Universidad Zaragoza
1542

Trabajo Fin de Grado

Aplicación Android para personas con problemas de visión mediante el reconocimiento de colores

Autor

Adrián Pérez Moreno

Directora

Ana María López Torres

Escuela Universitaria Politécnica de Teruel

2015

Aplicación Android para personas con problemas de visión mediante el reconocimiento de colores

RESUMEN

La aplicación desarrollada en este Trabajo Fin de Grado ha sido concebida como una herramienta enfocada en las personas con problemas de visión, sean una pérdida total o parcial de ésta o problemas únicamente relacionados con el color como el daltonismo.

Se ha creado una aplicación para dispositivos móvil, basada en el sistema operativo Android, la cual ha sido enfocada en todo momento en la usabilidad por parte de sus usuarios, por lo que se han primado dos elementos sobre el resto, la adaptabilidad y la sencillez para mejorar la experiencia de éstos.

Para aumentar la adaptabilidad a sus futuros usuarios se añadirán elementos como las respuestas mediante sintetizador de voz, los botones grandes y bien diferenciados o el manejo por voz en algunas ocasiones además se ha buscado la sencillez eliminando al máximo los elementos que puedan crear cualquier opción complicada como menús u opciones largas.

Cuando la aplicación sea iniciada, esta preguntará mediante texto si se desean los resultados mediante voz o no. Si no se ha presionado ningún botón con la respuesta en diez segundos, un mensaje de voz realizará la pregunta, activando tras él una captura de voz para obtener la respuesta. Tras este punto se abrirá la cámara y se tomará una imagen. Para finalizar tras pasar dicha imagen por un algoritmo de detección de los colores principales se muestra el nombre de los colores de la escena mediante un sencillo texto y por voz si así se ha pedido al principio, tras lo cual se da la opción de volver a tomar una imagen nueva mediante un gran botón.

ÍNDICE

ÍNDICE DE IMÁGENES

1. INTRODUCCIÓN Y OBJETIVOS

La aplicación desarrollada en este Trabajo Fin de Grado ha sido concebida como una herramienta tanto para personas con problemas de visión relacionados con el color, ya sea una pérdida total o algún otro caso menos severo como por ejemplo el daltonismo, como para usuarios que quieran entretenerse con la aplicación.

El objetivo principal es crear una aplicación que proporcione información sobre el color principal de una escena. Inicialmente preguntará a los usuarios si necesitan ayuda mediante voz para recibir los resultados finales y podrán responder mediante unos botones o mediante voz. Tras esto se abrirá la cámara del teléfono móvil y se realizara una fotografía del objeto que se desee saber el color. Una vez tomada, el algoritmo detectara los colores principales mostrando por pantalla su nombre ya sea por texto o mediante voz si así se ha indicado al inicio y finalmente se dará la opción mediante un botón de volver a capturar un nuevo objeto.

Esto puede ser muy útil para personas que deseen saber el color de algún objeto, como una prenda por ejemplo y no puedan saber con seguridad el color real, sea por el motivo que sea, puesto que además la aplicación contará con varias funciones para adaptarse a distintas personas con un grado de discapacidad visual.

Además para mejorar la usabilidad de la aplicación, durante el desarrollo se entrevistarán a posibles usuarios con problemas de visión para recoger sus opiniones respecto a la aplicación junto a posibles mejoras y fallos que existan según su opinión.

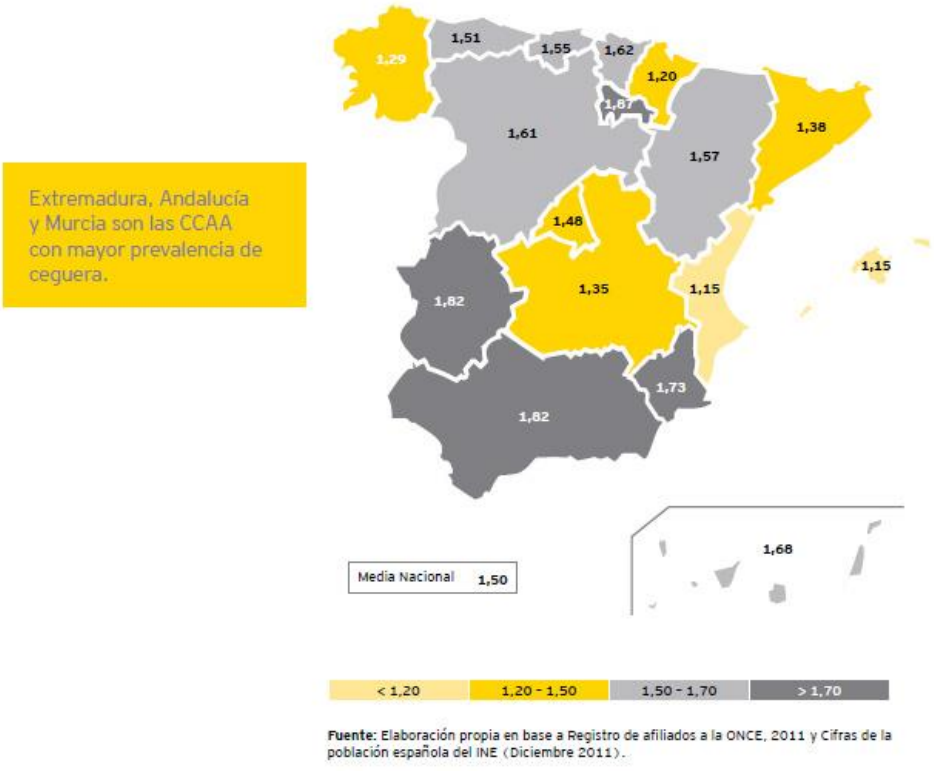
Se van a estudiar los distintos sistemas operativos posibles para escoger el más propicio teniendo en cuenta las características de los más usados además de la implantación que tiene cada uno en el mercado, siempre recordando que el objetivo de la aplicación es ayudar al máximo número posible de personas.

1.1. JUSTIFICACIÓN

Un importante factor que hay que conocer a la hora de hacer una aplicación móvil, es saber quiénes van a ser los usuarios potenciales. Dado que esta aplicación ha sido desarrollada desde un principio teniendo en mente que va a ser una herramienta practica para personas con problemas de visión, solo nos queda ampliar el estudio sobre todos los tipos de usuarios potenciales y quienes podrían ser usuarios secundarios o casuales.

Usuarios principales.

Estos son para los que ha sido diseñada la aplicación. Se componen principalmente por personas que tengan algún tipo de daltonismo, unos 2 millones de personas o que padezcan algún grado de ceguera, 979.200 personas afectadas por discapacidad visual, según datos de la Encuesta EDAD 2008 realizada por el INE. En la imagen 1.1 podemos ver una distribución de personas con ceguera según esta encuesta



(Imagen 1.1 Personas con discapacidad visual en España)

En estos casos, los usuarios lo serían por el carácter práctico de la aplicación, puesto que es una ayuda para saber con mayor certeza el color de una prenda o de cualquier tipo de objeto que estas personas no sepan o no estén totalmente seguras. Con esta aplicación se podría saber el color de una prenda antes de comprarla si saber por ejemplo si es más llamativo o apagado, el color de una prenda que ya tienen pero no saben cuál es el color exacto o simplemente si tienen curiosidad por el color de un objeto de su entorno.

Usuarios secundarios.

En este tipo de usuarios englobaremos a todos aquellos no comprendidos en el primer grupo, es decir, cualquier persona que no tenga ningún problema de visión del color.

Aquí el motivo principal para usar la aplicación es el entretenimiento o la curiosidad tecnológica de ver cómo funciona una nueva aplicación. Si bien este grupo no va a representar una gran cantidad de usuarios, también tiene que ser tenido en cuenta a la hora de pensar qué podría querer una persona que utilice la aplicación.

2. HERRAMEINTAS

Como ya se ha indicado, la aplicación se desarrolla para el sistema operativo Android.

Android es un sistema operativo inicialmente pensado para teléfonos móviles, al igual que iOS, Symbian y Blackberry OS. Lo que lo hace diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma. El sistema permite programar aplicaciones en una variación de Java llamada Dalvik. El sistema operativo proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, las llamadas, la agenda, etc.) de una forma muy sencilla en un lenguaje de programación muy conocido como es Java.

Lenguaje Java. Es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier.

En la realización de este Trabajo de Fin de Grado, se ha usado, de entre todos los entornos de programación Android, Android Studio.

Android Studio es un entorno de desarrollo integrado, IDE por sus siglas en inglés, que ha sido desarrollado para las plataformas Android. Basado en el software de JetBrains' IDEA y distribuido de forma gratuita, es una plataforma diseñada específicamente para trabajar y desarrollar aplicaciones para Android. Unas de las características más destacables es la renderización a tiempo real, construcción basada en Gradle, herramientas para detectar problemas de rendimiento, compatibilidad u otros, plantillas de diseños comunes de Android entre otras tantas.

Móvil Nexus S. Este dispositivo móvil o smartphone, cuenta con el SO Android versión 4.1.2, la cual es una versión "pura" o sin modificar del sistema operativo, lo que la hace idónea para el desarrollo de aplicaciones al no tener ninguna interferencia de la compañía constructora del teléfono.

Este va a ser usado durante todo el desarrollo de la aplicación, tanto observando la evolución del proyecto como probando cualquier cambio en el programa final.

En la imagen 2.1 se puede ver el aspecto de este modelo de smartphone.

(Imagen 2.1 Termina Nexus S)

3. DISEÑO

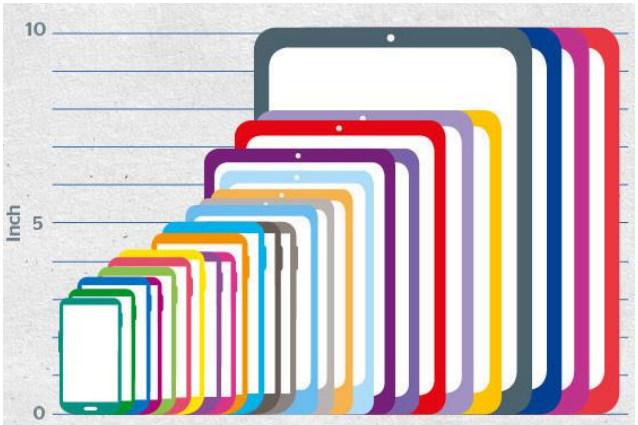
A la hora de diseñar una aplicación para dispositivos móviles, el primer paso que hay que dar es tener clara la filosofía que nuestra aplicación seguirá y cuáles son los objetivos o necesidades que queremos suplir con esta.

Una vez que tenemos más o menos clara la idea sobre la que queremos trabajar y su entorno, surgen una gran variedad de factores que hay que abordar para poder continuar con el proceso de diseño y desarrollo de la misma, dado que este tipo de plataforma se caracteriza, entre otras cosas, por la enorme diversidad tanto en hardware, como en SO o funciones de las propias aplicaciones.

3.1. TIPO DE DISPOSITIVO

Debido a la gran popularidad que cuentan los teléfonos móviles o "Smartphones" hay una gran cantidad de compañías que comercializan a su vez muy variados modelos. Esto significa que existen diferencias muy notables a tener en cuenta:

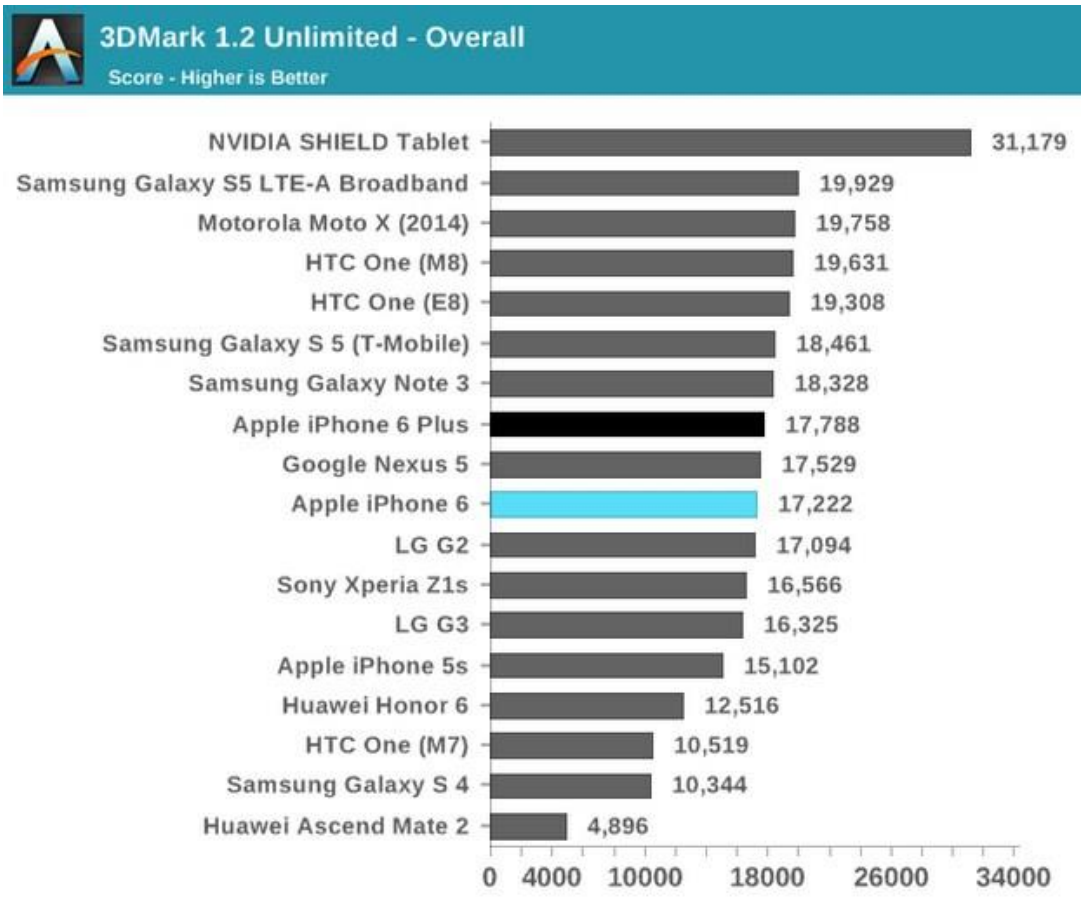
3.1.1. Tamaño de la pantalla. En consecuencia a la gran variedad de modelos existentes en el mercado, el tamaño de pantallas será muy variable y en todas, la aplicación ha de funcionar correctamente, por lo tanto a la hora de diseñar y desarrollar el interfaz, se ha de prestar atención a que un cambio de tamaño no distorsione o entorpezca el uso por parte de los usuarios. En la imagen 3.1 podemos ver unos ejemplos de lo variadas que pueden ser las pantallas y así hacernos una idea de que este punto no puede ser un cabo suelto que arruine la experiencia a una parte de los usuarios.



