Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería de Computadores

Trabajo Fin de Grado

Sistema inteligente de gestión y propuesta de vestuario

Autor: Marina Murillo Teruel

Tutora: Sira Elena Palazuelos Cagigas

2021

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Computadores

Trabajo Fin de Grado

Sistema inteligente de gestión y propuesta de vestuario

Autora: Marina Murillo Teruel

Tutora: Sira Elena Palazuelos Cagigas

Tribunal:

Presidente: Manuel Ocaña Miguel

Vocal 1º: Pablo Ramos Sainz

Vocal 2º: Sira Elena Palazuelos Cagigas

Fecha de depósito: 4 de octubre de 2021

"Supuestamente, el cerebro humano es algo parecido a una libreta que se adquiere en la papelería: Muy poco mecanismo y muchas hojas en blanco."

Alan Turing

Agradecimientos

Todos nuestros sacrificios son para este momento.

Linked horizon

Este trabajo ha sido bastante difícil de afrontar por el hecho de tener libertad total sobre él. Normalmente, los estudiantes tenemos a los profesores que nos guían indicándonos siempre en el transcurso de los trabajos lo que debemos o no debemos hacer o la plataforma donde se desarrolla. Siempre te dan algunas herramientas por dónde empezar, aunque sean desconocidas.

En este caso, fue algo diferente. Sólo había un campo inmenso donde estaba en mi mano construir lo que quisiese de la forma que quisiese con las herramientas que quisiese. Al ser una idea propia, le di mucha importancia y me permitió adquirir una motivación extra. Por ello hubo mucho prueba y error. El trabajo de dos meses se podía desechar fácilmente al ver que no se llegaba a nada productivo. Y en esos cambios y reintentos es donde más he aprendido, donde más he apreciado el esfuerzo de los demás y donde más me ha puesto a prueba.

Pero en todo este camino nunca he estado sola. Por ello, este proyecto se lo quiero dedicar a varias personas maravillosas:

A mis padres, por haberme enseñado desde pequeña este ámbito de la informática y haber disfrutado con ellos de este campo.

A mis profesores del colegio, por haber aprendido gracias a ellos la valiosísima lección de que no hay una asignatura que se pueda resistir.

A mi chico, por estar ahí incondicionalmente apoyándome por siempre.

A mis suegros, mi otra familia, por todo el apoyo moral constante recibido y entenderme en los malos momentos.

A mis compañeros y amigos de la universidad, sin sus apuntes y su apoyo no podría haber progresado tanto en esta carrera.

A mis profesores de la universidad, por haberme enseñado cuáles son mis aptitudes y hasta dónde puedo llegar.

Especial mención a Sira, mi tutora y la que más me inspiró desde el principio, viendo lo trabajadora y lo apasionada de su trabajo que es, además de apoyarme y creer en mi idea desde el principio.

Ha sido realmente duro y por momentos no creía que lo iba a conseguir, pero gracias a vuestras enseñanzas y ayuda, he podido completar esta etapa en mi vida.

Desde aquí, mil gracias.

Índice general

Índice de figuras

Índice de tablas

Índice de código fuente

Resumen

Palabras clave: Aplicacion, vestuario, inteligencia artificial, Android, Firebase.

Este proyecto describe los procesos de diseño y desarrollo de una aplicación para móviles Android para gestionar y proponer vestuario.

En primer lugar, el usuario realiza fotos de sus prendas, que se almacenarán en una base de datos con un conjunto de características rellenadas de forma automática a partir de inteligencia artificial (tipo de prenda: pantalón, chaqueta, etc.) y código (temporada, ocasión,..). El contenido de la base de datos se puede editar o eliminar si fuera necesario.

La aplicación propondrá conjuntos basados en la ropa almacenada, considerando el estilo seleccionado por el usuario, la temporada (fecha) y el color determinado por el usuario.

Abstract

This project describes the design and development processes of an Android mobile application for managing and suggesting clothing.

First of all, the user takes pictures of his clothes, which will be stored in a database with a set of characteristics automatically filled in from artificial intelligence (cloths: pants, jacket, etc.) and code (season, occasion,...). The content of the database can be edited or deleted if necessary.

The application will propose outfits based on the stored clothes, considering the style selected by the user, the season (date) and the color.

Resumen extendido

En este proyecto se ha diseñado y desarrollado una aplicación para móviles Android para gestionar y proponer vestuario.

Antes de implementarla se realizó un estudio de mercado y se llegó a la conclusión de que las aplicaciones similares que están disponibles no tienen soluciones para introducir las prendas de una forma rápida y cómoda. Tampoco proponen *outfits* de una forma automática, sino a través de profesionales con costes adicionales.

El objetivo final de la aplicación es proponer al usuario conjuntos a partir de su ropa, de forma fácil y sencilla, teniendo en cuenta la temporada (si es verano, invierno o entretiempo), una buena coordinación de los colores y que la ocasión sea adecuada (trabajo, deporte, etc.).

Para ello, inicialmente el usuario realiza fotos de sus prendas, que se almacenan en una base de datos de forma rápida y cómoda, con un conjunto de características rellenadas de forma automática a partir de inteligencia artificial (tipo de prenda: pantalón, chaqueta, etc.) y código (temporada, ocasión,..). De esta forma, no es tan tedioso rellenar las caracteristicas, y la información necesaria para hacer las propuestas. Además, las características de la prenda se pueden editar o eliminar en cualquier momento.

En el desarrollo de la aplicación se ha utilizado la herramienta Firebase, que tiene en su catálogo muchos servicios, entre ellos la base de datos no relacional y la inteligencia artificial.

