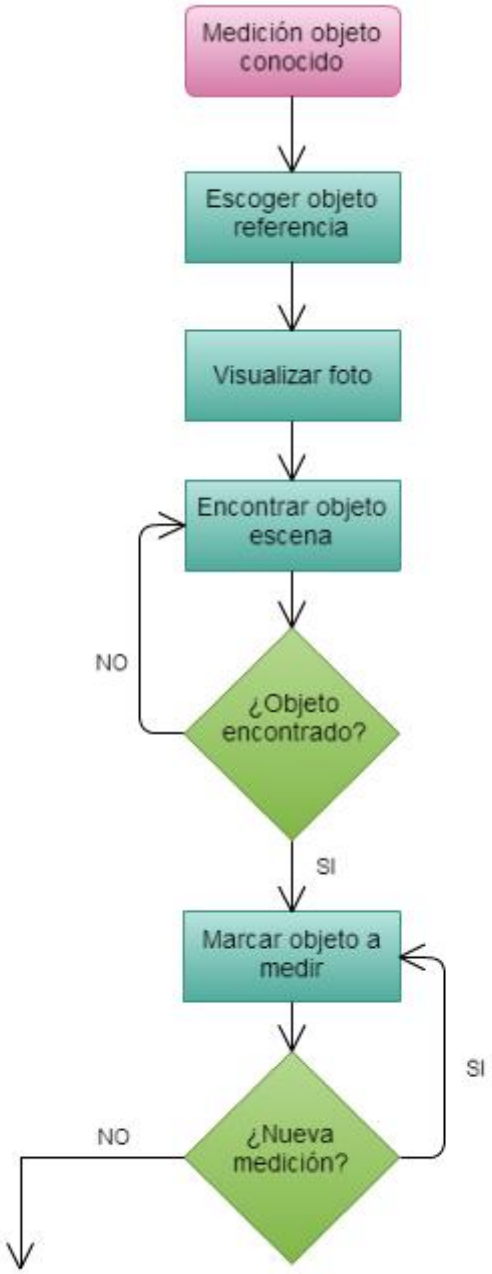



*Ilustración 22 - Flujo medición con objeto conocido*

### 7.2.5 Rotar foto

Entendemos por rotar foto simular la aplicación de rotaciones 3D sobre una imagen. Para ello cargamos la imagen de la galería, seleccionamos un valor de velocidad de ajuste y aplicamos rotaciones en los ejes vertical y horizontal, podemos repetir este proceso tantas veces como sea necesario.

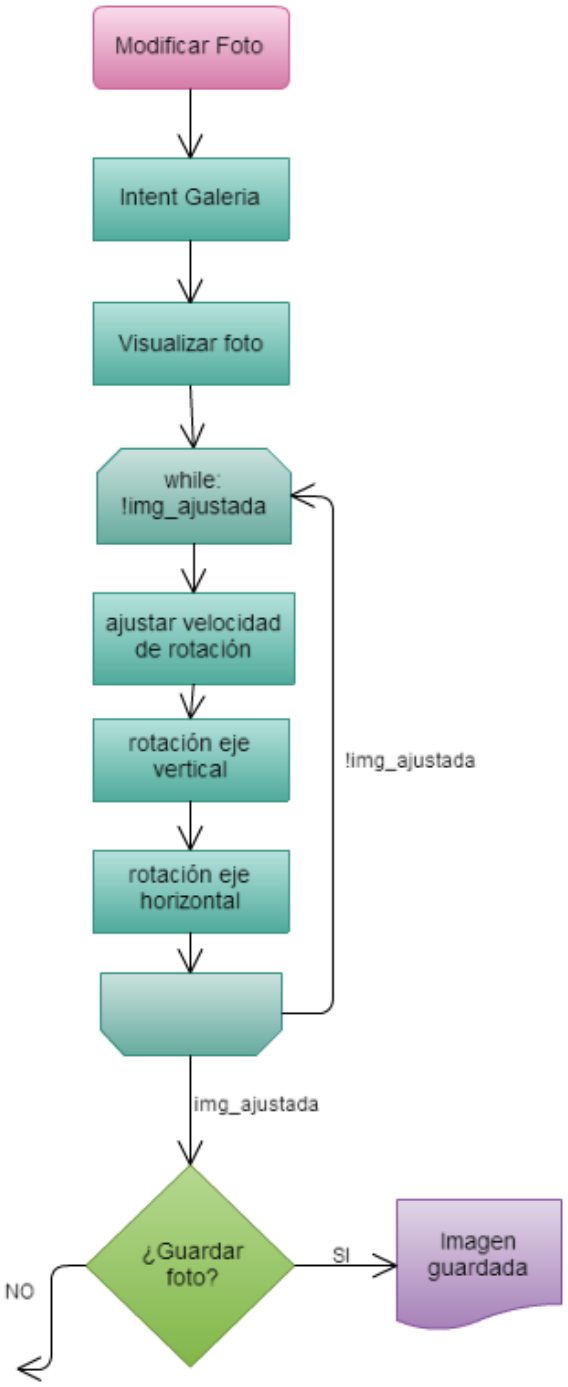



*Ilustración 23 - Flujo ajustar imagen*

### 7.2.6 Medición sobre pantalla

Para realizar una medición usando la pantalla de nuestro dispositivo como regla lo único que hay que hacer es colocar el objeto sobre la pantalla o pegado a un lateral del dispositivo y arrastrar dos dedos sobre ella para acotar la distancia que deseamos medir