Galaxy Phone	Size	Galaxy Phone	Size	Galaxy Tablet	Size
Y	3 inch	S4	5 inch	Note 8.0	8 inch
Young	3.27 inch	S4 Active	5 inch	Tab 3	8 inch
Ace	3.5 inch	S4 Nexus	5 inch	Tab	10.1 inch
Fame	3.5 inch	Note	5.3 inch	Tab 2	10.1 inch
Ace 2	3.8 inch	Note 2	5.5 inch	Tab 3	10.1 inch
Ace 3	4 inch	Mega	5.8 inch	Note 10.1	10.1 inch
S3 Mini	4.3 inch	Mega	6.3 inch		
S4 Mini	4.3 inch	Tab 2	7 inch		
S4 Zoom	4.3 inch	Tab 3	7 inch		
S3	4.8 inch	Tab	7.7 inch		

(Imagen 3.1 Tamaño de móviles y tablets)

3.1.2. Potencia del dispositivo. Este es un punto muy importante a tener en cuenta y derivado nuevamente de la gran variedad de modelos. Muchos de estos tienen grandes capacidades de procesamiento, mientras que otros cuentan con unas capacidades más limitadas tal y como vemos en la imagen 3.2, que nos muestra una clasificación en cuanto a rendimiento de varios modelos existentes en el mercado. Por lo tanto, será conveniente que la aplicación no necesite mucha potencia para poder funcionar, y así asegurar que podrá funcionar en el mayor número posible de terminales.



(Imagen 3.2 Gráfica potencia de smartphones)

3.1.3. Otros. Para finalizar este apartado y siguiendo en la misma línea, habrá que tener en cuenta que no todos los dispositivos móviles cuentan con los mismos componentes, y en nuestro objetivo por intentar que el máximo de usuarios sean capaces de usar la aplicación se superará este problema no usando ninguna parte que no sea común a todos los smartphones. En la imagen 3.3 se pueden observar la mayoría de partes comunes a todos los dispositivos.

(Imagen 3.3 Partes comunes de un móvil)

3.2. SISTEMA OPERATIVO

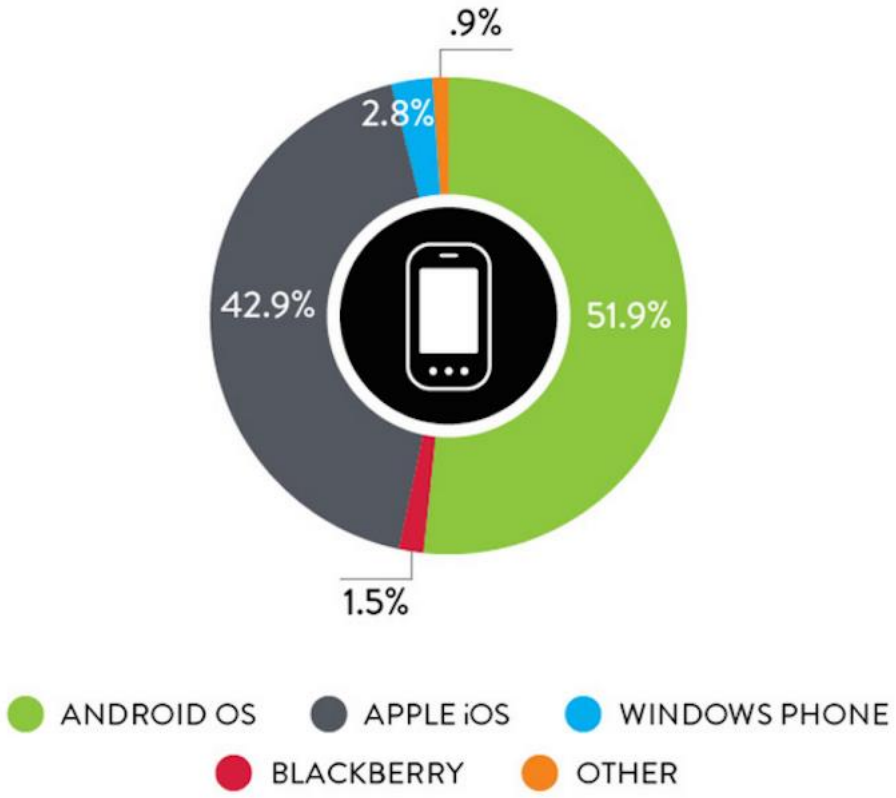
Este es un apartado muy importante a tener en cuenta, debido a que el desarrollo de aplicaciones para distintos SO es totalmente distinto y por lo tanto incompatibles entre plataformas sin un trabajo de adaptación prácticamente desde la base de la propia aplicación. Los principales sistemas que existen hoy en el mercado son: Android, iOS, Windows Phone, BlackBerry OS y Symbian.

Una vez hemos visto que habrá que escoger solo una plataforma en la que desarrollar la aplicación, se va a estudiar dos facetas principales, características de los entornos de desarrollo y sistemas más extendidos.

3.2.1. Entornos de desarrollo. Dado que cada sistema operativo es creado por una compañía distinta y con distintos objetivos en mente, las herramientas que éstas ofrecen a los desarrolladores son totalmente distintas. Así pues vamos a ver las principales características que nos ofrece cada uno:

- iOS: es el sistema operativo utilizado en los dispositivos de Apple. Apple lo presenta como el sistema operativo móvil más avanzado en el mundo. Tiene las desventajas de tener funcionalidades compatibles únicamente entre iPhones.
- Android: es un sistema operativo de Google para smartphones, PDA y terminales móviles. Sus principales ventajas son el coste de los smartphones, una gama de aplicaciones es muy extensa y la compatibilidad entre modelos. Entre las desventajas tenemos el mayor consumo de la batería.
- Windows Phone: Este es el sistema operativo que lanzó Microsoft. Las ventajas son ejecución rápida y compatible con Office. Las principales desventajas son pocas aplicaciones disponibles, ausencia de multitarea y ausencia de la tecnología Flash.
- BlackBerry OS: es un sistema operativo cuya principal característica es permitir el uso de varias funciones al mismo tiempo en el ámbito profesional. Entre sus ventajas encontramos una navegación web rápida y ser muy práctico para leer los emails. Por el otro lado su principal desventaja es la ergonomía.
- Symbian: comprado por Nokia, Symbian ha sido utilizado por la marca Nokia, Samsung, y Sony Ericsson. Las ventajas son un sistema operativo fiable (presencia desde hace 10 años en el mercado), mayor duración de la batería y un sistema multitarea bien desarrollado. Sin embargo en las desventajas nos encontramos con un mercado de aplicaciones muy pobre y la interfaz poco estética y poco intuitiva.

3.2.2. Sistemas más extendidos. Este apartado puede llegar a tener incluso más importancia que el anterior, puesto que si nuestra aplicación está disponible en el sistema más común del mercado, aumenta la facilidad con la que los usuarios la podrán usar y encontrar. En la imagen 3.4 se puede ver la distribución del mercado de SO durante el año 2014:



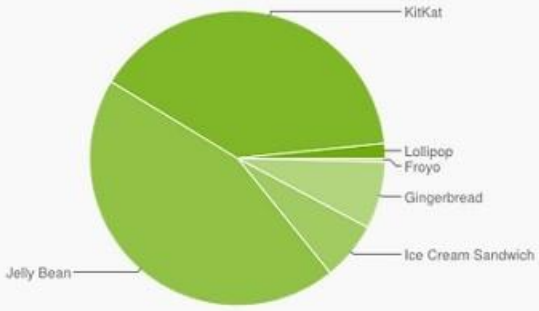
(Imagen 3.4 distribución de SO móviles)

Gracias a este gráfico podemos saber que Android y iOS dominan la gran parte de este mercado, por lo que hacer una aplicación para cualquier otro sistema operativo limita en gran medida que usuarios podrán usarla.

En este momento surgiría el conflicto sobre si es mejor iniciar el desarrollo, pensando en la implantación del SO, para Android o iOS, pero si nos fijamos en el mercado únicamente español (imagen 3.5), donde sería lanzada inicialmente la aplicación podemos ver que a diferencia del marcado a nivel mundial, aquí solo predomina uno de los sistemas, Android. Por lo tanto parece posible que la plataforma elegida sea ésta, pero entonces hay que prestar atención a una nueva variable, las versiones de Android, puesto que pueden llegar a existir incompatibilidades entre estas, así que ahora hay que observar la imagen 3.6 que muestra una tabla con su grafico sobre el reparto de usuarios en las distintas versiones del sistema operativo.

(Imagen 3.5 Distribución SO en España)

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	7.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	6.4%
4.1.x	Jelly Bean	16	18.4%
4.2.x		17	19.8%
4.3		18	6.3%
4.4	KitKat	19	39.7%
5.0	Lollipop	21	1.6%



Data collected during a 7-day period ending on February 2, 2015.
Any versions with less than 0.1% distribution are not shown.

(Imagen 3.6 Reparto de usuarios en versiones Android)

3.2.3. **Conclusiones**.

Dado que el objetivo principal de la aplicación es maximizar la cantidad de usuarios que la puedan usar y no el uso de alguna característica concreta de un sistema operativo, se ha tenido en cuenta el sistema más extendido, aunque también teniendo a la vista las características de cada uno de ellos. Por esto, tanto si nos fijamos en la distribución mundial, con un 51.9%, o la española, con un 85.9%, vemos que Andorid es la mejor opción en cuanto a cantidad de usuarios se refiere, por lo tanto será la elegida.

Prosiguiendo con esta filosofía de aumentar al máximo la cantidad de usuarios que puedan tener la aplicación, habrá que escoger una versión base de Android, a partir de la cual todos los móviles con esa versión o alguna superior serán capaces de instalar y hacer funcionar correctamente la aplicación. Usando como base el gráfico anterior (imagen 3.6), se ha situado como versión mínima de desarrollo la API 16 de Andorid, en la que confluyen suficientes herramientas de desarrollo y la mayoría de usuarios, con un 85,8% del total

3.3. APLICACIONES SIMILARES.

Se ha realizado un análisis de las operaciones disponibles en Google Play con una funcionalidad similar a la que es objetivo de este proyecto. A continuación se presentan alguna de ellas y se comparan con la aplicación desarrollada en este trabajo.

3.3.1. Resumen de funciones

Si analizamos las aplicaciones disponibles para Android que tienen un funcionamiento parecido a la realizada, lo primero que podemos observar es que la gran mayoría de éstas están enfocadas hacia un uso más lúdico o especializado para el trabajo con los colores.

La estructura que siguen suele coincidir: primero abrir la cámara directamente tras el inicio de la aplicación y mientras se está ejecutando analizan el color del centro de la pantalla mostrándolo en una esquina de ésta junto a algunos datos más técnicos que no todos los usuarios sabrán cómo usar, por ejemplo el código del color en HTLM o RGB. Complementariamente muestran el nombre del color captado, prácticamente siempre en inglés, que en bastantes de los casos analizados, pueden llegar a ser demasiado complejos o específicos, como por ejemplo color “awesome” (impresionante), “bazaar” (bazar) o “brass” (latón) que serán difícilmente identificables con un color en concreto por la mayoría de usuarios.

3.3.2. Ejemplos de aplicaciones parecidas

Algunas de las aplicaciones más parecidas, sus características destacadas son:

Color ID (Free)

En esta aplicación veremos la imagen que está tomando la cámara mientras al mismo tiempo muestra en esa misma pantalla una muestra del color, el nombre de éste en inglés y su código en hexadecimal. Como opción añadida hay un botón para congelar la pantalla y poder leer los resultados. (Imagen 3.7)

(Imagen 3.7 Pantalla de la aplicación Color ID (Free))

Color Detector

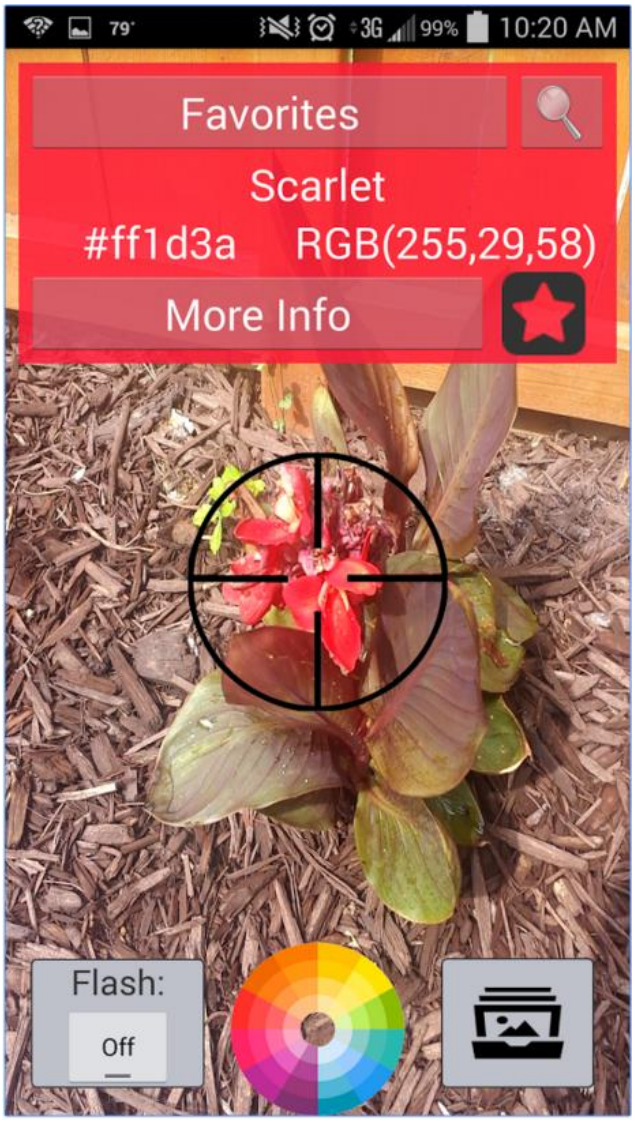
En este caso se obtienen prácticamente los mismos resultados que en el caso anterior, ya que de igual modo vemos la cámara a pantalla completa (Imagen 3.8) junto a una muestra del color analizado y los datos obtenidos tras analizar el color, nuevamente el nombre del color específico en inglés pero esta vez se añade información adicional sobre el color.