Capítulo 1

Introducción

Durante eones, el ser humano ha demostrado su capacidad de anteponerse a las adversidades: la alimentación, el transporte... Todo ello necesitó un avance tecnológico que ayudase a que tareas como la caza y la recolección fuesen más sencillas y óptimas.

Siglos después, la época de la Revolución Industrial le recordó con fuerza a la especie esa capacidad de anteponerse a las adversidades y buscar soluciones.

Hoy en día, nuevas tecnologías abren nuevos caminos y nuevos retos buscan ser solucionados. En esta época de prisas, agilidad, de rápidas respuestas, se exige estar más atento, más alerta y más perceptivo al entorno y a cubrir satisfactoriamente la función básica de relación. Una de las carencias que contempló este proyecto y que ha buscado cubrir es la necesidad de ir bien vestido para el individuo y para los demás en una sociedad donde la primera impresión es clave para relacionarse.

Una persona promedio tiene muchas tareas diarias que, con el tiempo, se han ido cubriendo con electrodomésticos, pero existen otras tantas que no, la mayoría de ellas enfocadas a la salud personal. Esta memoria recoge un proyecto que busca dar respuesta a la pregunta que más quebraderos de cabeza puede dar: ¿qué me pongo hoy?

Por ello, en este trabajo, se ha desarrollado una aplicación de gestión y propuesta de vestuario donde se digitaliza el fondo de armario, rellenando de forma automática (haciendo uso de la inteligencia artificial y otros métodos) la mayoría de información y guardándola en una base de datos para que después se sugieran conjuntos correctamente combinados.

1.1 Estado del Arte

Para poder llevar a cabo este proyecto, primero se debe hacer un estudio de mercado para investigar y buscar aplicaciones, proyectos o trabajos que tienen que ver en mayor o menor medida con este TFG.

Se han encontrado una serie de aplicaciones que abarcan la idea o algún concepto parecido. Hay algunas que recomiendan (según algunos parámetros como la meteorología, ubicación, asistencia a eventos, etc.) la ropa más adecuada como Cloth, otras aplicaciones que ofrecen recomendaciones personalizadas de un profesional (iWarda), que muestran lo que está de moda como Stilycious o que te permiten organizar los atuendos que llevarás semanalmente (Qué Vestir), así como compartir y ver los *outfits* formados y los de los demás (21 Buttons y Combyne).

A continuación, se enumeran los productos junto con sus características diferenciadoras.

- **Cloth**[19]**.** Se comparten fotos de los *outfits*, recomienda atuendos según el tiempo que hace.
- **Wishi**[20]**.** Armario virtual con una comunidad para compartir el contenido personal.
- **Stylebook**[21]**.** Se puede planear lo que llevar entre semana.
- **Stylicious**[22][23]**.** Se planifican atuendos inspirados en los estilistas que se encuentran en la aplicación. También se observa lo que está en tendencia.
- **ASAP54**[24]**.** Con una foto recomienda artículos similares e indica en qué tiendas se puede comprar.
- **Combyne**[25]**.** Suben fotos de los *looks*, comentar en ellas e incluso permite comprar productos directamente desde la aplicación.
- **Qué vestir**[26]**.** Se planean los atuendos de cada día.
- **Smart Closet**[27]**.** Sugiere qué atuendos llevar puestos y tiene la opción de planear y compartir con amigos los modelos que se van a llevar puestos.
- **Dress It**[28]**.** Recomienda qué ropa llevar y en un futuro los desarrolladores se plantean implementar una red social.
- **21 Button**[29]**.** Red social donde se suben los atuendos llevados durante el día y se puede saber dónde se ha comprado.
- **iWarda**[30]**.** Ofrece múltiples servicios, entre ellos la digitalización del armario, espacio extra y asesoramiento profesional.

1.2 Conceptos

Seguidamente se enuncian una serie de conceptos para poder comprender todos los puntos que se tratan a lo largo del proyecto. Es necesario detallar algunas definiciones para dar un contexto adecuado y poder comprender la totalidad del proyecto.

1.2.1 Android

Android es un sistema operativo móvil basado en Linux junto con otros *softwares* de código abierto. Fue concebido para dispositivos con pantalla táctil como teléfonos móviles o *tablets*, llegando a automóviles y televisores.

Inicialmente fue desarrollado por Android Inc. y presentado en 2007 para promover el código abierto (*Open Source*) en teléfonos y ordenadores. Fue comprado en 2005 por Google.

En cuanto a las distintas versiones de Android, cabe mencionar que se denominan con nombres de dulces ordenados alfabéticamente hasta la versión 10. De esta manera, la primera versión de Android se llamó *Apple Pie* (tarta de manzana) y la última que tuvo esa codificación, *Pie* (tarta). Actualmente la última versión es la 12.

Figura 1.1: Logo Android[1].

Figura 1.2: Logo Apache[2].

1.2.1.1 Historia

Android fue desarrollado en 2003 por la empresa del mismo nombre, Android Inc., fundada[31] por Andry Rubin, Rich Miner, Nick Sears y Chris White. En un principio se anunció como un sistema operativo orientado a conectar cámaras digitales a ordenadores sin necesidad de cables.

Tras el fracaso inicial, se replanteó para dispositivos móviles y no fue hasta dos años después que Google se interesó por esta compañía comprándola por 50 millones e incorporando a los cuatro fundadores en plantilla.