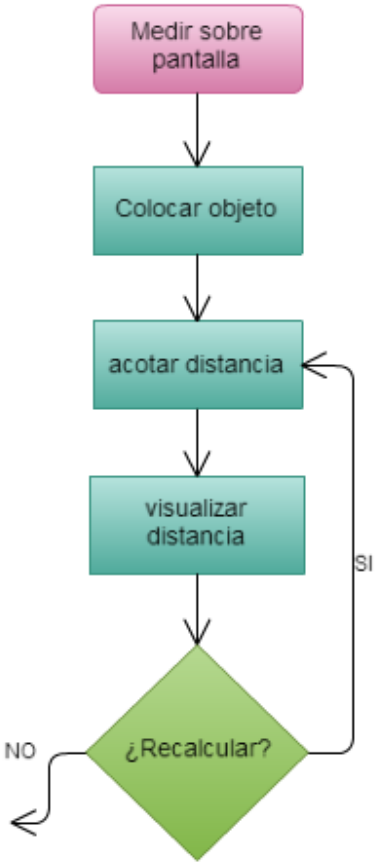



*Ilustración 24 - Flujo medición sobre pantalla*

### 7.2.7 Recortar foto

Recortar foto permite eliminar de una imagen elementos no deseados. Para lograr esta tarea se hará uso del editor de imágenes que incorpore el dispositivo, lo único que será necesario hacer es seleccionar la parte de la foto que se quiere conservar. Esta opción es útil para registrar objetos y usarlos como referencia.

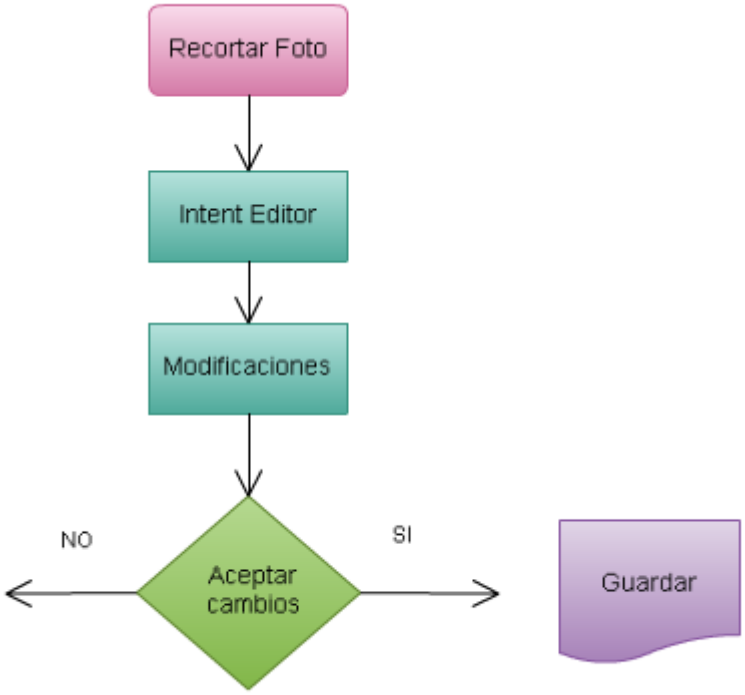



*Ilustración 25 - Flujo recortar foto*

## 7.3 Persistencia

En esta subsección se tratará el tema de acceso a los datos manejados por nuestra aplicación. La información que nuestra aplicación necesita tener almacenada es mínima, básicamente se reduce a unos datos básicos de ajuste y a las imágenes usadas para obtener mediciones.

Para la información relativa a los ajustes de la aplicación, se hace uso de *SharedPreferences,* un mecanismo ligero que permite gestionar información mediante pares clave valor, nos servirá para guardar las opciones elegidas por el usuario en el menú de ajustes.

La información gestionada por el *SharedPreferences* se almacenará en un fichero XML y será necesaria para guardar datos como la altura del usuario, necesaria para la función "telémetro" o para elegir los colores del texto y los trazos mostrados sobre imágenes a fin de poder crear contraste entre la imagen y estos.

Para el almacenamiento y la recuperación de imágenes se hace uso de la galería del dispositivo. En la mayoría de los casos las imágenes se recuperan lanzando un *intent* a la galería y recuperando la imagen mediante su dirección de almacenamiento. En raras situaciones, para acceder a las imágenes se usan las funciones proporcionadas por la librería OpenCV con el objetivo de evitar conversiones de tipos innecesarias.

# 8. Resultados

Con todo el trabajo de desarrollo finalizado y la aplicación completamente operativa, solamente resta mostrar ejemplos de su uso y comentar los resultados obtenidos. En este capítulo nos centraremos en mostrar algunos ejemplos de las diversas funciones que ofrece nuestra aplicación una vez se encuentra instalada y ejecutándose en un dispositivo.

Inicialmente, al lanzar la aplicación, lo primero que se puede ver es el menú lateral desde el que acceder a las funciones de la aplicación, en cuya cabecera podemos encontrar el icono que se ha diseñado para la aplicación.

Si seleccionamos la opción medición sobre foto, tras escoger la foto y dibujar las medidas de la referencia y el objeto a medir, se puede observar el resultado obtenido.