(Imagen 3.8 Pantalla de la aplicación Color Detector)

Color Capture & Identifier

Como podemos ver, nuevamente de reproduce la misma estructura de la aplicación, mostrando la cámara con el nombre del color y los datos de éste, aunque en este caso se añade la opción de obtener en una pantalla secundaria mucha más información técnica (Imagen 3.9).

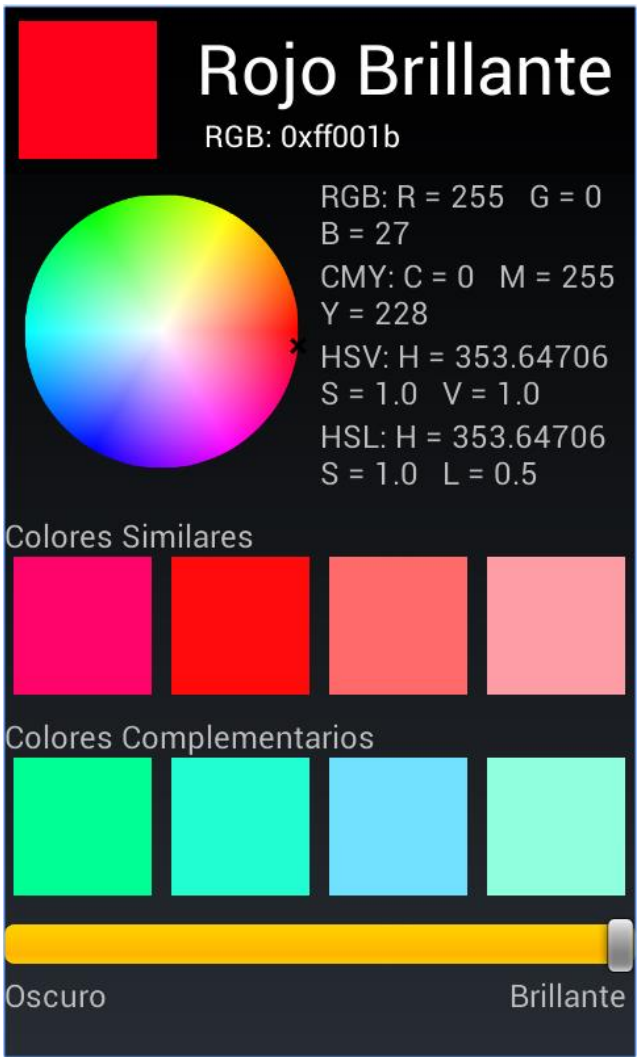
El uso no varía respecto a las anteriores, puesto que los datos numéricos del color solo los usarán unos pocos usuarios interesados en la parte más técnica del tema.



(Imagen 3.9 Pantalla de la aplicación Color Capture & Identifier)

Color Scanner

En este caso hay dos diferencias principales respecto al resto de casos analizados y es que la cámara solo está funcionando mientras se capta la imagen, por lo que la información se muestra en una pantalla nueva (imagen 3.10) y que esta aplicación tiene la opción de mostrar los resultados en castellano. Junto a estas, también hay pequeñas diferencias como la muestra de colores similares o los complementarios a estos, pero tanto el uso como el resto de información es muy similar al de las demás aplicaciones revisadas.



(Imagen 3.10 Pantalla de la aplicación Color Scanner)

(Todas las imágenes de este apartado han sido obtenidas de sus respectivas páginas en play.google.com)

3.3.3. Principales diferencias con la aplicación desarrollada en este proyecto.

Una vez vistos los ejemplos disponibles más próximos a la aplicación, podemos comparar las principales diferencias y estudiar si estos son ventajas o no en determinados casos.

Objetivo y uso: Esta es la mayor diferencia, dado que en todas las aplicaciones el objetivo principal es el entretenimiento y algunas de ellas tienen el objetivo secundario de ser útiles a las personas daltónicas, mientras que en mi aplicación el usuario con problemas de visión ha sido el centro y fin del desarrollo, lo que afecta directamente a la manera de ser usada, ya que por ejemplo, se ha añadido el control por voz para ayudar al usuario.

Algoritmo de detección de color: Tras unas pruebas con estas aplicaciones, se puede comprobar que su algoritmo está diseñado para la detección y análisis de un solo color, por otro lado yo he tenido en cuenta la posibilidad de que el objeto que está delante de la cámara tenga más colores. Esto es una ventaja en sí misma, pero como se puede analizar más de un color, se genera una carga computacional algo mayor en el dispositivo, lo que podría llegar a ser una desventaja.

Muestra de los resultados: Este apartado está íntimamente relacionado con los objetivos de la aplicación, porque mientras las apps analizadas intentan maximizar la cantidad de datos mostrados, yo he simplificado al máximo este proceso, para facilitar la experiencia. Nuevamente esto podría ser una desventaja o no dependiendo del tipo de usuario y cuales sean sus objetivos, pero esperando que sea usada principalmente por el público objetivo, esto es una ventaja.

3.4. USABILIDAD.

Diseño centrado en el usuario

La usabilidad en las aplicaciones móviles es uno de los elementos fundamentales para el éxito de las mismas

Lo primero que hay que tener en cuenta, en términos de usabilidad, es que una aplicación debe ser efectiva y eficiente. Esto significa que debe ofrecer al usuario lo que busca de una manera rápida y sencilla. Una aplicación debe satisfacer necesidades en el menor tiempo posible y sin necesidad de que el usuario tenga grandes conocimientos de su uso o requiera un largo proceso de aprendizaje.

Las aplicaciones deben llegar a la efectividad y la eficiencia mediante una buena experiencia de usuario. Sólo de esta manera se generarán en los usuarios una serie de emociones que le aporten confianza para repetir. Por tanto, la usabilidad es un mecanismo fundamental a la hora de fidelizar a los usuarios en el ámbito de las aplicaciones móviles.

Cuanto mayor sea el grado de usabilidad, más probabilidades habrán de agradar y satisfacer a los usuarios que se la hayan descargado.

Para conocer el contexto de la aplicación es muy importante conocer al público objetivo al que va dirigida, conociendo su comportamiento en este entorno, por eso este punto ha sido central durante todo el desarrollo de la aplicación.

Dado que la aplicación tiene un público muy concreto, es relativamente fácil conocer de qué manera hay que seguir con el desarrollo para aumentar su usabilidad. Así, entre las aplicaciones adaptadas a las personas con problemas de visión, las recomendaciones principales que se van a seguir son:

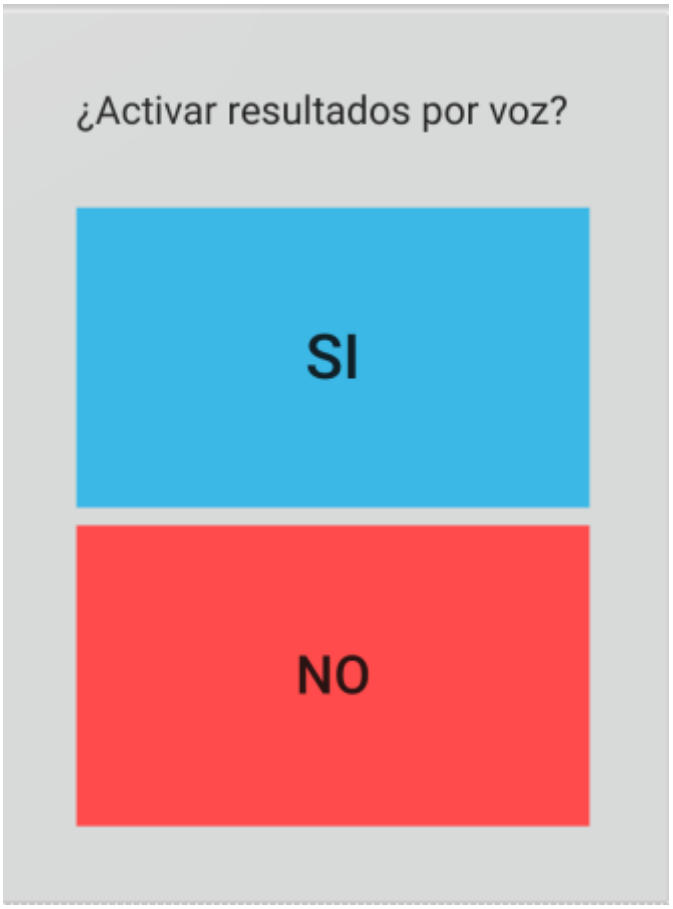
1. Identificación de objetos de la interfaz de usuario de forma genérica, todos los mensajes, sistemas de ayuda y textos que aparezcan en la aplicación para explicar su funcionamiento o interaccionar con el usuario, se deben poder entender sin dificultades con un lenguaje claro y sencillo.
2. Nombre de los elementos de la interfaz. Debe garantizarse que todos los elementos de la interfaz, como casillas de verificación, botones o texto estático, están perfectamente identificados y son únicos en su contexto
3. Nombres cortos y concisos. Deben utilizarse nombres que sean cortos y que no incluyan su función, de forma que el texto se diferencie del rol, el estado y el valor del elemento, información no visible pero que los lectores de pantalla verbalizarán a petición del usuario. Un etiquetado incorrecto podría producir una lectura poco natural, como por ejemplo “Botón, botón de reproducción”.

4. Elección del método de entrada. Se debe permitir al usuario elegir el dispositivo de entrada preferido, ya sea el teclado, trackpad, pantalla táctil o la conexión de productos de apoyo que los sustituya, de forma que pueda manejarse totalmente la aplicación con cualquiera de los métodos.
5. Elección del método de salida. Se debe proporcionar al usuario la posibilidad de elegir sistemas redundantes y combinados de salida para el sonido, imágenes, texto y gráficos. Sin embargo hay aplicaciones (productos de apoyo) que precisamente tienen como principal función dar una alternativa de salida adaptada a los usuarios con diversidad funcional.
6. Pasos para realizar una acción. El software debe estar diseñado para minimizar el número de pasos que debe realizar el usuario para activar cualquier opción. Lo deseable es que el usuario alcance su objetivo en no más de dos o tres pasos.
7. Mensajes comprensibles. Los mensajes emitidos deben ser cortos, sencillos y redactados en un lenguaje claro para el usuario no técnico.
8. Controles temporales. Evitar los controles de interfaz de usuario que se extinguen o desaparecen después de un tiempo determinado. Si este comportamiento es importante para la aplicación, debe proporcionarse una interfaz alternativa para estas funciones.
9. Navegación circular. La navegación entre elementos de la interfaz debe ser circular, de forma que el foco vuelva desde el último elemento al primero
10. Redundancia en la información auditiva y visual. La información relevante ofrecida en formato de audio o vídeo por las aplicaciones, debe también ser suministrada en otros formatos alternativos. Por ejemplo, subtítulos de la pista de audio en un vídeo o audiodescripción para los contenidos multimedia.
11. Color. La utilización del color es importante para realzar o resaltar la información, pero no debe usarse nunca como la única forma de transmitirla.
12. Combinación de colores. Deben proporcionarse combinaciones de colores predefinidas que hayan sido diseñadas teniendo en cuenta las necesidades de las personas con diversidad funcional visual. Los servicios de accesibilidad de algunos sistemas operativos incluyen la funcionalidad de establecer la visión de la pantalla en alto contraste o con combinaciones de colores para personas con dificultades visuales
13. Alternativas a los avisos sonoros. Los usuarios con dificultad auditiva o que trabajan en entorno ruidosos o cuando deba utilizarse el dispositivo en silencio, deben poder activar una alternativa visual o háptica (por vibración) de los avisos sonoros.
14. Lector de texto. Deben ofrecerse funciones que permitan enviar cualquier información textual a una salida mediante síntesis de voz. La lectura de texto es adecuada para personas con problemas de visión
15. Lector de pantalla. La salida en síntesis de voz debe aparecer inmediatamente después de ocurrir el evento que la originó. Este requisito tiene relación con la utilización de un lector de pantalla.

Usabilidad en la aplicación desarrollada

Durante el proceso de creación y desarrollo de la aplicación, los puntos y facetas concretas seguidas para conseguir una buena usabilidad por parte de nuestros usuarios son:

Interfaz inicial con grandes botones y colores diferenciados (Imagen 3.11)



(Imagen 3.11 Pantalla inicial de la aplicación)

Pregunta redundante a la pregunta inicial mediante voz si no ha sido pulsada ninguna tecla pasados unos segundos y también captura de la respuesta mediante voz.

Posibilidad de recibir los resultado finales mediante sintetizador de voz además de por texto.

Mensajes sencillos tanto para dar instrucciones como para dar los resultados de la aplicación.

Posibilidad de navegación circular, volviendo a captar nuevos objetos mediante un sencillo botón al final de la aplicación.

3.5. CASOS DE USO

Tras haber estudiado las herramientas disponibles y las necesidades específicas de los usuarios a los que está destinada esta aplicación, se definieron las acciones a realizar por el usuario y los requisitos funcionales y no funcionales del software a programar.

La aplicación ha sido diseñada para tener un solo tipo de usuario, cuyas acciones en relación a ésta serán las siguientes:

- El usuario puede elegir el método de salida de los resultados mediante botones.
- El usuario puede elegir el método de salida de los resultados mediante reconocimiento de voz.
- El usuario puede realizar una imagen.
- El usuario puede volver a tomar otra imagen tras la muestra de los resultados.