Pasaron otros dos años y el 5 de noviembre de 2007 Google anuncia la primera versión de su sistema operativo para móviles, junto con la creación de un consorcio de 78 compañías que desarrollarían estándares abiertos para estos dispositivos tanto en *hardware* como en *software*. Esto derivó en que Google liberase la mayoría del código de Android bajo la licencia de Apache[32], una licencia de código abierto y libre. Dicha licencia consiste en la obtención del código y los derechos de autor una vez, es decir, no es necesaria la facilitación del código después de actualizarse.

A fecha de octubre de 2008 salió al mercado el primer dispositivo móvil con la primera versión oficial de Android[33].

Tres años después, Android consigue una cuota de mercado de más del 50 % en Estados Unidos superando a grandes marcas conocidas como Apple hasta ahora líder en el mercado gracias a su sistema operativo iOS.

Desde entonces y hasta ahora, el desarrollo de este sistema no ha parado de crecer con continuas actualizaciones, mejoras de rendimiento, seguridad, soporte de tecnologías y nuevas funciones. Ha conglomerado una gran comunidad en los últimos años que ha permitido extender las funcionalidades de los dispositivos. En 2018 ya se superaban los dos millones de aplicaciones en Google Play, la tienda oficial.

Figura 1.3: Fundadores de Android. De derecha a izquierda: Nick Sears, Chris White, Andy Rubin y Rich Miner.

Figura 1.4: Primer móvil con Android 1.0: T-Mobile G1[3].

1.2.1.2 Arquitectura

La arquitectura[34] que compone este sistema operativo es la siguiente.

- **Núcleo Linux.** Su núcleo le proporciona los procesos básicos del sistema como seguridad, gestión de memoria y muchas funciones más. Es la responsable de la capa de abstracción[1] entre *hardware* y *software*.
- **Almacenamiento.** Usa SQLite, una pequeña base de datos utilizada para almacenar datos locales.
- **Multitarea.** Las aplicaciones que no se ejecutan en primer plano reciben ciclos de reloj.
- **Conectividad.** Es capaz de soportar un gran número de tecnologías de conectividad. Entre ellas se pueden destacar Bluetooth, Wi-Fi, LTE, NFC y GPRS.
- **Mensajería.** Se usan varias formas de mensajería tanto en la nube, gracias al servicio de *Firebase Cloud Messaging* (FCM), como fuera de ella con SMS y MMS.
- **Videollamada.** Tiene soporte de videollamadas a través de *Hangouts*.
- **Navegador web.** Su navegador está basado en el motor de renderizado de código abierto WebKit, emparejado con el motor JavaScript de Google Chrome.
- **Características basadas en voz.** Es posible realizar búsquedas de voz en Google.
- **Soporte multimedia.** Android soporta más de 15 formatos multimedia.

[1] Elemento que se utiliza para ocultar la complejidad del *hardware* mediante *software*.

- **Soporte para *streaming*.** Realiza una descarga progresiva de HTML gracias a HTML5.
- **Soporte para *hardware* adicional.** Es capaz de soportar un gran número de tipos de sensores y *hardware* como cámaras adicionales, pantallas, giroscopios, etc.
- **Soporte de Java.** La mayoría de las aplicaciones están escritas en Java, pero el *bytecode* no es ejecutado directamente, sino que primero se compila en un ejecutable Dalvik[35] y se ejecuta en la máquina virtual (MV) Dalvik[2]. Cada aplicación corre su propio proceso, con su propia instancia de la MV.
- ***Runtime* de Android.** Incluye un set de bibliotecas base de C/C++ que proporcionan la mayor parte de las funciones disponibles en el sistema.
- **Entorno de desarrollo.** Los desarrolladores poseen acceso a las APIs que utilizan las aplicaciones base. De esta forma pueden reutilizar componentes de manera satisfactoria. Además, incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del *software*. Actualmente el entorno oficial de desarrollo es Android Studio.
- ***Tethering*.** Da una posibilidad de usar el dispositivo como un punto de acceso alámbrico o inalámbrico. De esta forma se puede utilizar para dar una red Wi-Fi a un dispositivo que no disponga de conexión a internet.
- **Multi-táctil.** Android tiene soporte nativo para pantallas capacitivas con soporte multitáctil. La funcionalidad fue originalmente desactivada a nivel de núcleo (posiblemente para evitar infringir patentes de otras compañías). Más tarde, Google lo habilitó de forma nativa.
- **Diseño de dispositivo.** Permite adaptarse a todo tipo de resoluciones desde menores a mucho mayores. También soporta gráficos 2D y 3D.

1.2.1.3 Ventajas y desventajas

Como ya se ha mencionado y a diferencia de algunos de sus competidores, Android es de código abierto y cualquier aparato de cualquier compañía es capaz de funcionar con este sistema operativo, adaptando el código a sus dispositivos. Esto da lugar a que haya más diversidad y que el sistema operativo crezca y se mejore más rápidamente.

Por el contrario, una de las grandes desventajas para la comunidad de Android, es en lo referente a las actualizaciones del sistema. Normalmente, las actualizaciones de los sistemas son automáticas y están disponibles para todos los dispositivos a la vez, pero en Android no puede realizarse así, ya que, debido a la diversidad de compañías que lo utilizan, cada una de ellas planea sus actualizaciones. De esta forma, cabe la posibilidad de que algunos dispositivos queden obsoletos por no actualizarse más. Algo positivo para los usuarios es que se pueden utilizar actualizaciones no oficiales para alargar la vida de los dispositivos.