En azul tenemos la referencia, y en verde la medida que deseamos conocer. Para

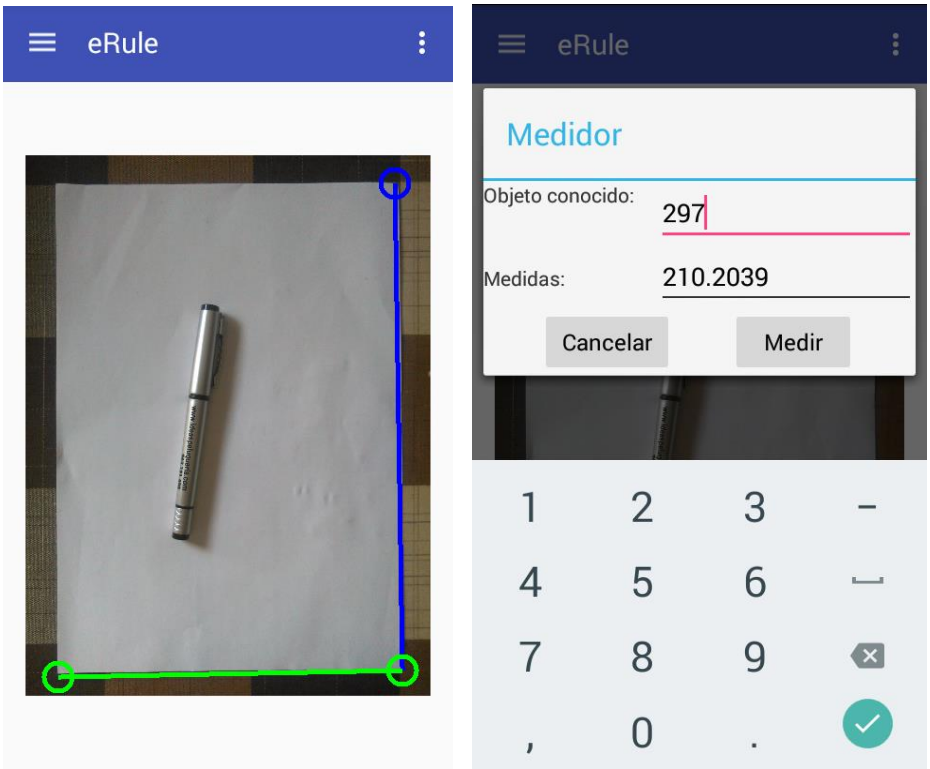



*Ilustración 26 - Prueba de medición 1*

facilitar la comprensión de los datos y la veracidad de los resultados, se ha optado por medir el ancho de un folio DIN A4 tomado como referencia su largo, ya que dichas medidas son conocidas y se consideran estándares. Introduciendo el largo del folio en milímetros, se puede observar como el ancho obtenido solo varía del estándar en aproximadamente 0.2 milímetros.

En este caso vamos a probar las funciones metro electrónico y cálculo de alturas, para ello solo debemos seleccionar esta opción en el menú lateral.

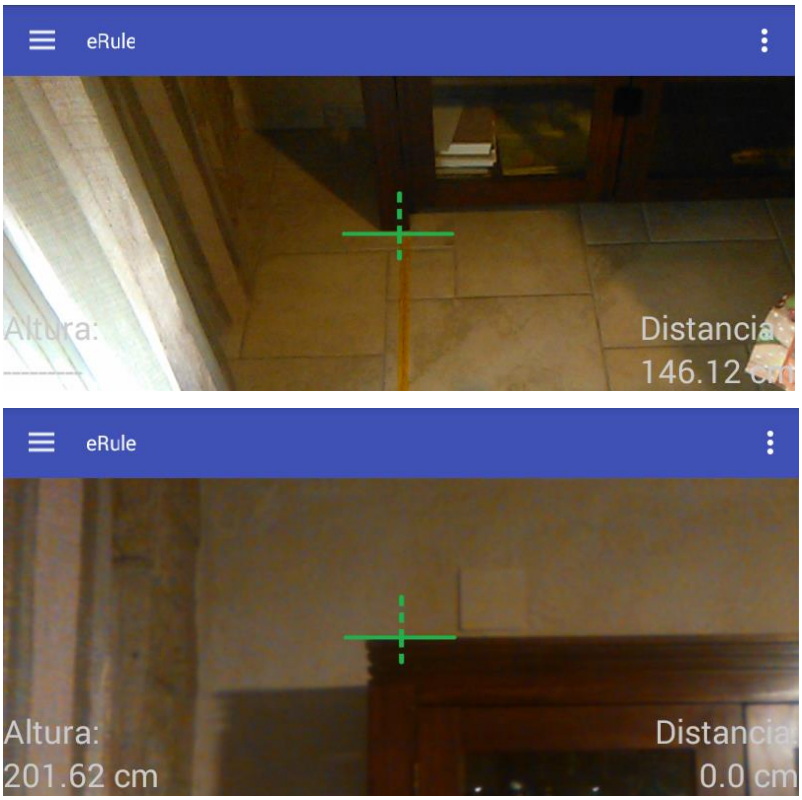



*Ilustración 27 - Medición de distancia/altura*

En la captura superior se puede apreciar la distancia a la que se encuentra un objeto y en la inferior su altura, los valores obtenidos son 146.12 y 201.61 centímetros respectivamente, para facilitar la comprobación de los datos se ha introducido en la escena una cinta métrica de 1.5 metros, de esta forma podremos valorar de forma objetiva los resultados, como puede observarse para la distancia el error obtenido es de menos de 4 centímetros y para la altura de poco más de 1 centímetro.

La siguiente prueba será el resultado de convertir la pantalla de nuestro dispositivo en un utensilio para medir, para ello solamente hay que colocar un objeto pegado a un lateral del dispositivo o sobre la pantalla y acotar la distancia a medir con los dedos como se muestra en la imagen siguiente.