Requerimientos funcionales

- La aplicación podrá interactuar con el usuario a través de la pantalla táctil o de una interfaz de audio, proporcionando al usuario la capacidad de decidir cuál de estas opciones utilizar.
- La aplicación determinará el color principal de una escena capturada con la cámara del dispositivo móvil de entre una opción de 64 posibilidades.
- La aplicación comunicará al usuario el resultado de la identificación de color.

Requerimientos no funcionales

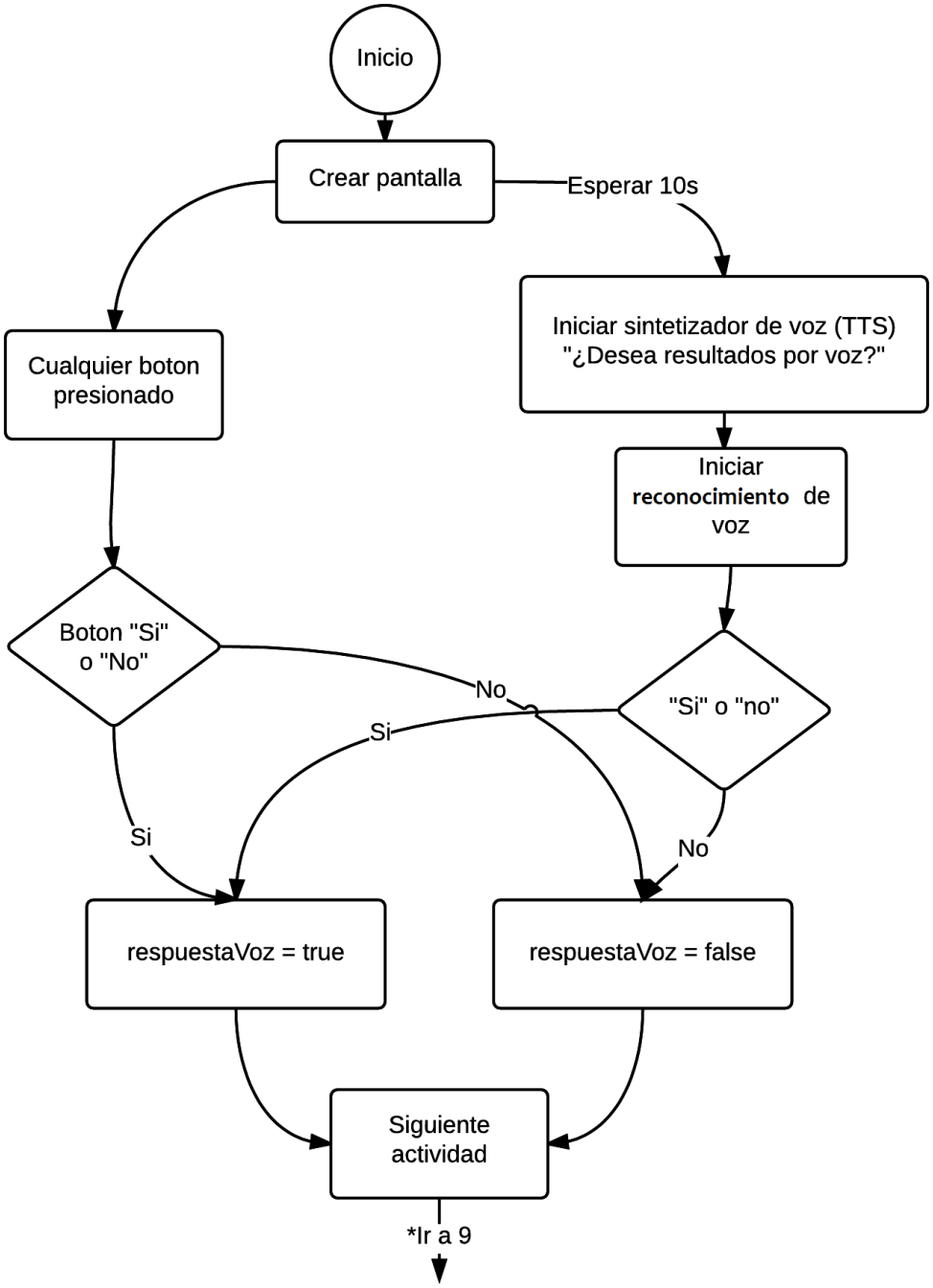
Estos son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. En esta aplicación los requisitos no funcionales son:

- Funcionar en dispositivos móviles con Android 4.1.2 o superior.
- Visualización correcta de la pantalla.
- Móvil con cámara digital.
- Conexión a internet para acceder al servicio de reconocimiento de voz.
- Acceso al micrófono para captar la voz del usuario.

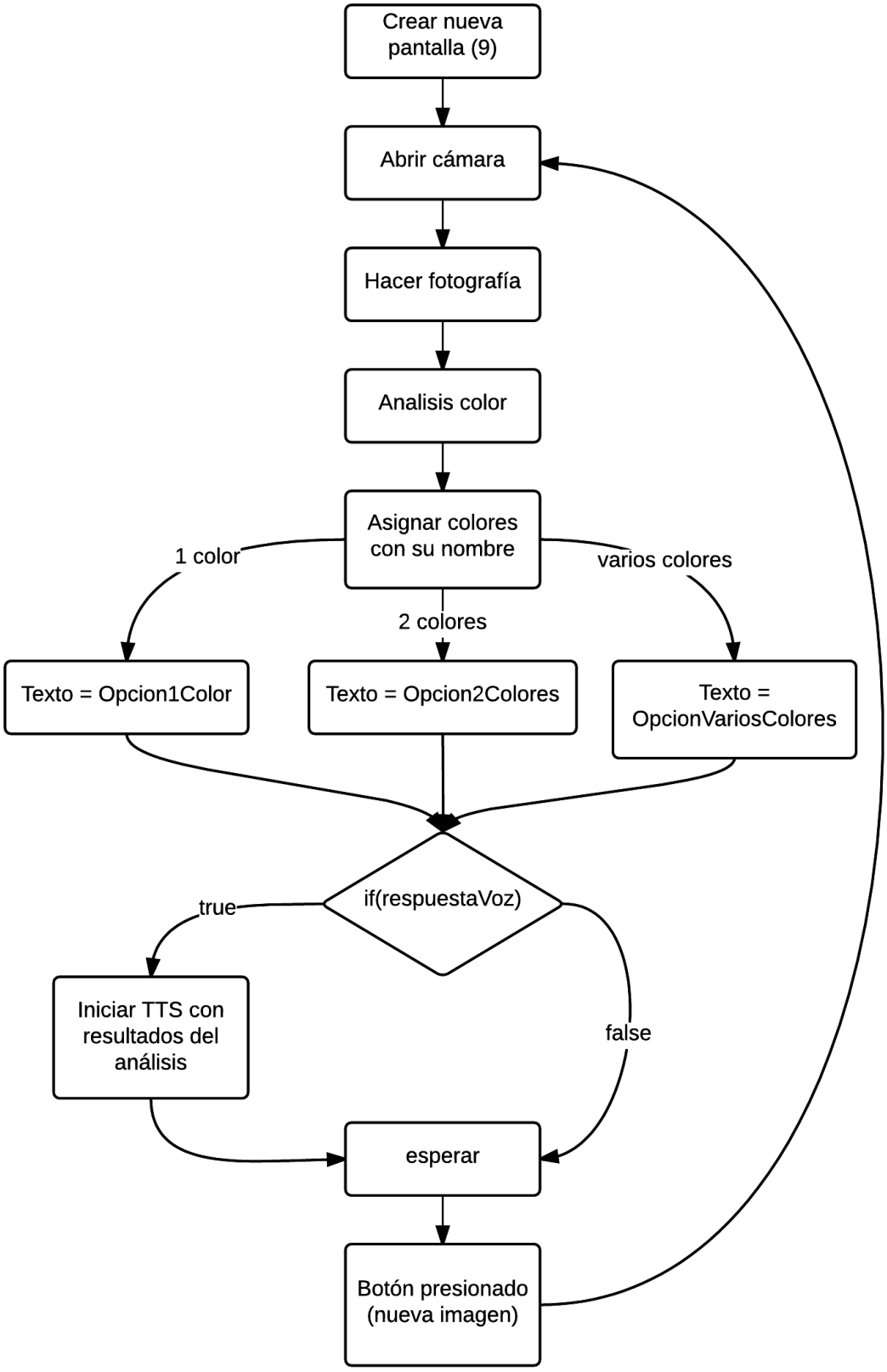
4. FUNCIONAMIENTO DE LA APLICACIÓN

4.1. INTRODUCCIÓN

La aplicación se basa en dos pantallas principales o *layouts*. El paso de una a otra se realizará mediante la toma de una fotografía del objeto. El diagrama de flujo o pseudocodigo de la primera pantalla lo podemos ver la imagen 4.1, mientras que el de la segunda pantalla lo vemos en la imagen 4.2.



(Imagen 4.1 Diagrama de flujo de la aplicación)



(Imagen 4.2 Diagrama de flujo de la aplicación)

Iniciamos el proceso que sigue la aplicación en la imagen 4.1. Tras la apertura de la aplicación se generará la parte visual con que el usuario puede interactuar, al mismo tiempo se iniciará un contador de 10 segundos invisible para el usuario. Durante este tiempo se puede elegir mediante dos grandes botones si se quieren los resultados mediante voz y texto o solo texto. Si pasado este tiempo no ha habido respuesta, la aplicación preguntará esto mediante voz y esperará recibir una respuesta también mediante la voz del usuario. Una vez que la aplicación conoce que tipo de resultados quiere el usuario, pasamos al diagrama de la imagen 4.2.

Una vez hemos pasado a la siguiente fase de la aplicación, en la que se generará la nueva pantalla por ahora oculta, ya que al mismo tiempo se realiza un *intent*, una acción más o menos automática, se abre la cámara para realizar una fotografía. Tras aceptar la fotografía con la que se está conforme, se inicia todo el algoritmo que calcula y clasifica los colores asignándoles un nombre. Dependiendo de si ha sido uno, dos o varios colores los que se han detectado, el mensaje que aparecerá en la pantalla será diferente, pero en todos aparecerá el nombre del color o colores principales. Si al inicio se eligió la opción de respuesta mediante voz, en este momento se activará nuevamente el sintetizador de voz y ahora dirá los resultados obtenidos, explicando además como volver a captar un nuevo color.

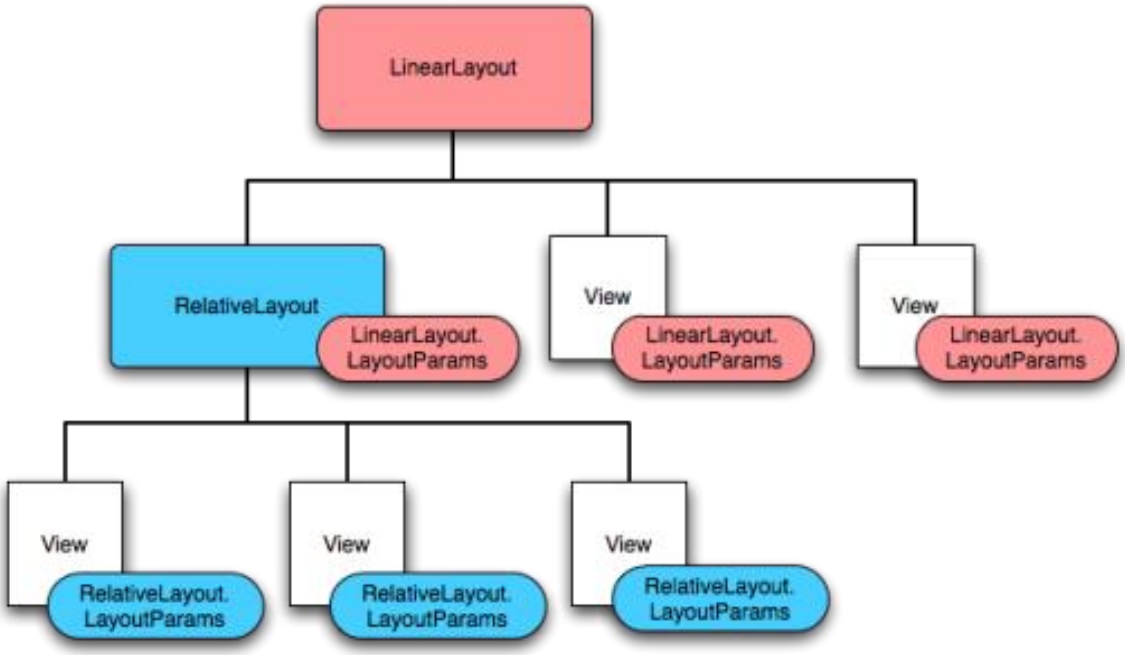
Tras esto la aplicación permanecerá mostrando por pantalla los resultados hasta que sea cerrada o se presione el botón de nueva imagen, en este caso se volvería a abrir la cámara repitiendo el proceso desde ahí.

4.2. INTERFAZ

Los programas en Android se componen de Activitys, que son cada “pantalla” que podemos observar, estos a su vez constan de una parte de código y otra visual.

La parte visual o interfaz está compuesta por diferentes Views, que son cada uno de los controles, widgets o elementos gráficos básicos que permiten la interacción con el usuario. En Android tenemos una serie de vistas predefinidas, como el TextView, Button, checkbox entre otras, pero además podemos crearnos una nosotros mismos.

Cada interfaz se organiza de manera jerárquica, en la que suele haber un LinearLayout en la cúspide, el cual es una manera de organizar los siguientes elementos en filas lineales. Un ejemplo de cómo se podría organizar lo podemos ver en la imagen 4.3.