Otra ventaja más de este sistema operativo es la facilidad para desarrollar e instalar cualquier tipo de aplicación en los diversos aparatos que tienen compatibilidad, ya que no hace falta que sean descargadas de la tienda oficial, pueden ser aplicaciones de terceros o ajenas a Google. Por tanto, le da muchas facilidades al desarrollador, a costa de que el sistema no es tan seguro debido sus vulnerabilidades a los *malware* o virus. Es posible llegar a robar la información que se encuentre en el dispositivo infectando al dispositivo camuflado como actualización o incluso con una aplicación de la tienda no oficial de Google.

[2]Máquina virtual optimizada para Android y móviles con batería, memoria y procesador limitados.

1.2.2 Inteligencia Artificial

La inteligencia artificial (IA)[36] es la ciencia que se encarga de crear programas que procesan información como el aprendizaje o el razonamiento lógico. Entre los objetivos de esta rama de investigación, se busca emular la mente humana basándose en la forma que tiene esta raza de pensar mediante algoritmos que imitan estos razonamientos. Hay una gran cantidad de subgéneros que engloban esta gran materia como el *deep learning* (aprendizaje profundo) contenido a su vez en el *machine learning* (aprendizaje automático).

1.2.2.1 Historia

Pero esta tecnología no es tan novedosa como parece[37], pues entendidos en la materia, se ponen de acuerdo en señalar su origen en la antigua Grecia con Aristóteles[38]. Este filósofo hace distinción entre materia y forma, lo que hoy en día se ha traducido como computación simbólica y abstracción de datos. También aportó la lógica y la idea de que el estudio del pensamiento por sí mismo aporta la base de cualquier conocimiento. Se encargó de describir las reglas que sigue la mente para llevar al ser humano a conclusiones racionales.

Figura 1.5: Alan Turing[4].

No sería hasta siglos después, que Alan Turing[39], definido como el padre de las ciencias de la computación, ayudó a comprender los ordenadores. Se encargó de desarrollar la máquina encargada de descifrar los mensajes codificados que se enviaban los alemanes en la Segunda Guerra Mundial. Pero además de eso, teorizaba sobre 'máquinas' que podían funcionar solas. En 1936, ayudó a sentar las bases sobre el futuro de la ciencia computacional y la inteligencia artificial al publicar los «Números Calculables»[40]. Aquí nace el concepto de la 'Máquina de Turing'[3] (figura 1.6) con el primer algoritmo y resultó ser la precursora de las computadoras digitales.

En la década de los 50 la inteligencia artificial creció exponencialmente llegando a tener máquinas que podían resolver problemas algebraicos, jugar o entender un idioma. En 1950 Turing volvió a publicar un artículo. Esta vez se titulaba *"Computing Machinery and Intelligence"* y defendía la posibilidad de emular el pensamiento humano a través de la computación además de enunciar el Test de Turing.

Más adelante, en 1956, los científicos John McCarthy, Marvin Minsky, Claude Shannon, Ray Solomonoff, Alan Newell, Herbert Simon, Arthur Samuel, Oliver Selfridge, Nathaniel Rochester y Trenchard

[3]Dispositivo que manipula símbolos mediante unas reglas en una cinta.

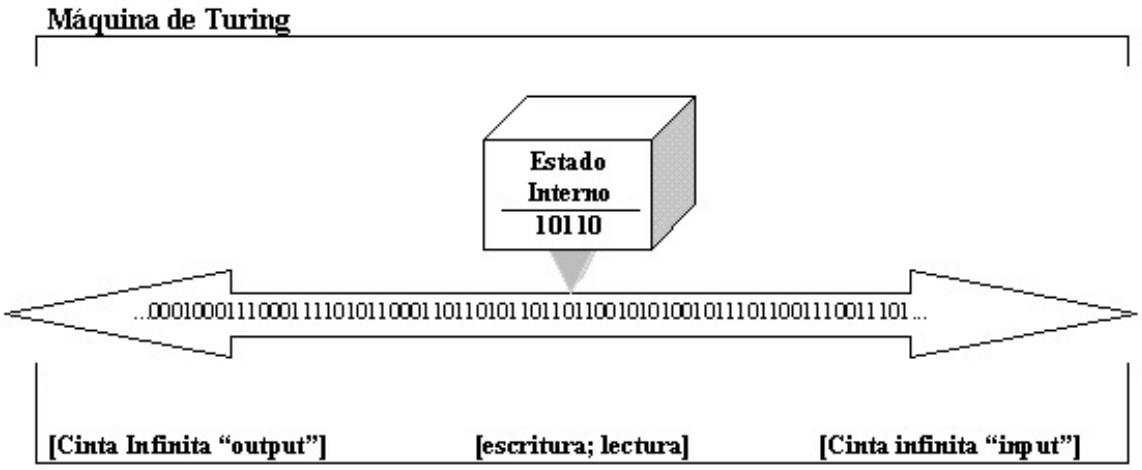

Figura 1.6: Máquina de Turing[5].

More se reunieron en la conferencia de *"Dartmouth Summer Research Project on Artificial Intelligence"*[41] (figura 1.8) donde se trataban temas del pasado y del futuro. La conferencia fue organizada por John McCarthy, que por aquel entonces era profesor de matemáticas en la universidad de Dartmouth, "partiendo de la base de la conjetura de que todos los aspectos del aprendizaje o cualquier otra característica de la inteligencia pueden, en principio, describirse con tanta precisión que se puede hacer una máquina para simularlos"[4][42]. En esa conferencia se acuñó por primera vez en la historia el término inteligencia artificial (IA)[43]. Hasta ahora, todos los artículos correspondientes a esta área se titulaban de diferentes maneras siempre refiriéndose a autómatas inteligentes.