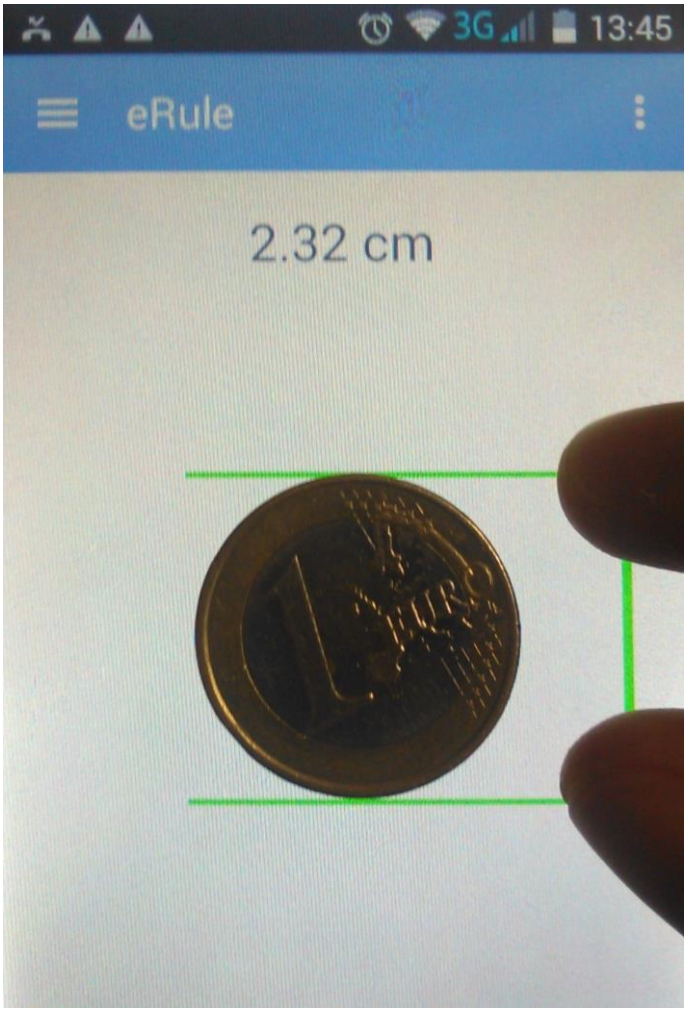



*Ilustración 28 - Regla*

Como puede observarse, de nuevo se ha usado un objeto cuyas medidas son fáciles de identificar, el diámetro de una moneda de un euro es un estándar conocido y mide 23.25 milímetros, la medición obtenida por la aplicación es de 2.32 centímetros, lo cual nos deja un error de 0.05 milímetros que es más que aceptable.

Para finalizar con las pruebas, mostraremos un ejemplo de medición usando un objeto previamente registrado (con sus medidas almacenadas). Para obtener una medición primero debemos localizar el objeto en la escena y seguidamente marcar la distancia a medir.

Se puede observar que como referencia se ha usado una tarjeta de crédito y se desea medir el ancho de un DIN A4. Como ya se ha comentado anteriormente, aunque puede parecer de poco interés medir un folio cuyas medidas se conocen, realizar esta labor nos permitirá evaluar la calidad de la medición. En este caso nuestra aplicación nos ofrece un valor de 210.1507 frente a los 210 milímetros que mide realmente, por lo que el error es aproximadamente de 0.15 milímetros.

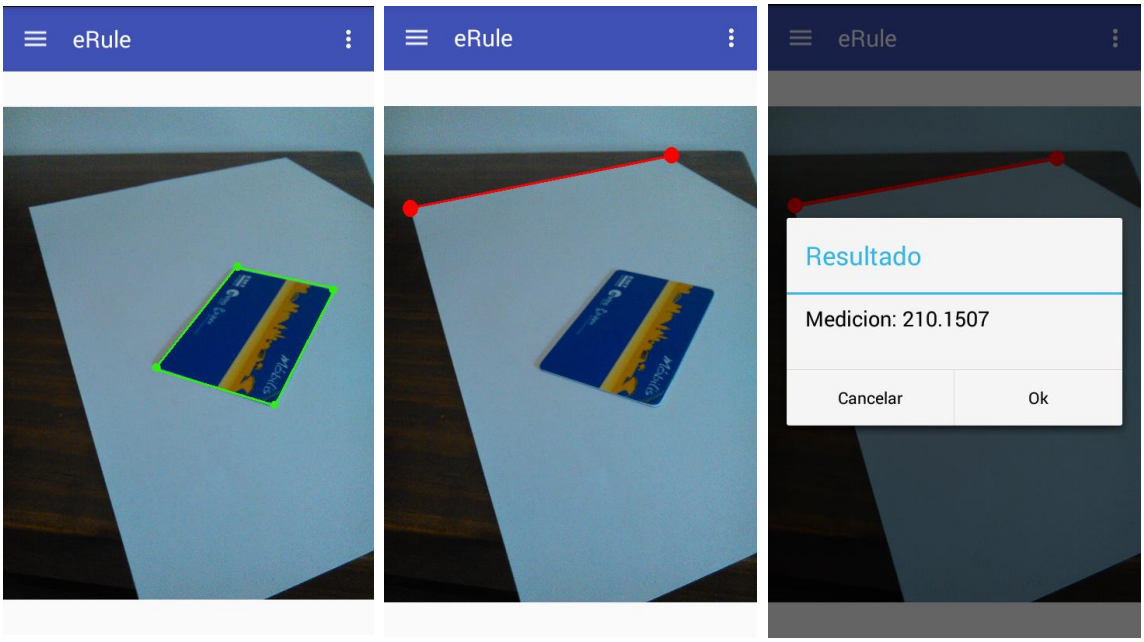



*Ilustración 29 - Prueba de medición 2*

Ahora realizaremos la misma medición, pero usando la detección automática de objetos. Podemos observar a la izquierda el objeto localizado de forma automática y a la derecha la medición obtenida (en este caso está en centímetros). Se puede observar que los resultados no son tan buenos, pero sí aceptables.