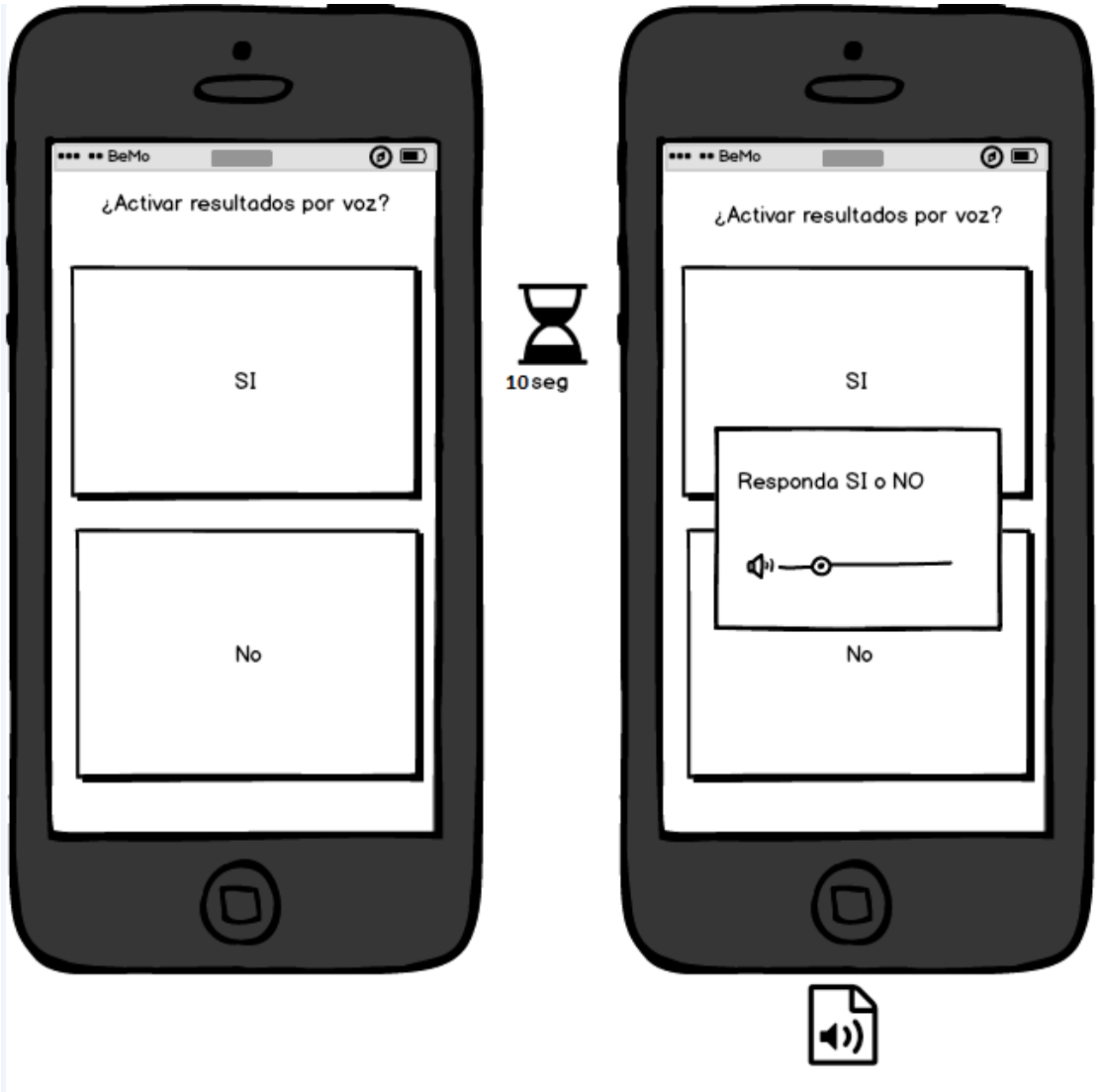
(Imagen 4.3 Ejemplo jerarquía de interfaz)

A la hora de realizar el interfaz de la aplicación hay que continuar con el objetivo puesto en la accesibilidad para que el uso de esta resulte fácil y cómodo para los usuarios. Vamos a recordar algunos de los puntos más importantes a tener en cuenta en este momento del desarrollo:

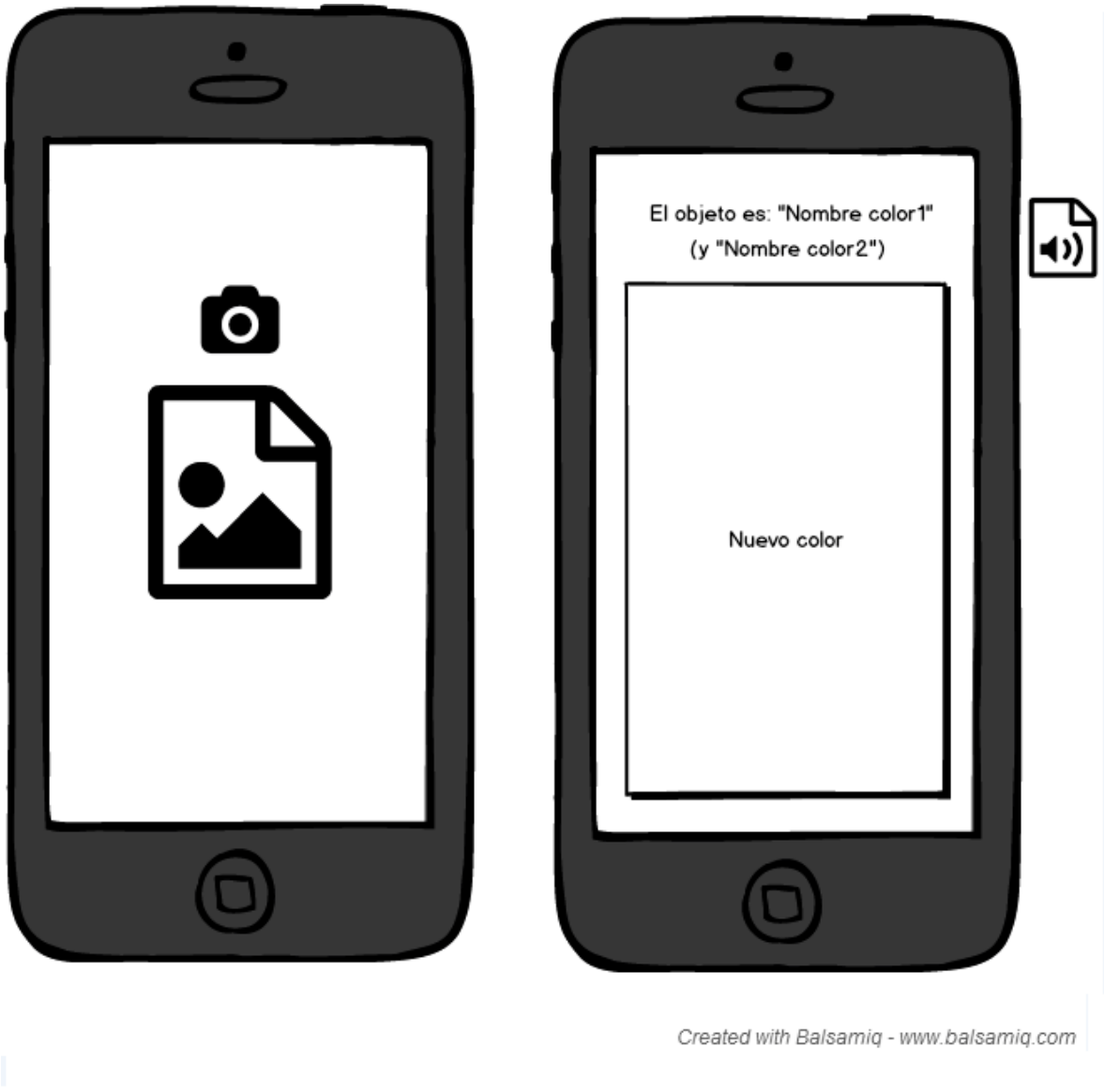
1. Identificación de objetos de la interfaz de usuario de forma genérica, todos los mensajes se deben poder entender sin dificultades con un lenguaje claro y sencillo.

2. Mensajes comprensibles. Los mensajes emitidos deben ser cortos, sencillos y redactados en un lenguaje claro para el usuario no técnico.
3. Color. La utilización del color es importante para realzar o resaltar la información, pero no debe usarse nunca como la única forma de transmitirla.
4. Combinación de colores. Deben proporcionarse combinaciones de colores predefinidas que hayan sido diseñadas teniendo en cuenta las necesidades de las personas con diversidad funcional visual.

Basándonos en esto se ha realizado un esquema (imagen 4.4 y 4.5) del resultado buscado en las diferentes interfaces de la aplicación y las etapas entre estas que nos ayudará a ver con mucha más claridad los siguiente pasos del desarrollo del interfaz.



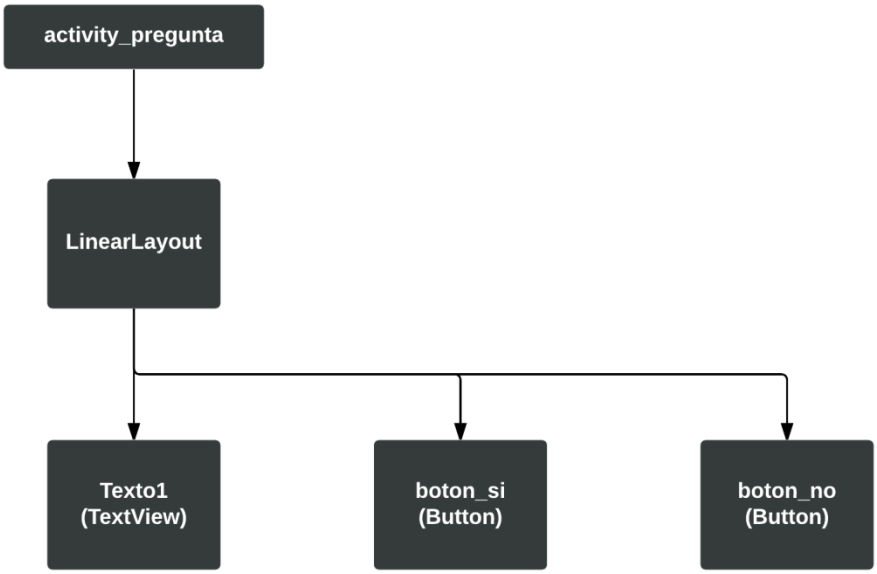
(Imagen 4.4 Esquema del interfaz 1)



(Imagen 4.5 Esquema del interfaz 2)

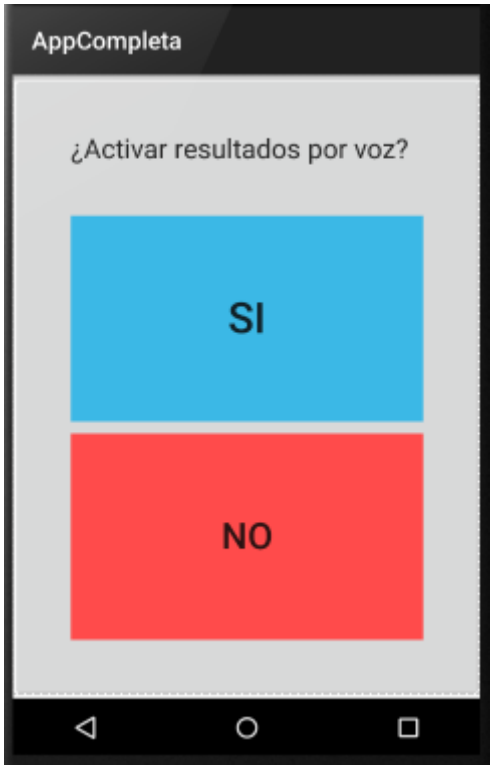
Interfaz de la aplicación

Una vez que tenemos el esquema de cómo queremos que sea la interfaz, llega el momento de aplicarlo en nuestra aplicación. El primer paso será realizar un diagrama (Imagen 4.6) con la jerarquía de la primera pantalla, que se llama *activity_pregunta*.



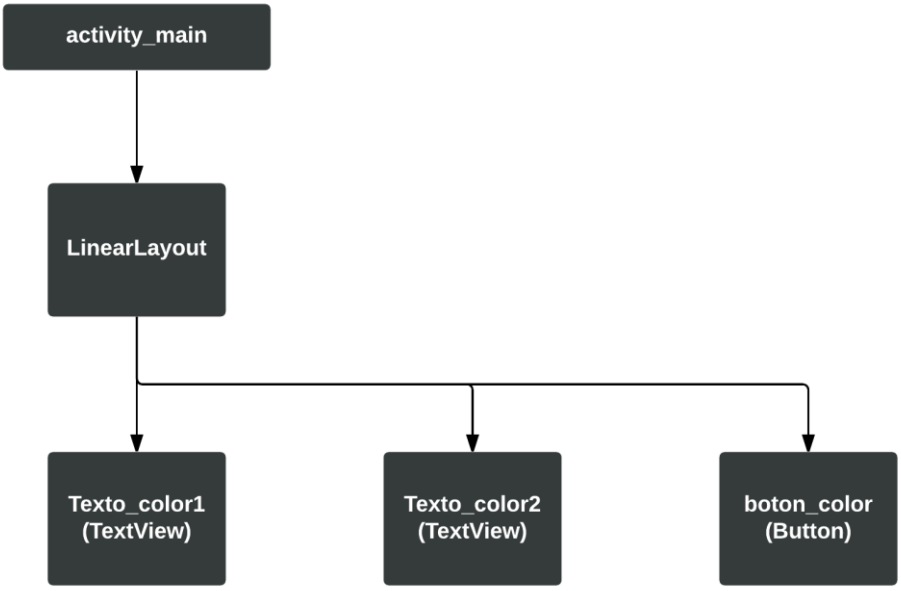
(Imagen 4.6 Jerarquía de la primera pantalla)

Tras esto se realiza la interfaz en sí misma usando la herramienta de Android Studio:



(Imagen 4.7 Primer interfaz de la aplicación)

Volvemos a realizar un esquema (Imagen 4.8) con la jerarquía de la siguiente pantalla, muy parecido al anterior, que en este caso se llama *activity_main*.



(Imagen 4.8 Jerarquía de la segunda pantalla)

Del mismo modo y por último realizamos la interfaz principal, que mostrará los resultados que hayan sido obtenidos por el algoritmo:

(Imagen 4.9 Segundo interfaz de la apicación)

Ahora que ya vemos el resultado final de las interfaces, podemos ver claramente en que partes se han usado los puntos más importantes en la accesibilidad de la aplicación y un extracto del código que define el primer interfaz (imagen 4.10):

En la primera pantalla (Imagen 4.7) se han usado tanto los puntos 1 y 2 haciendo lo más claro posible el texto que pregunta sobre los resultados, como los puntos 3 y 4 dado que se han usado colores para mejorar la manera en la que se transmite y comprende la información, teniendo cuidado que estos colores no sean demasiado llamativos o se puedan confundir entre sí, usando colores fácilmente diferenciables incluso para daltónicos.