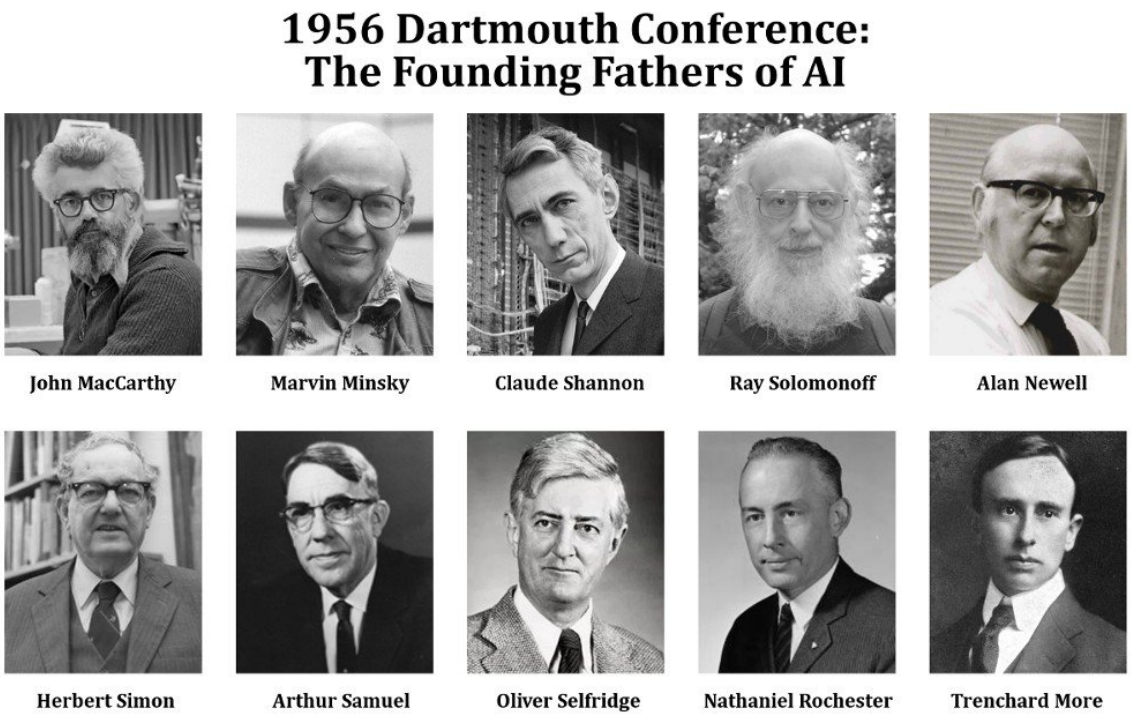

Figura 1.7: Participantes en la conferencia de Dartmouth[6].

[4] John McCarthy, 1956 - *"To proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it."*

Además, en este mismo año nacería el perceptrón[44] diseñado por Frank Rosenblatt, la unidad básica de las redes neuronales de hoy en día. Un perceptrón toma una o varias entradas binarias y produce una sola salida también binaria. Para calcularla, Rosenblatt presenta el concepto de "pesos", un número real que expresa la importancia de la respectiva entrada con la salida.

En 1966, Joseph Weizenbaum del MIT creó un programa con el que pretendía ser capaz de engañar a los humanos haciéndoles pensar que estaban hablando con otra persona. Se llamaba ELIZA[45]. Este primer *chatbot*[46] fue diseñado como un método para mostrar la superficialidad de la comunicación entre el hombre y la máquina. Así, la IA sólo reconocía palabras clave y preguntaba sobre ellas. De esta manera el sujeto de pruebas podía pensar que había una persona detrás del ordenador que entendía sus preocupaciones y problemas.

Posteriormente, a finales de la década de 1960, Terry Winograd, en ese entonces estudiante en el *Massachusetts Institute of Technology* (MIT), desarrolló el sistema SHRDLU[47], al que posteriormente, se añadió un entorno 3D en la Universidad de UTAH. Era un programa[48] en el que se debía describir cómo manipular y cambiar el entorno con frases simples en inglés. Lo destacable es que, como el mundo se podía describir de forma muy precisa, era posible una comprensión computacional del lenguaje del ser humano, con un conjunto reducido de palabras en inglés.

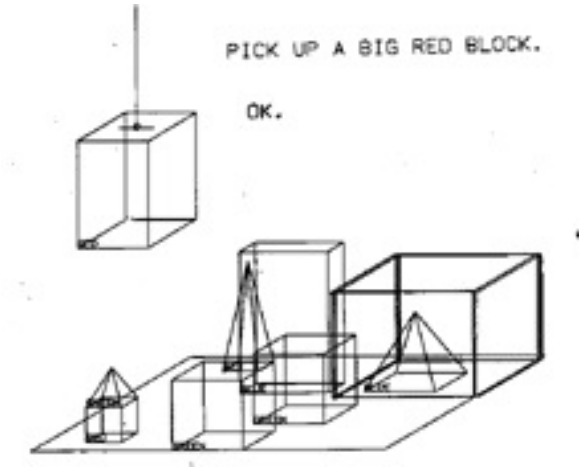

Figura 1.8: Espacio tridimensional de SHRDLU.