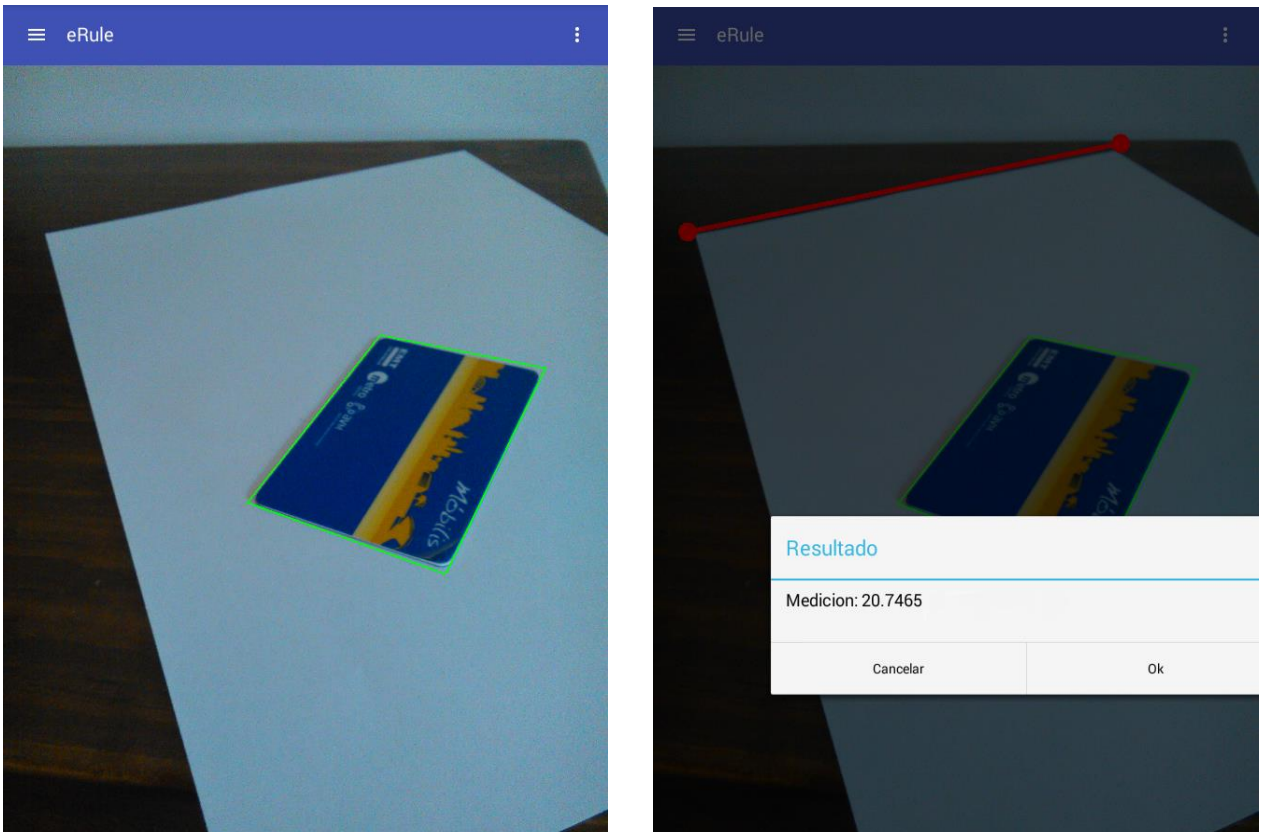



*Ilustración 30 - Prueba de medición 3*

La falta de precisión que podemos encontrar se debe a que el *matcher* no localiza los bordes de los objetos de una forma extremadamente precisa. Una alternativa al *matcher* sería emplear el clasificador en cascada que proporciona la librería OpenCV para fines similares. En este trabajo no se ha incorporado debido a que requiere muchas imágenes para entrenar y se sale un poco del ámbito de una aplicación móvil, sin embargo, los resultados obtenidos son bastante buenos.

# 9. Conclusión

Durante el desarrollo de este proyecto se ha trabajado con distintas tecnologías, el propósito de este apartado será ofrecer una opinión de cada una de ellas y una valoración general de la aplicación.

Primero de todo, decir que la elaboración de una aplicación Android para dispositivos móviles es una tarea muy laboriosa, sobre todo si partes sin ningún tipo de conocimiento acerca de Android. Realizar este proyecto ha supuesto muchas horas de estudio previo, documentación y cursos hasta poder alcanzar los conocimientos necesarios para desarrollar una aplicación completamente funcional, sin embargo, la plataforma de desarrollo para Android es muy agradecida, cualquier amante de la programación y el desarrollo de aplicaciones encontrará las horas invertidas gratamente recompensadas.

El entorno de desarrollo Android Studio es muy completo y práctico, su sistema de ayuda introduce al programador muy rápido su manejo, haciendo que crear aplicaciones sea una tarea rápida e intuitiva a pesar de la verbosidad inherente al lenguaje Java.

La API que Google nos proporciona para programar en Android, es una de las más completas con las que he trabajado a lo largo de estos cuatro años de carrera, contiene información actualizada y unas guías muy prácticas y sencillas, sin embargo, la gran cantidad de actualizaciones que sufre hace que sea difícil crear aplicaciones que sirvan en todos los dispositivos y que estén siempre actualizadas.

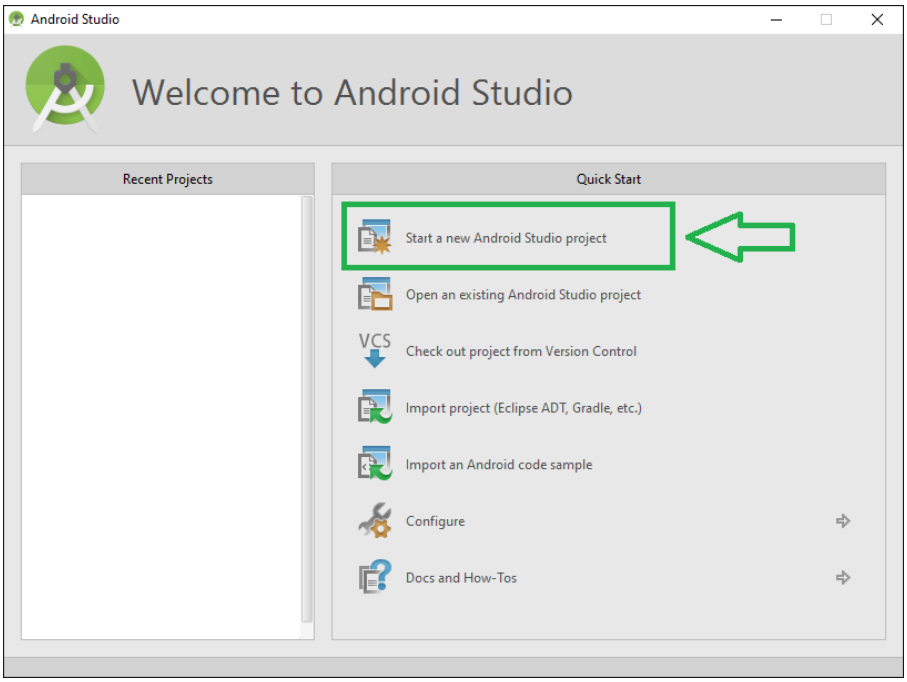
La librería de visión artificial escogida, OpenCV, es muy completa y eficiente, su impacto en la aplicación resultante no ha sido tan notorio como se esperaba en un principio, pero se ha trabajado mucho con ella, durante el desarrollo del proyecto se han ido estudiando diferentes usos para dicha librería, como podría ser el reconocimiento automático de objetos en una escena, pero la mayoría se han ido descartando por no proporcionar unos resultados aceptables, o que se estima que un usuario no aceptaría en una aplicación para móviles.