En la segunda pantalla (Imagen 4.9) se vuelven a usar los puntos 1 y 2, tanto en la manera que se simplifica la manera de decir el color del objeto como el nombre del color en sí mismo, que se ha intentado hacer lo más común y fácil posible. Además se ha usado el punto 3, utilizando colores para resaltar información, en este caso el botón para tomar una nueva imagen.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="48dp"
    style="@style/Base.Widget.AppCompat.Button">

    <TextView
        android:id="@+id/texto_color1"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_gravity="center_horizontal"
        android:layout_weight="0.8"
        android:text="¿Activar resultados por voz?"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_marginTop="0dp" />

    <Button
        style="@style/Base.Widget.AppCompat.Button"
        android:id="@+id/boton_si"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_gravity="center_horizontal"
        android:layout_weight="2"
        android:text="Si"
        android:background="#33B5E5"
        android:clickable="true"
        android:layout_marginBottom="5dp"
        android:textSize="35dp" />
```

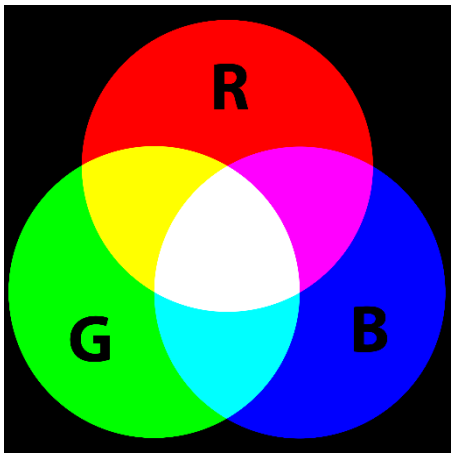
(Imagen 4.10 Extracto del programa)

4.3. DETERMINACIÓN DEL COLOR

Para realizar el análisis del color propiamente dicho, lo primero que se ha realizado es la obtención de los colores en coordenadas de color RGB. A continuación, vamos a explicar qué significa y como se puede trabajar con estos valores.

4.3.1. Colores RGB

El término RGB es la sigla en inglés de red, green, blue (en castellano “rojo, verde y azul”) y hace referencia a la composición del color en términos de la intensidad de los



colores primarios de la luz como podemos ver en este esquema.

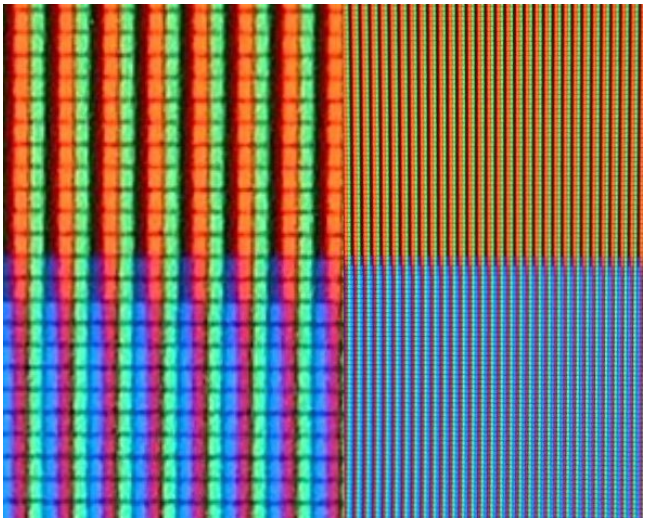
(Imagen 4.11 Esquema intensidad RGB)

Esto no es más que un método para poder codificar un color de forma digital. Así, a cada color le corresponderá un código compuesto por tres valores, las cuales serán el valor de la intensidad de cada uno de los colores primarios.

Esta codificación podrá ser en forma hexadecimal (formato HTML), donde cada una de las intensidades tomará un valor entre #00 y #FF y serán colocadas en orden formando el código en sí mismo (color RGB = #RRGGBB), o en su forma decimal, que se obtendría pasando a decimal el código completo HTML el cual tendrá un valor entre #000000 y #FFFFFF. En muchas ocasiones, por comodidad principalmente, se trabaja con los términos R, G y B de manera separada, tanto en hexadecimal como en decimal, por lo que el código a usar tendrá este aspecto: (R, G, B) con valores nuevamente entre 0 y #FF en hexadecimal y 255 en el caso decimal.

El código RGB se ha ido imponiendo en el mundo de la informática principalmente debido a que la gran mayoría de los monitores son de 24 bits y esto permite usar los 16,7 millones de colores que se pueden obtener mediante este formato.

En la imagen 4.12 podemos ver como usan este formato los monitores en concreto, de tecnología LCD. Usando píxeles de los tres colores con los que estamos trabajando, se varía la intensidad de éstos para formar los distintos colores que podemos ver.

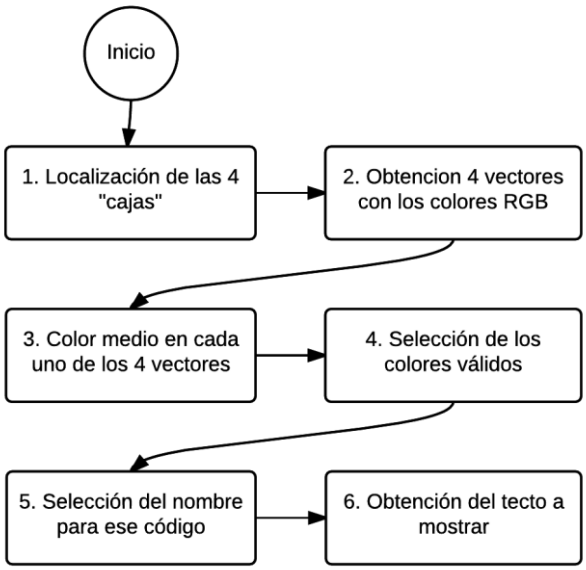


(Imagen 4.12 Pantalla LCD)

4.3.2. Uso en la aplicación

Una vez que se tiene claro cómo se va a codificar el color a partir de ahora, podemos centrarnos en el proceso que sigue la aplicación desde la imagen tipo bitmap que obtiene a partir de la cámara, hasta que se averigua el nombre del color que estamos estudiando.

Los pasos, resumidos en un diagrama de flujo, que se realizan durante todo este proceso son:

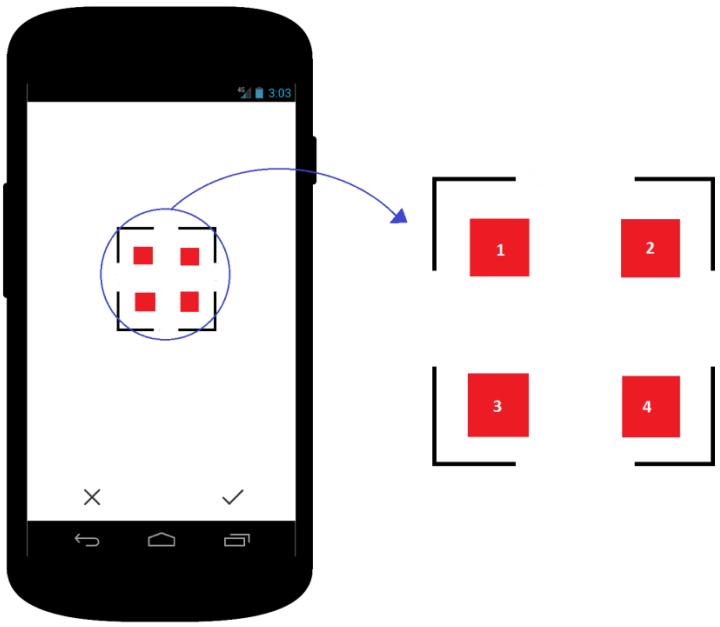


(Imagen 4.13 Esquema de detección de color)

1.- Localización de las 4 “cajas”

En este paso se usan funciones para obtener el tamaño de la imagen que ha sido tomada, para averiguar las coordenadas de estas “cajas”, pero ¿qué son?

Son simplemente cuatro zonas de 10x10 pixels en torno al centro de la imagen, que serán usadas para ser comparadas entre sí y de este modo conseguir saber si el objeto al cual hemos tomado la fotografía es de un solo color, de dos, o de varios, como podría ser el caso de una prenda estampada.



(Imagen 4.14 Esquema de las zonas de detección del color)

En este esquema se puede ver la localización aproximada de las zonas que son estudiadas, puesto que la distancia entre ellas variará con la resolución de la cámara usada.

El código usado en concreto será:

```
int width = imagen.getWidth();
int height = imagen.getHeight();

//Puntos de inicio en el frag1
int start_x1 = ((width/2)-15);
int start_y1  = ((height/2)-15);
//Puntos de inicio en el frag2
int start_x2 = ((width/2)+5);
int start_y2  = ((height/2)-15);
//Puntos de inicio en el frag3
int start_x3 = ((width/2)-15);
int start_y3  = ((height/2)+5);
//Puntos de inicio en el frag4
int start_x4 = ((width/2)+5);
int start_y4  = ((height/2)+5);
```

(Imagen 4.15 Extracto del código de localización de zonas)

Como se puede ver esto constará de dos funciones que obtienen el tamaño de la imagen y unas operaciones para calcular y guardar los puntos de origen de las zonas.

2.- Obtención de los vectores RGB

Una vez que tenemos las zonas designadas, el siguiente paso es por medio de la función *getPixels()*,propia de Android, que obtiene los valores numéricos del color de cada uno de los píxeles que éstas contienen.

Esta función nos devolverá un vector compuesto por valores enteros de tipo decimal los cuales pueden ser pasados a hexadecimal y de este modo ser usados como códigos RGB con el color que estábamos buscando. Estos vectores serán guardados en las variables *frag1-frag4* correspondientes, que son vectores de longitud 100, los cuales se pueden asociar a cada uno de los cuatro fragmentos analizados.

Este proceso puede ser llevado a cabo mediante el siguiente código, el cual como se ha explicado es guardado en las variables *frag* y necesitará cierta información de entrada como la anchura de la zona analizada o las coordenadas de inicio de esta:

```
//Obtencion de los 4 fragmentos de color
imagen.getPixels(frag1, 0, 10, start_x1, start_y1, 10, 10);
imagen.getPixels(frag2, 0, 10, start_x2, start_y2, 10, 10);
imagen.getPixels(frag3, 0, 10, start_x3, start_y3, 10, 10);
imagen.getPixels(frag4, 0, 10, start_x4, start_y4, 10, 10);
```

(Imagen 4.16 Extracto del código de obtención de los vectores)

3.- Color medio en cada uno de los 4 vectores

Ahora que ya tenemos los colores codificados de cada punto de las zonas que nos interesan, el siguiente paso es hacer la media en cada una de las zonas, para evitar que cualquier fallo o pequeña diferencia en la imagen distorsione nuestros datos.

Para abordar esta tarea he creado una función que dependiendo de los datos de entrada, hará la media de uno de los componentes, R, G o B, del vector que le haya sido introducido.

Por lo tanto habrá que repetir este proceso para cada uno de los componentes en cada una de las cuatro zonas. Un ejemplo de cómo se realiza esto para la primera zona sería:

```
//Obtencion de las componentes medias RGB en cada fragmento
RR1 = (mediaColor(frag1, 2, 4)); //Dara el rojo (RR) ponderado en decimal
GG1 = (mediaColor(frag1, 4, 6)); //Dara el verde (GG) ponderado en decimal
BB1 = (mediaColor(frag1, 6, 8)); //Dara el azul (BB) ponderado en decimal
```

(Imagen 4.17 Extracto del código de selección de parámetros RGB)

El funcionamiento de este proceso es muy sencillo, ya que se trata de un bucle *for* el cual transforma a hexadecimal el código del color, no podemos olvidar que estaba en decimal tras su obtención, selecciona solo los dos dígitos que codifican el color que en cada caso estemos buscando, lo vuelve a pasar a decimal para facilitar el trabajo con estos números y finalmente realiza la media de todos los elementos que le han ido pasando.

El núcleo principal de esta función es:

```
for(i=0;i<100;i++){
    numAuxS = Integer.toHexString(pixels[i]);   //Obtenemos el color en hex -> 0xFFRRGGBB
    numColorS = numAuxS.substring(a,b); //a y b son las posiciones en el num hex
    //2 a 4 ->RR, 4 a 6->GG, 6 a 8->BB
    numColor = Integer.parseInt(numColorS, 16); //Pasa el color a decimal
    sumatorio += numColor;