Esta herramienta, contaba con una memoria lo suficientemente potente como para comprender el contexto de la conversación. Si se hacía referencia a un objeto describiéndolo de forma precisa y a continuación se volvía a hacer referencia al mismo sin describirlo del todo, el programa sabía a qué objeto del entorno se estaba refiriendo. También se podía preguntar sobre las estadísticas del mundo o el historial de acciones.

En 1996, IBM creó una IA llamada Deep Blue. Este ordenador se enfrentó al entonces campeón mundial de ajedrez Garri Kaspárov. Al principio no tuvo éxito, pero la versión mejorada un año después fue capaz de lograr la victoria frente al ajedrecista ruso.

Ya en el nuevo siglo y contando con importantes avances tecnológicos, la multinacional IBM desarrolló una supercomputadora llamada Watson, que ganó en tres oportunidades a dos de sus máximos campeones en el juego de Jeorpardy, un concurso televisivo de cultura general.

El ingeniero Ray Kurzweil afirma que, en el año 2045, la inteligencia artificial supera a la humana. Los ordenadores serán capaces de razonar mejor que un humano. Sin embargo, hay detractores que piensan que es bastante difícil que en tan corto período de tiempo las máquinas sean capaces de pensar por sí mismas.

1.2.2.1.1 Test de Turing

Alan Turing planteó el Test de Turing[49] en su ensayo *"Computing Machinery and Intelligence"* de 1950 sentando las bases que permiten discernir entre una máquina y una persona.

Esta prueba es un examen de la capacidad que posee una máquina, no para entender lo que se le está comunicando, sino para intentar imitar las respuestas que daría un ser humano. Entonces, una persona mantiene una conversación escrita en lenguaje natural evaluando a un humano por una parte y una máquina por otra durante 5 minutos. Si el evaluador no puede discernir entre ambos, esta habría pasado el test.

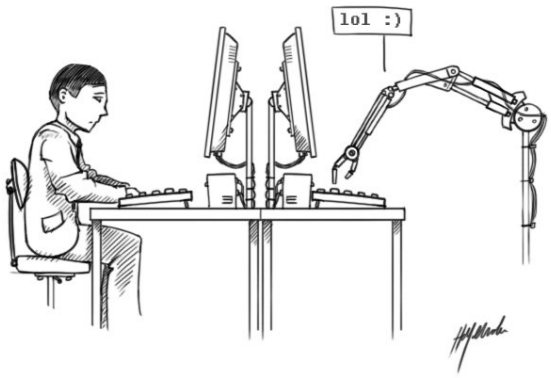

Figura 1.9: Máquina utilizando una expresión del ser humano[7].

Desde que fue creada, la prueba ha demostrado ser altamente influyente, además de transformarse en un concepto importante en la filosofía de la inteligencia artificial.

1.2.2.2 Disciplinas

La inteligencia artificial en los últimos años ha ido creciendo y ha formado un gran grupo que ha acogido a muchas de las tecnologías que actualmente son esenciales para el día a día. Esta gran masa de algoritmos se puede dividir en varias ramas[50][51] (figura 1.2.2.2). A continuación, se describen las más populares.

- **Aprendizaje automático:** Se basa en el entrenamiento de algoritmos a través de datos que se le facilitan donde se especifica lo que se desea clasificar. Es la rama utilizada en este proyecto y una de las más populares. Se utiliza en los *deepfakes*[5] y coches automáticos.
- **Procesamiento de lenguaje natural:** En esta rama se busca mejorar la capacidad de las máquinas para entender palabras escritas u oídas en lenguaje humano, a diferencia de una computadora la cual entiende lenguajes de programación. Esta tecnología es utilizada en *chatbots*, asistentes virtuales digitales y en motores de búsqueda para filtrar spam.
- **Sistemas expertos:** Se encarga de resolver problemas como si fuera un humano con conocimiento recopilado. Éstos imitan el proceso de razonamiento que los expertos utilizan para resolver problemas específicos. Ejemplos de aplicaciones en este campo son la detección de estructuras químicas moleculares a partir de su análisis espectro gráfico (DENDRAL) o la determinación de diagnósticos en enfermedades infecciosas.

[5] Generación de caras donde se superponen los rasgos faciales de una persona con otro rostro.

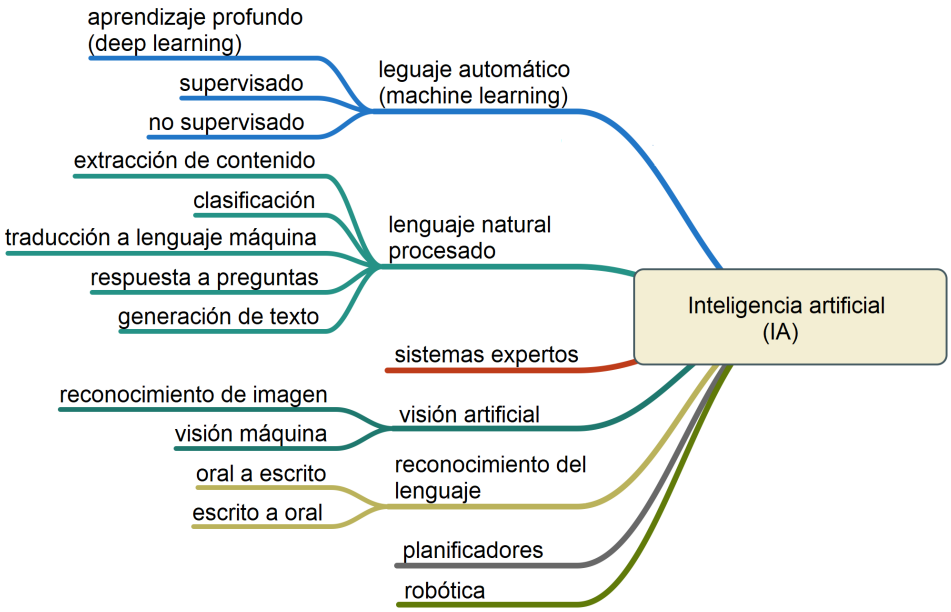

Figura 1.10: Algunas de las ramas de la inteligencia artificial[8].