En lo referente a la aplicación, se puede afirmar sin lugar a dudas que es bastante completa y funcional, proporciona los mecanismos necesarios para realizar mediciones de una forma práctica y sencilla sin consumir muchos recursos del dispositivo, lo que la hace apta para dispositivos de cualquier gama. Los resultados obtenidos pueden considerarse satisfactorios, puesto que las medidas obtenidas se ajustan muy bien a las reales.

# A. Crear un proyecto OpenCV

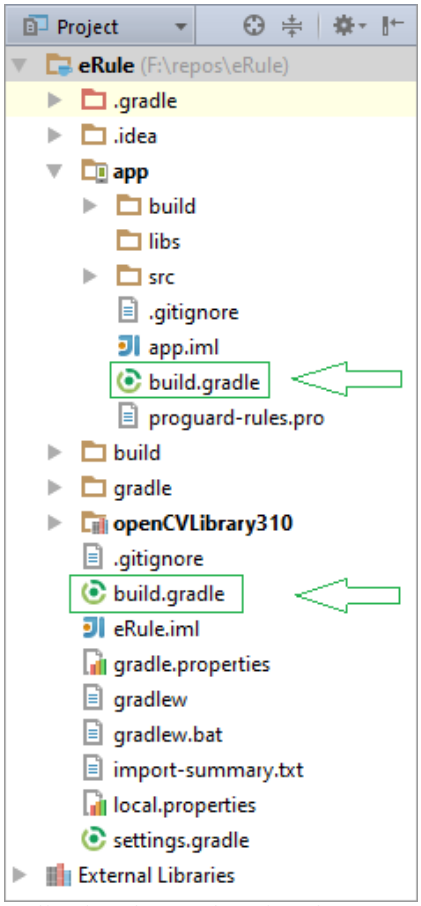
El motivo para crear un apéndice que muestre como incluir la librería OpenCV en un proyecto Android, subyace en la falta de información proporcionada por la web oficial de los propietarios de dicha librería, ya que en sus orígenes la versión para Android (OpenCV4Android) se mostró al mundo cuando Eclipse era la principal tecnología empleada para programar aplicaciones Android, a pesar de que actualmente se utiliza Android Studio como principal herramienta de desarrollo, la web oficial de OpenCV sigue mostrando muchos de sus tutoriales con ejemplos basados en Eclipse.

El primer paso que debemos seguir es crear un proyecto en Android Studio de la forma en la que lo haríamos normalmente, seleccionando los parámetros y las propiedades acorde a la finalidad de nuestra aplicación, seguidamente y si todavía no lo hemos hechos, deberemos descargar la librería de la página web oficial de OpenCV (http://opencv.org/downloads.html)en la versión con la que queramos trabajar.