}

media = (sumatorio / 100);              //media de colores en decimal
```

(Imagen 4.18 Extracto del código del color medio)

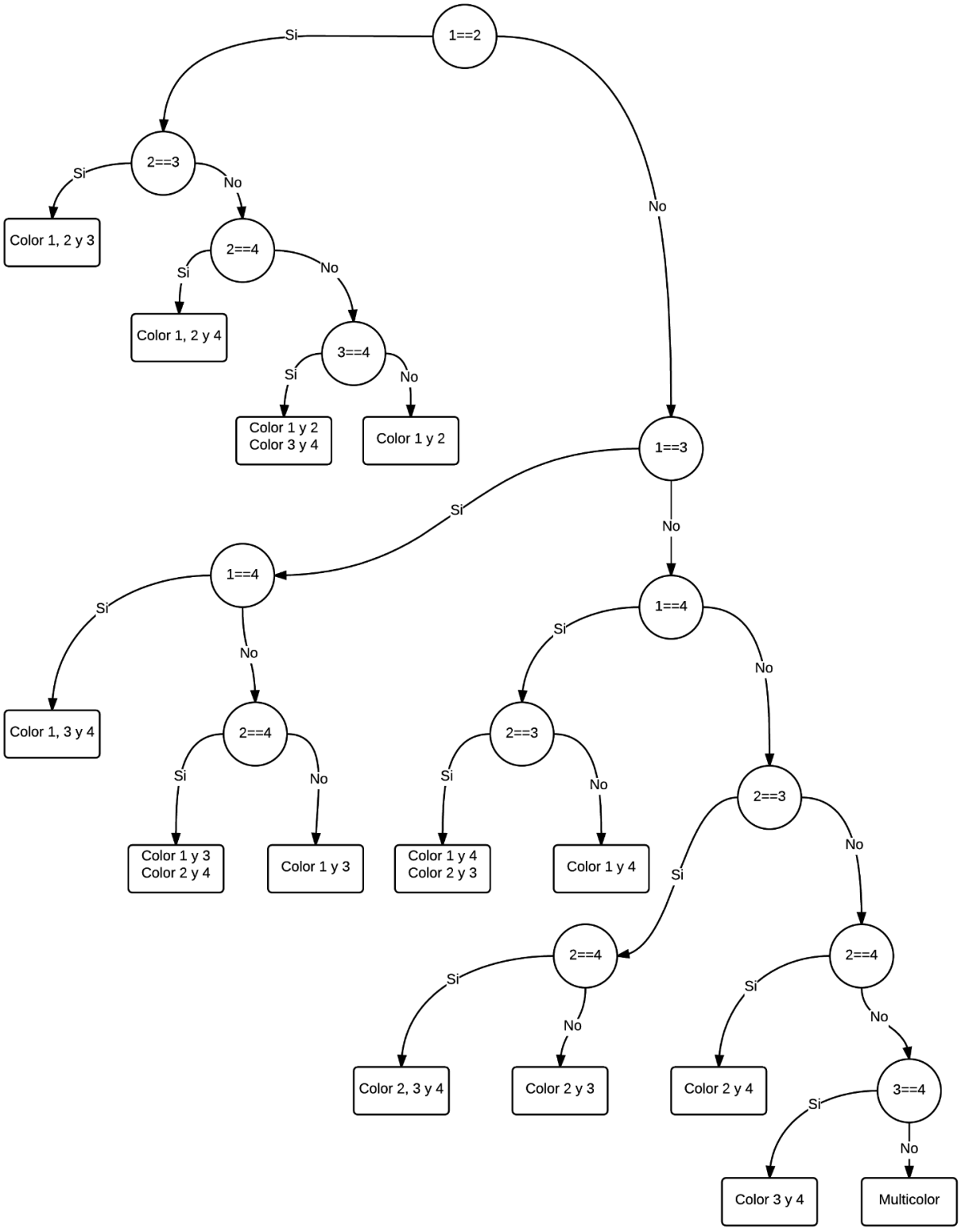
4.- Selección de los colores válidos

Esta tarea que a priori puede parecer sencilla, se complica sobre todo para el caso en el que tenemos cuatro colores distintos.

Se ha fijado que el algoritmo pueda distinguir entre todos estos casos:

- Si hay solo un color, el cual aparecerá en las cuatro zonas estudiadas, será considerado el color principal.
- Si tres de las zonas comparten color y solo una de ellas tiene un valor diferente, éste será tomado como anomalía, ya sea por ser muy poco dominante en el objeto o simplemente un detalle que casualmente ha acabado en una de nuestras zonas.
- Si dos de las zonas tienen un color coincidente pero las otras tienen un color distinto a las primeras y a la vez diferente entre sí,se tomará como color principal el que aparecía en dos de las zonas, mientras que los colores que aparecían en las otras zonas serán descartados por los mismos motivos que en el caso anterior. Hay que destacar que en este caso la posibilidad de obviar un color principal debe ser tomada en cuenta y sería una de las posibles mejoras del algoritmo en versiones futuras.
- Si hay dos zonas iguales y las otras dos, en este caso sí que coinciden. En este caso serán tomados como colores principales ambas parejas, sin importar la posición en la que se encuentren.
- Finalmente el caso de que ninguna de las zonas estudiadas coincida con ninguna de las otras. En este no habrá ningún color que predomine sobre el resto, pero podemos intentar conseguir un tono más genérico de objeto haciendo la media entre todas las zonas, y quizás hacernos una idea del color mayoritario. Este caso podría ser el de una prenda estampada o un objeto multicolor.

Una vez que tenemos los detalles de cómo ha de funcionar este algoritmo se puede diseñar un diagrama de flujo (Imagen 4.19) con el que partiendo de los cuatro colores y comparándolos entre sí, obtengamos uno de los casos anteriormente descritos.



(Imagen 4.19 Diagrama de selección de colores)

Si nos fijamos, esto no más que el pseudo-código de esta función. Por lo tanto una vez está claro este punto el siguiente paso programar el esquema, en este caso con un conjunto de bucles *if* para ir avanzando por el diagrama.

Un ejemplo de la función que realizada todo esto es:

```
public void seleccionColor(int RR1,int GG1, int BB1, int RR2,int GG2, int BB2,
                           int RR3,int GG3, int BB3, int RR4,int GG4, int BB4){

    if(CompColor(RR1,GG1,BB1,RR2,GG2,BB2)){   //1=2?

        if(CompColor(RR2,GG2,BB2,RR3,GG3,BB3)){//2=3?
            //color 1,2,3 Fallo 4
            numColores = 1;
            RFinal1 = ((RR1+RR2+RR3)/3);
            GFinal1 = ((GG1+GG2+GG3)/3);
            BFinal1 = ((BB1+BB2+BB3)/3);
        }else{//no 2=3
            if(CompColor(RR2,GG2,BB2,RR4,GG4,BB4)) {//2=4?
                //color 1,2,4 fallo 3
                numColores = 1;
                RFinal1 = ((RR1+RR2+RR4)/3);
                GFinal1 = ((GG1+GG2+GG4)/3);
                BFinal1 = ((BB1+BB2+BB4)/3);
            }else{//no 2=4
                if(CompColor(RR3,GG3,BB3,RR4,GG4,BB4)){//3=4?
                    //Color 1, 2 y Color 3,4
                    numColores = 2;
                    RFinal1 = ((RR1+RR2)/2);
                    GFinal1 = ((GG1+GG2)/2);
                    BFinal1 = ((BB1+BB2)/2);

                    RFinal2 = ((RR4+RR3)/2);
                    GFinal2 = ((GG4+GG3)/2);
                    BFinal2 = ((BB4+BB3)/2);

                }else{//no 3=4
                    //Color 1, 2 fallo 3,4
                    numColores = 1;
                    RFinal1 = ((RR1+RR2)/2);
                    GFinal1 = ((GG1+GG2)/2);
                    BFinal1 = ((BB1+BB2)/2);

                }
            }
        }
```

(Imagen 4.20 Extracto del código de selección de colores)

Como podemos ver se le introducen todos los valores RGB de las cuatro zonas en estudio, tras esto se inician los condicionales para seguir el diagrama y se finaliza calculando el componente medio correspondiente en cada caso al color principal además de actualizando una variable que será usada más tarde con la que obtenemos el número de colores principales que hay.

Se puede observar que la comparación se realiza a través de la función externa CompColor(), ya que la cantidad de colores que se pueden generar mediante el sistema RGB es tan elevada que se hace necesario establecer unos márgenes en torno a los cuales consideraremos que un color es el mismo que otro o no.

Puesto que de estos márgenes dependerá la precisión con la que el algoritmo considere a un color igual a otro, es de gran importancia que estos valores no sean ni demasiado pequeños, ya que consideraría distintos colores que son prácticamente indiferenciables para nuestro ojo, ni demasiado grandes, porque igualaría colores que son claramente diferentes arruinando todo el trabajo anterior y posterior.

Vista la importancia de este margen, finalmente he escogido que la función considere dos colores como iguales si la diferencia de los tres componentes del RGB es menor a 32 puntos de intensidad en su valor decimal (#20 en hexadecimal) en cada uno de los componentes. Es decir, los tres componentes han de estar a menos de esta distancia para que el color sea considerado como igual.

Esta diferencia en porcentaje respecto a la intensidad total del 7%, lo que teniendo en cuenta que raramente se dará la diferencia máxima en más de un componente, hará que la diferencia "total" de los colores comparados sea siempre menor de este valor.

El código que realiza esta comparación es:

```
//Comprueba si dos colores tienen los parametros R, G y B a menos de 32 puntos
public boolean CompColor (int RRdec1, int GGdec1,int BBdec1,int RRdec2, int GGdec2,int BBdec2){
    int margen = 32;

    return (((RRdec1>(RRdec2-margen))&&(RRdec1<(RRdec2+margen)))&&  //Comprobamos si RR dentro de margen
            ((GGdec1>(GGdec2-margen))&&(GGdec1<(GGdec2+margen)))&& //Comprobamos si GG dentro de margen
            ((RRdec1>(BBdec2-margen))&&(BBdec1<(BBdec2+margen))));  //Comprobamos si BB dentro de margen
}
```

(Imagen 4.21 Extracto del código comparación de colores)

Así, una vez terminado todo este proceso hemos conseguido, partiendo de cuatro colores diferentes, saber de cuáles se componen los colores principales y sus valores concretos.

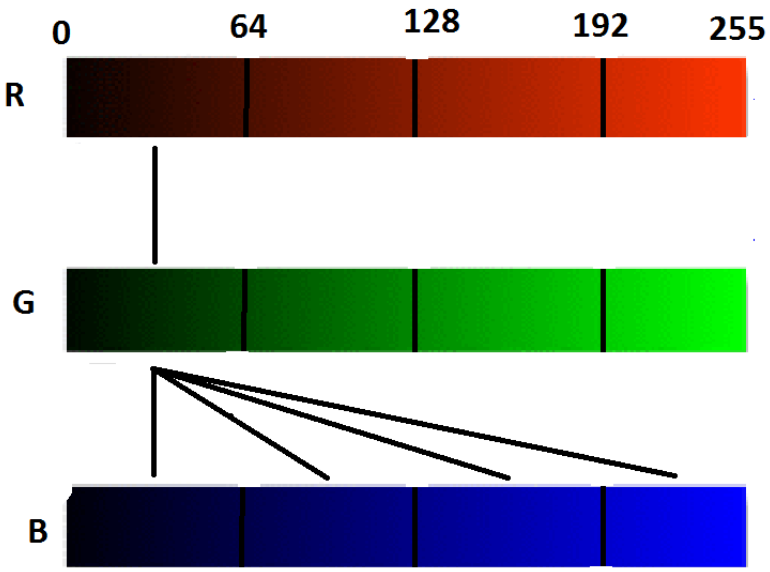
5.- Selección del nombre para ese código

Cuando hemos llegado a este punto, el siguiente paso es asignar a cada color su nombre para ser rápidamente interpretado. El problema inmediato surge cuando pensamos que hay 16,7 millones de colores, por lo que resulta imposible asignar un nombre a cada color exacto.

El procedimiento más sencillo en este caso es seleccionar unos márgenes dentro de cada color y que todos los colores que entren en ese rango sean llamados del mismo modo.

En este caso en particular, me he decantado por dividir cada rango de intensidades del RGB en cuatro partes más pequeñas y de este modo considerar iguales a todos los colores que se encuentren dentro del mismo cuarto de intensidad en rojo, verde y azul.

En el siguiente esquema (Imagen 4.22) se puede ver un ejemplo de cómo obtener los cuatro primeros colores con este sistema:



(Imagen 4.22 Esquema de obtención de colores)

Si bien a primera vista parece muy simple este tipo de decisión, hay que tener en cuenta que con cuatro divisiones en las tres intensidades que tenemos nos da un total de 4^3 = 64 colores distintos, lo que es una cantidad aceptable si la comparamos con cinco o seis divisiones de cada color, 125 y 216 colores distintos respectivamente, más aun si tenemos en mente el objetivo de la aplicación, ayudar a gente con problemas de visión, que no necesitarán una concreción excesiva en el nombre del color. De igual modo, pese a que es fácil que nuestro ojo distinga más de esos 64 colores, en el uso diario prácticamente nadie o en muy pocos campos se necesitarán nombres más detallados que los de estos tonos.

A la hora de llevar a cabo esto en la aplicación, se realiza una búsqueda mediante condicionales *if*, hasta delimitar para cada color los márgenes en los que se encuentra y para posteriormente asignar el nombre de color más común o conocido que caracteriza ese rango.

Un posible inconveniente es que algunos colores dentro de los 64 grupos tengan nombres diferenciables debido a que solamos distinguir entre ellos o el caso contrario, en el que varios grupos sean denominados de la misma manera, además de que al existir relativamente pocos colores, la iluminación tiene un gran importancia, pues un exceso o la falta de esta puede hacer cambiar enormemente el color obtenido por la cámara de teléfono móvil.

Este es un extracto del inicio del código (Imagen 4.23), en el que se puede ver como se encuentran los primeros 16 grupos de colores:

```
if (R < 64){
    if (G < 64){
        if (B < 64){
            nombre = "Negro";
        }else if(B < 128){
            nombre = "Azul marino";
        }else if(B < 192){
            nombre = "Azul";
        }else{
            nombre = "Azul";
        }
    }else if(G < 128){
        if (B < 64){
            nombre = "Verde";
        }else if(B < 128){
            nombre = "Turquesa oscur
        }else if(B < 192){
            nombre = "Azul";
        }else{
            nombre = "Azul";
        }

    }else if(G < 192){
        if (B < 64){
            nombre = "Verde";
        }else if(B < 128){
            nombre = "Verde claro";
        }else if(B < 192){
            nombre = "Turquesa";
        }else{
            nombre = "Azul claro";
        }

    }else{
        if (B < 64){
            nombre = "Verde lima";
        }else if(B < 128){
            nombre = "Verde";
        }else if(B < 192){
            nombre = "Turquesa";
        }else{
            nombre = "Azul claro";
        }