- **Visión artificial:** Está compuesto por tecnologías para reconocer información en imágenes, formas, caras, colores, etc. Las aplicaciones de la visión artificial[52] son tan diversas como la detección de intrusos, el mantenimiento de la seguridad vial, el control de los procesos en automatización industrial o la conducción automática. Se utiliza en dispositivos como cámaras inteligentes o sensores de visión para detectar anomalías o reconocer objetos.

- **Reconocimiento de la voz:** Parte de las reglas gramaticales de la lengua y analiza el habla en directo para entender al interlocutor y actuar en consecuencia. Los tonos de voz o el humor pueden alterar el significado de una frase.

- **Planificación automática:** Trata sobre la creación de planes de un robot para que sea autosuficiente. Los programas que incorporan estos algoritmos se denominan planificadores.

1.2.2.3 Ventajas y desventajas

Esta tecnología tiene muchos beneficios[53] si se utiliza de forma correcta, ya que es posible procesar grandes cantidades de información en muy poco tiempo y puede ser una pieza clave a la hora de clasificar o discernir determinadas clases de objetos, así como actuar por sí solo como un autómata, de forma que tome las decisiones que más favorezcan su labor. Por ejemplo:

- **Automatización de procesos.** Los robots se pueden encargar de las tareas repetitivas y rutinarias sin intervención de los humanos haciéndolas rápidamente y sin fallos.

- **Potencia las tareas creativas.** Liberando a los humanos de labores repetitivas, se permite que se puedan dedicar a cometidos donde desarrollen su creatividad y dejar a los robots que se encarguen de las tareas que superan la limitación humana, como la minería.

- **Aumenta la precisión.** Las máquinas son capaces de superar la agilidad mental humana con respecto a las decisiones y las acciones tomadas. Optimizan los resultados tanto a nivel de producción como de negocio.

- **Minimiza el error humano.** Reduce los fallos provocados por las limitaciones del ser humano. En algunas empresas se instalan cámaras con IA para asistir en la detección de posibles anomalías en los productos que producen. De esta forma se detectan fisuras o defectos imperceptibles para el ojo humano.
- **Velocidad de respuesta.** Con la información que tiene almacenada de la producción de cierta cadena industrial, puede analizarla y llevar a cabo acciones en tiempo real.
- **Mantenimiento predictivo.** Si se realiza un mantenimiento del equipamiento industrial basado en los tiempos y condiciones de funcionamiento de los mismos, permite incrementar su rendimiento y ciclo de vida.
- **Compatibilidad de sectores.** Se puede aplicar a una gran cantidad de sectores, desde el sector sanitario hasta la aviación.

Sin embargo, todavía es una tecnología relativamente nueva y se puede apreciar en la fase de aprendizaje, donde probablemente cometerá algunos fallos[54] cuando haga el trabajo asignado. Y en un futuro, cuando esté mucho más desarrollada, socialmente derivará algunos problemas éticos tales como la extinción de algunos trabajos. A continuación, se enumeran algunos puntos negativos:

- **Disponibilidad de datos.** Debido a que en la mayoría de las bases de datos puede faltar parte de la información, es necesario invertir más recursos para completar estos vacíos y que no deriven errores.
- **Falta de profesionales cualificados.** Para poder implementar estas tecnologías se necesita a gente especializada en esta área con años de formación y aunque está aumentando gradualmente todavía hay falta de profesionales en esta área.
- **Coste y tiempo elevados.** Los proyectos de inteligencia artificial requieren de mucho tiempo y dinero. Primero, para diseñar un modelo que haga las tareas indicadas de forma correcta y segundo, para poder adaptarse al espacio. También se incluyen los costes económicos para el mantenimiento.

En definitiva, la IA es la mayor aliada de las empresas ya que les permite ser mucho más competitivas y obtener mayores beneficios, sobre todo en entornos de fabricación y producción. Por ello, cada vez hay una mayor demanda de estos perfiles en el sector industrial.

1.2.2.4 Aplicaciones

Hoy en día esta tecnología está en auge por muchas razones[55]. Entre ellas destaca la facilidad de las grandes empresas para poder recopilar gigantescas masas de información por la cantidad de personas que hacen uso de estas tecnologías, el avance en el mercado de componentes electrónicos como gráficas o microprocesadores cada día más potentes (capaces de hacer tareas en las que antes se necesitaba mucho tiempo en el menor tiempo posible), los avances en los algoritmos y modelos en los que se basa, etc.

Por ello, muchas empresas y particulares alrededor del mundo hacen uso de esta tecnología revolucionando muchos ámbitos, desde el empresarial al social con aplicaciones en medicina, como la detección de cáncer en el cuerpo, en la agricultura, luchando contra la deforestación, en el sector servicios con controles de calidad, vigilancia y reconocer los rostros de las diferentes personas, los asistentes virtuales, los robots y muchas más.