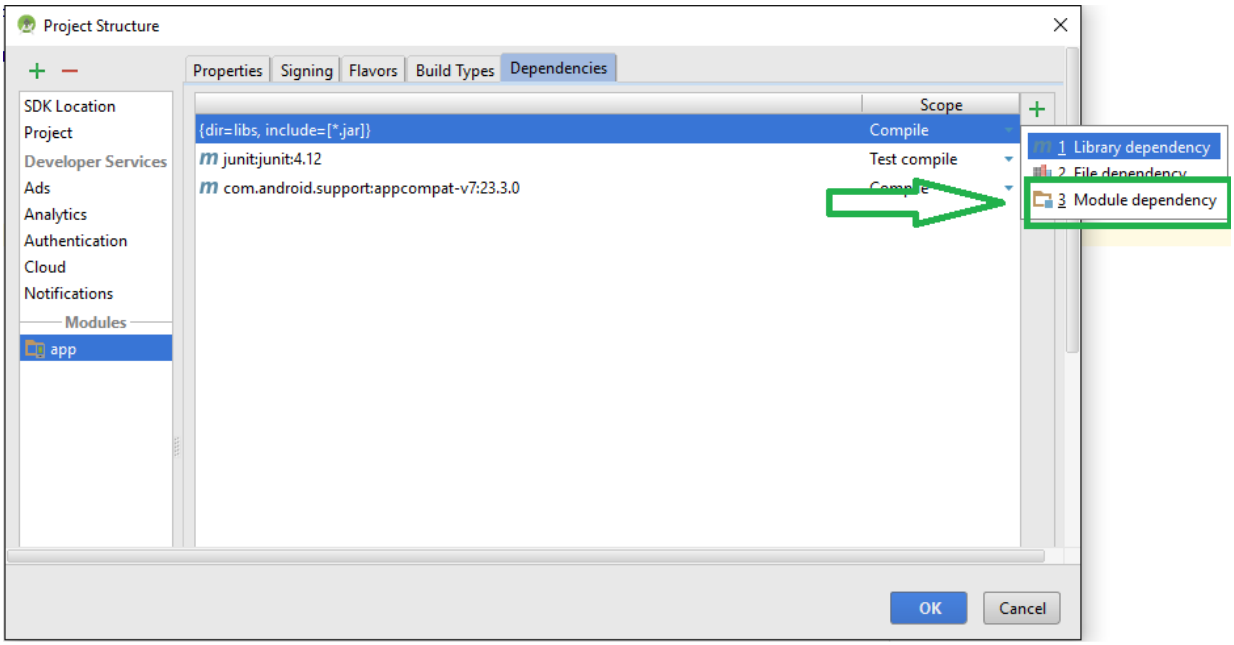



Con la librería ya descargada y descomprimida, desde nuestro proyecto seleccionamos la opción importar módulo (*File -> New -> Import Module*) y seleccionamos el directorio *sdk/java* de nuestra carpeta descomprimida y en el

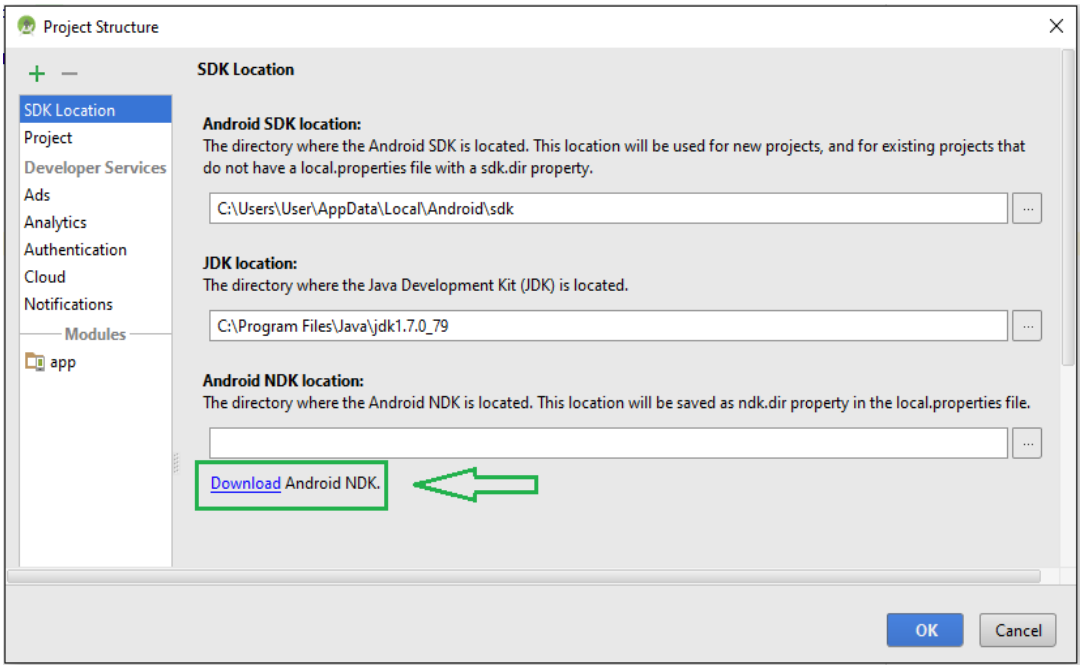
módulo recién importado modificamos el archivo *build.gradle* para que las versiones coincidan con las del proyecto.




Ahora debemos de añadir la dependencia de nuestro proyecto al módulo importado, para ello pulsamos F4, nos desplazamos a la pestaña dependencias y añadimos la dependencia que nos sugiere Android Studio, que será la dependencia a la librería.



Par poder hacer uso de la librería es necesario tener instalado el *Android NDK* debido a que la mayor parte del código está implementado en C++, si no lo tienes instalado solo tienes que hacer clic en *download Android ndk,* que encontrarás en *File -> Proyect Structure*.




Para finalizar con la instalación de la librería, en nuestro proyecto, creamos el siguiente directorio *app/src/main/jniLibs* y en el añadimos todos los ficheros con extensión ".so" que encontraremos en la carpeta *sdk/native* de nuestra librería, podemos obviar algunos archivos, como los referentes a la plataforma x86 si nuestro dispositivo no la utiliza.

OpenCV se encuentra instalado y listo para usarse, la inicialización más sencilla es una inicialización estática, simplemente debemos procurar que se ejecute el siguiente código al iniciar nuestra aplicación.

```
static {
    if(!OpenCVLoader.initDebug()){
        Log.d("OpenCV","Open cv not loaded");
    }
    else Log.d("OpenCV","Open cv working");
}
```

# B. Guía de usuario

En este apartado nos centraremos en explicar de la forma más detalla posible, lo primero que deberemos hacer será usar el administrador de archivos de nuestro dispositivo para instalar la aplicación mediante el fichero ".apk" generado. Con la aplicación ya instalada hacemos clic sobre ella para abrirla y veremos la siguiente pantalla.

### Tomar fotografía

En el menú lateral tenemos todas las opciones para medir que ofrece nuestra aplicación, las opciones de ajustes y configuración. Para comenzar clicaremos sobre la opción "Tomar foto". El funcionamiento de dicha función es bastante simple, se abrirá la cámara de nuestro dispositivo y podremos usarla de la forma en la que habitualmente lo hacemos.

### Medir sobre una fotografía

La opción "medición sobre foto" permitirá al usuario realzar medidas sobre una fotografía almacenada en el dispositivo, en primer lugar y de forma automática se abrirá la galería y el usuario deberá escoger una imagen que se mostrará de fondo, en segundo lugar, el usuario deberá tocar los extremos del objeto que servirá como referencia y posteriormente los extremos del objeto a medir. Cuando los objetos estén marcados se mostrará un diálogo para que el usuario introduzca el valor de la referencia, si pulsamos el botón "medir" obtendremos el valor del objeto y podremos pasar a marcar otro objeto para medir, si pulsamos cancelar deberemos empezar el proceso marcando nuevamente otra referencia.