    }
}else if(R < 128){
```

(Imagen 4.23 Extracto del código de asignación de nombres)

En las imágenes 4.24 y 4.25 podemos ver unas tablas con todos los nombres de los colores usados así como una muestra del color central representativo en cada uno de los rangos determinados para cada uno de los rangos.

	B = 32	B = 96	B = 160	B = 224
R = 32 G = 32	Negro	Azul marino	Azul	Azul
R = 32 G = 96	Verde	Turquesa oscuro	Azul	Azul
R = 32 G = 160	Verde	Verde claro	Turquesa	Azul claro
R = 32 G = 224	Verde lima	Verde	Turquesa	Azul claro
R = 96 G = 32	Granate	Violeta	Morado	Morado
R = 96 G = 96	Amarillo oscuro	Gris	Lila	Azul
R = 96 G = 160	Verde	Gris verdoso	Gris azulado	Azul cielo
R = 96 G = 224	Verde lima	Verde claro	Aguamarina	Azul claro

(Imagen 4.24 Primera tabla con los colores usados)

	B = 32	B = 96	B = 160	B = 224
R = 160 G = 32	Rojo	Fucsia	Rosa oscuro	Morado
R = 160 G = 96	Marrón	Gris rojizo	Morado	Morado
R = 160 G = 160	Amarillo mostaza	Gris amarillento	Gris	Lila
R = 160 G = 224	Verde lima	Verde claro	Verde claro	Turquesa
R = 224 G = 32	Rojo	Fucsia	Rosa	Lila
R = 224 G = 96	Naranja	Rosa grisaceo	Rosa	Lila
R = 224 G = 160	Amarillo mostaza	Marrón claro	Rojo pálido	Lila
R = 224 G = 224	Amarillo	Amarillo claro	Crudo	Blanco

(Imagen 4.25 Segunda tabla con los colores usados)

6.- Obtención del texto a mostrar

Una vez tenemos el nombre del color como tal, solo nos queda crear el texto que será mostrado dependiendo de la cantidad de colores que han sido detectados.

Esto se puede hacer usando una variable que muestra el número final de colores a tener en cuenta, la cual ha sido actualizada en la función de selección de colores ya que en cada caso ya era posible saber la combinación de la que se trata.

El código que se encarga de esto es el siguiente:

```
switch (numColores) {
    case 1:
        texto1 = "El color del objeto es: "+ nombreColor1;
        break;
    case 2:
        texto1 = "Los colores del objeto son: "+ nombreColor1;
        texto2 = " y "+ nombreColor2;
        break;
    case 4:
        texto1 = "El objeto tiene varios colores";
        texto2 = " y el principal es:"+nombreColor1;
        break;
    default:
        texto1 = "El color del objeto es: "+ nombreColor1;
        break;
}
```

(Imagen 4.26 Extracto del código de asignación de respuesta)

Como podemos ver se han tenido en cuenta los casos en los que haya un solo color principal, dos colores principales y el caso especial en el que cada zona tiene un color diferente.

Así pues, se termina el apartado de detección del color, desde el inicio con los datos de cada punto en RGB hasta el texto con el nombre de los colores del objeto.

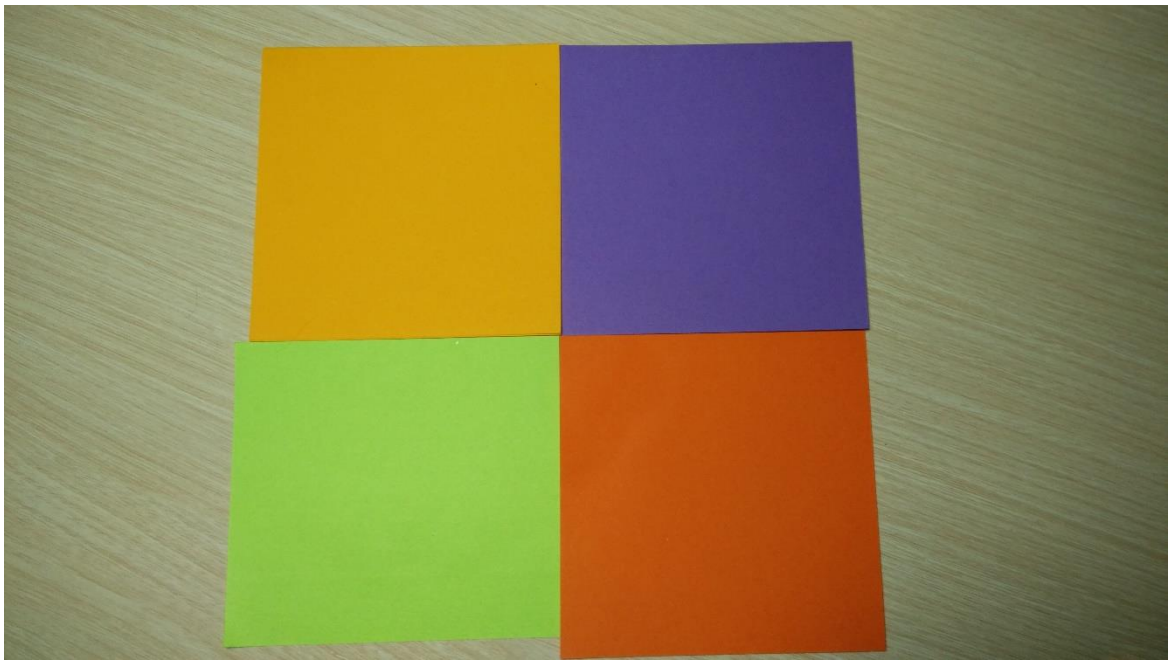
5. PRUEBAS DE LA APLICACIÓN

Una etapa importante del proceso de desarrollo de la aplicación, es determinar cuál es su funcionamiento en cada fase de programación. Para esto el mejor método es ir probando el funcionamiento de cada funcionalidad que queremos implementar en la aplicación por separado, y así ha sido como se ha hecho en este caso.

Desde un principio se ha ido desarrollando cada apartado de manera individual en una pequeña aplicación independiente para poder probar con la mayor eficacia posible el nuevo código que se desarrollaba en cada caso, eliminando al mismo tiempo la posibilidad de que ocurriese algún problema con el código ya existente.

De todos los apartados que se han probado en este punto, cabe destacar el de *detección de colores*, ya que ha sido con diferencia el que tuvo que pasar por más versiones diferentes, unas cinco, hasta llegar a la final en la que el algoritmo de detección de colores fue a prueba y funcionó correctamente. Esto se realizó mediante unas versiones modificadas de la aplicación en las que se mostraba el resultado de cada paso y el valor de cada fragmento de color detectado. Toda esta información no está disponible para el usuario en la aplicación final.

Los objetos utilizados para facilitar todo este proceso han sido láminas de distintos colores (Imagen 5.1) con las que comprobar las distintas funcionalidades de este algoritmo.



(Imagen 5.1 Laminas usadas en las pruebas)

Pruebas de la aplicación final

Una vez que la aplicación estuvo terminada se ha utilizado y testeado numerosas veces. El objetivo principal ha sido comprobar que todo el código funcionaba de la manera esperada además de testear que no se producían fallos inesperados en determinadas ocasiones.

En el caso de que exista un solo color no suele haber problemas, a no ser que la iluminación sea muy mala o excesiva, por lo que el color captado por la cámara no sea el real.

Unos ejemplos de pruebas realizadas sobre varios colores con las láminas de la imagen 5.1 son:

Si usamos las dos láminas de la derecha el resultado que da la aplicación es:

El objeto es: Lila y Naranja

Podrían existir dudas respecto al segundo color, pero dada la importancia de la luz, podemos dar por buena esta prueba.

En la prueba con las dos láminas superiores el resultado que da la aplicación es:

El objeto es Amarillo mostaza y Violeta

En este caso vemos un cambio de nombre para un mismo color, que ha sido debido a un cambio en la iluminación y ajustes internos de la cámara para ajustar la imagen.

Finalmente, si tomamos las cuatro laminas, el resultado obtenido es:

El objeto tiene varios colores y el principal es: Gris amarillento

En este último ejemplo vemos que al mediar los cuatro colores, su suma se acerca al gris, al mezclar colores muy distintos, pero sigue refiriéndose como amarillento por el predominio de este tono en las láminas amarilla y naranja.

Tras esta etapa de testeo se ha continuado con otras dos muy diferentes pero a su vez muy importantes, que han sido llevadas a cabo por usuarios ajenos al desarrollo de la aplicación:

- Pruebas por usuarios potenciales. Este es el grupo más importante para probar el comportamiento de la aplicación, así como sus posibles mejoras y puntos fuertes. Se ha entrevistado a personas con problemas de visión y el resultado ha sido satisfactorio, ya que les ha gustado mucho la posibilidad de tener los resultados mediante sonido y han afirmado que a ellos y a otros muchos les podría ser de ayuda en determinados casos. También mostraron como posible mejora un aumento de los colores detectados y la automatización del proceso de realizar la fotografía.
- Pruebas por usuarios secundarios. Estos no pudieron evitar ponerse en la piel de un usuario con problemas de visión, lo que se ha traducido en que han detectado tanto los mismos puntos fuertes de la aplicación, buena usabilidad adaptada y aceptable detección de color, como las mismas mejoras, una mejora del número de colores detectados y mayor facilidad para realizar las fotografías.

6. CONCLUSIONES

La aplicación que ha sido desarrollada en este Trabajo Fin de Grado ha sido concebida como un instrumento enfocado en las personas con problemas de visión, sean una pérdida total o parcial de ésta o problemas únicamente relacionados con el color como el daltonismo. Con esta aplicación, el dispositivo móvil detecta los colores principales en una escena mediante una cámara y se los transmite mediante texto o voz al usuario.

Se ha creado una aplicación para dispositivos móvil, basada en el sistema operativo Android, la cual ha sido enfocada en todo momento en la usabilidad por parte de sus usuarios, por lo que se han primado dos elementos sobre el resto, la adaptabilidad y la sencillez para no complicar la experiencia de éstos.

La adaptabilidad es fácilmente visible en elementos como las respuestas mediante sintetizador de voz, los botones grandes y bien diferenciados o el manejo por voz en algunas ocasiones. La sencillez ha sido introducida eliminando al máximo los elementos que puedan crear cualquier opción complicada.

Cuando la aplicación sea iniciada, esta preguntará mediante texto si se querrán los resultados mediante voz o no. Si no se ha presionado ningún botón con la respuesta en diez segundos, un mensaje de voz realizará la pregunta, activando tras él una captura de voz para obtener la respuesta. Tras este punto se abre la cámara y se toma una imagen. Para finalizar tras pasar dicha imagen por el algoritmo se muestra el nombre de los colores de la escena mediante un sencillo texto y por voz si así se ha pedido al principio, tras lo cual se da la opción de volver a tomar una imagen nueva mediante un gran botón.

Dificultades

Desde el inicio de este Trabajo Fin de Grado me han surgido gran cantidad de problemas y dificultades. El primero fue mi desconocimiento previo de todas las herramientas usadas, desde la programación de aplicaciones para móviles hasta el propio lenguaje Java en sí mismo, al no ser parte básica del grado que he estudiado, Electrónica y Automática.

La siguiente dificultad en orden de importancia fue abordar el mundo de las aplicaciones y elementos adaptados a personas con problemas de visión, ya que al no ser un tema muy común, los conocimientos que se suelen tener son bastante limitados.

Finalmente han ido surgiendo más dificultades a lo largo de todo el desarrollo del Trabajo, pero ya dentro de lo esperable, como problemas escurridizos en el código o intentar usar bibliotecas externas para el desarrollo del algoritmo de obtención de los colores que resultaba más un obstáculo que una ayuda.

Líneas futuras

En cuanto a encontrar posibles mejoras de la aplicación ha sido fácil gracias a la ayuda de personas con problemas de visión, que probaron la aplicación dando su opinión.

Los dos cambios principales que se podrán realizar son la mejora del algoritmo para detectar más colores y hacerlo de una manera mucho más eficaz, puesto que el que ha sido realizado tiene sus limitaciones en cuanto a la clasificación de colores como diferentes o no y la mejora del proceso de toma de fotografías, pues este depende del software propio de cada móvil por lo que no se controla todo este proceso desde la aplicación siendo la principal barrera en la adaptabilidad total de la aplicación a los usuarios.

Otra posible mejora sería añadir un sistema de detección de voz no dependiente de internet, y por ende aumentar la facilidad de uso de la aplicación además de una disminución de consumo de batería, pese a que la carga de datos son mínimos.

Para acabar también se podría mejorar la cantidad de nombres de colores así como los nombres en sí mismos. Para la mayoría de colores la cantidad usada de 64 diferentes es suficiente, pero hay algunas zonas en las qué podría llegar a ser útil una mayor diferenciación entre estos, además de una posible mejora en el nombre que los designa a la hora de mostrar los resultados.

Valoración personal

Una vez desarrollado y terminado el Trabajo Fin de Grado puedo decir que ha sido una gran experiencia. Principalmente por el aprendizaje de tantos conocimientos nuevos en Java y en el desarrollo de aplicaciones para Android que no habría obtenido si no lo hubiera realizado.

También ha sido fantástico poder ver las necesidades de las personas con problemas de visión y poder poner mi granito de arena ayudándoles.

Puesto que fui miembro del equipo EUPT Bikes, ya sabía que un proyecto de este tipo iba a necesitar una gran cantidad de horas y más aun de esfuerzo, pero finalmente ha sido una experiencia tan satisfactoria como esperaba que fuese.

BIBLIOGRAFIA

http://alvinalexander.com/java/jwarehouse/android/core/java/android/speech/SpeechRecognizer.java.shtml

Último acceso noviembre 2015

http://developer.android.com/intl/es/reference/android/graphics/Bitmap.html

Último acceso noviembre 2015

http://developer.android.com/intl/es/reference/android/graphics/Color.html

Último acceso noviembre 2015

https://en.wikipedia.org/wiki/RGB_color_model

Último acceso noviembre 2015

http://androideity.com/2011/10/27/trabajando-con-recursos-en-android/

Último acceso noviembre 2015

http://es.ccm.net/faq/8470-analisis-de-los-sistemas-operativos-para-smartphones

Último acceso diciembre 2015

http://html-color-codes.info/codigos-de-colores-hexadecimales/

Último acceso diciembre 2015

http://elbauldelprogramador.com/programacion-android-interfaz-grafica/

Último acceso noviembre 2015

http://petrnohejl.github.io/Android-Cheatsheet-For-Graphic-Designers/

Último acceso noviembre 2015

http://daltonismogrupo19.blogspot.com.es/

Último acceso diciembre 2015

http://www.ey.com/ES/es/Industries/Life-Sciences/Informe-sobre-la-ceguera-en-Espana--Estado-y-costes-de-las-discapacidades-visuales

Último acceso diciembre 2015

http://www.ceapat.es/InterPresent1/groups/imserso/documents/binario/appsaccesibles.pdf

Último acceso diciembre 2015