### Regla

Si deseamos convertir la pantalla de nuestro dispositivo en una regla, debemos marcar la opción "regla" del menú. La pantalla se pondrá en blanco, esto será indicativo de que la aplicación está lista para medir, colocamos el objeto sobre la pantalla o pegado al lateral izquierdo del teléfono y arrastrando dos dedos sobre la pantalla acotamos los extremos del objeto que deseamos medir, para facilitar la acotación del objeto se mostrará una ayuda visual. Cuando consideremos que las cotas están bien situadas, retiramos los dedos y podremos observar el resultado en la parte superior de la pantalla.

### Metro electrónico

Para medir distancias como si nuestro dispositivo fuese un metro laser, lo primero que debemos hacer es seleccionar la altura a la que sujetamos el dispositivo en los ajustes, una vez hecho esto, la obtención de distancias es prácticamente automática, se mostrará de fondo una pre-visualización de la cámara con un puntero en el centro, que usaremos para apuntar a la base del objeto cuya distancia deseamos conocer, la distancia se irá mostrando en la parte inferior derecha del dispositivo a medida que movamos el dispositivo, para obtener la altura, debemos de obtener la

distancia del objeto y salvarla mediante un toque sobre la pantalla, posteriormente apuntaremos a la parte superior del objeto y realizaremos un nuevo toque. La altura se mostrará en la parte inferior izquierda.

### Medida con objeto

Si deseamos hacer uso de uno de nuestros objetos previamente almacenados (se pueden almacenar nuevos objetos y medidas en ajustes), para ello deberemos incluirlo en una imagen mediante la opción “tomar foto”, al seleccionar la opción del menú “medición con objeto” la galería nos preguntará qué imagen cargar, deberemos seleccionar dicha imagen. El primer paso será identificar nuestro objeto, para ello deberemos de marcar los bordes mediante toques (por ejemplo, en el caso de un cuadrado clicar sobre las cuatro esquinas), y seguidamente marcar los extremos del objeto a medir, la medida se nos mostrará sobre la pantalla.

Si tenemos activada la opción “detección automática” en los ajustes, no será necesario realizar el primer paso, simplemente marcamos los bordes del objeto a medir y la aplicación nos proporcionará las medidas.

Para mejorar los resultados de la detección automática, es conveniente que, en la foto de referencia, el objeto usado como referencia ocupe la mayor parte posible de la imagen. Para facilitar tal fin en los ajusten encontramos la opción “recortar foto” que permite ir eliminando partes de una imagen hasta obtener el objeto de interés.

### Ajustes

Dentro del submenú de ajustes tenemos múltiples opciones para variar los parámetros de nuestra aplicación, si hacemos clic en “rotar foto”, podremos cargar una imagen, fijar un valor que marcará la velocidad de giro, y rotarla sobre los ejes vertical y horizontal arrastrando un dedo sobre la imagen en sentido que queramos rotarla. La imagen se puede guardar presionando sobre el símbolo del disquete para reutilizarla posteriormente.

La opción “recortar foto” permite eliminar partes de una fotografía. Para ello se mostrará un cuadrilátero sobre una imagen que el usuario puede modificar en forma y tamaño. Al pulsar acepar todo lo que quede fuera del cuadrilátero será eliminado de la imagen.

Si seleccionamos “opciones”, encontraremos una lista de valores a modificar, el color de los trazos para mediciones, la altura para el telémetro y la más importante, la posibilidad de registrar un objeto con sus medidas.

# Bibliografía


Muhammad, Amgad (2015). *OpenCV Android programming by example.* Birmingham: Packt Publishing.

Baggio, Lelis (2015). *OpenCV 3.0 computer vision with java*. Birmingham: Packt Publishing.

Thakar, N. y Kapur, S. (2015) Mastering OpenCV Android application programming. Birmingham: Packt Publishing.

Androdeity (2011). *Arquitectura de Android* http://androideity.com/2011/07/04/arquitectura-de-android/ [Consulta 5 - 5 - 2016]

Nieto Gonzales, Alejandro. Xataca Android (2011). *¿Qué es Android?* http://www.xatakandroid.com/sistema-operativo/que-es-android [Consulta 27 - 4 - 2016]

Android developers. *Develop*. https://developer.android.com/index.html [Consulta 1 – 4 - 2016]

OpenCV. *OpenCV API Reference*. *http://docs.opencv.org/2.4/modules/refman.html* [Consulta 25 - 4 - 2016]

Stackoverflow (2015). *Android Camera preview in a fragment*. http://stackoverflow.com/questions/25604742/android-camera-preview-in-a-fragment [Consulta 12 -5 -2016]

Stackoverflow (2013). Take photo from camera in fragment http://stackoverflow.com/questions/15408240/take-photo-from-camera-in-fragment [Consulta 12 - 5 - 2016]

Software Intel (2013). *Desarrollo de aplicaciones de sensores en teléfonos y tabletas Android basados en procesador Intel® Atom™*. https://software.intel.com/es-es/android/articles/developing-sensor-applications-on-intel-atom-processor-based-android-phones-and-tablets [Consulta 15 - 4 - 2016]

Android Material (2013). *Android Architecture and Platform Initialization*. *http://madhusudhanrc.blogspot.com.es/2013/04/android-architecture-and-platform.html* [Consulta 25 – 5 – 2016]

Software de Comunicaciones. *Arquitectura Android* *https://sites.google.com/site/swcuc3m/home/android [Consulta 25 -5 -2016]*

Android Studio. *Meet Android Studio.* *https://developer.android.com/studio/intro/index.html#project_structure [Consulta 17 – 4 - 2016]*