1.2.3 Bases de datos

Una base de datos[56] es una herramienta capaz de almacenar datos, organizarlos y relacionarlos para poder hacer búsquedas sobre determinados subconjuntos. De esta forma, es accesible de manera remota por una gran cantidad de usuarios para consultar la información de forma rápida y precisa mediante un lenguaje.

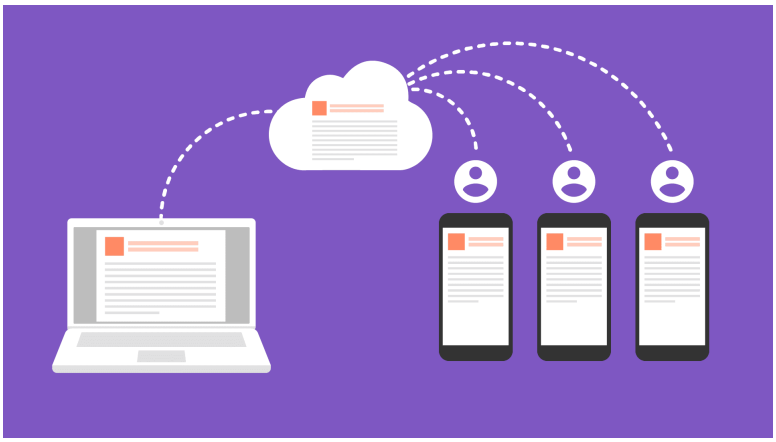

Figura 1.11: Base de datos que interconecta diferentes tipos de dispositivos[9].

1.2.3.1 Tipos de bases de datos

Hay varias formas de construir la estructura lógica de la base de datos[57]. Su sistema de gestión (*Database management system* o DBMS) estandariza las reglas y métodos para crear una base de datos con un modelo determinado.

Para saber el tipo[58] que necesita un proyecto, primero hay que valorar las ventajas y desventajas de cada una de las opciones. A pesar de que hay un tipo que los usuarios suelen recurrir sobre todas ellas, cada una se adapta mejor según el entorno en el que se encuentre.

1.2.3.1.1 Relacional

El modelo de bases de datos relacional es el más utilizado y conocido dada su sencillez en la gestión y en el mantenimiento de la información. Se organiza en tablas de filas y columnas con sus claves primarias (registros únicos, es decir, una columna con datos que no se repiten) y secundarias (campos que hacen referencia a otras tablas). En este modelo no suele haber información duplicada porque hay datos que se relacionan y se referencian entre sí.

Para poder crear una base de datos relacional, primero se deben estudiar las reglas que definen la integridad de las variables para garantizar que la base de datos es escalable, rápida y precisa.

Las más conocidas son Microsoft SQL Server y MySQL.

El lenguaje que se usa para conectarse con la base de datos y realizar las diversas operaciones es SQL (Lenguaje de Consulta Estructurado o *Structured Query Language*). Las sentencias SQL se realizan mediante *queries*: consultas con numerosas funcionalidades para añadir, editar, obtener, analizar o eliminar información. De esta manera, con sentencias complejas para asociar y referenciar datos entre sí, no se necesitan campos duplicados para consultar una información almacenada.

1.2.3.1.2 No relacional

Las bases de datos no relacionales (NoSQL o *Not only SQL*)[59] surgen como respuesta a limitaciones de las anteriores con modelos relacionales. Éstas no siguen el esquema tradicional de filas y columnas, sino que la información se almacena en una estructura de tipo JSON[6].

Las consultas en una base de datos NoSQL son mucho más rápidas. No hay que establecer relaciones entre los datos ya que todos los que se necesitan se ponen en una misma estructura.

Las más populares son MongoDB y Cassandra. El lenguaje que se utiliza para comunicarse con éstas varía según el tipo que se use.

1.2.3.1.3 Orientado a objetos

El concepto del diseño de este tipo de bases de datos proviene de la programación orientada a objetos (POO) y el concepto de las relacionales formando una unión bastante sólida.

Este modelo utiliza los objetos como entidades con un identificador único, teniendo propiedades y métodos. También se incluye el concepto de clase como colección de objetos con las mismas características.

Una ventaja de este modelo es que, al establecer objetos como representaciones de entidades con sus correspondientes propiedades, se asemeja a la realidad y facilita en gran medida su diseño y comprensión.

1.2.3.1.4 Orientado a grafos

En el diseño de grafos, la importancia de las relaciones es equiparable a la de los propios datos almacenados. Este tipo de bases de datos utiliza nodos como entidades o atributos. Cada uno de los nodos o vértices del modelo se comunica a través de aristas, relacionándose entre sí.

Se ha ido aumentando la utilización de este tipo de bases de datos en los últimos años, a medida que se necesitaban otras formas de expresar la información que se desea serializar. De esta manera, hay más variedad de formatos y cada una de ellas se adapta a unos modelos determinados de negocio.

Esta base de datos se utiliza sobre todo en redes sociales, ya que es de gran importancia recopilar información sobre las interacciones de cada usuario para con otros usuarios. Así, se pueden recomendar amistades, por ejemplo.

1.2.3.2 Diferencias entre SQL y NoSQL

Una vez se han tratado todos los tipos de bases de datos, se analizan las diferencias[60] entre el modelo de datos SQL y el NoSQL. Este último es con el que se ha trabajado a lo largo de este proyecto.

La principal diferencia entre la base de datos SQL y la NoSQL es la forma en la que almacenan la información. La primera la almacena en tablas y la segunda en JSON, por tanto, ya se manejan y acceden de forma completamente distintas. Además, hay que tener en cuenta que, de esta forma, en las tablas siempre se tiene la misma información, y en cambio, JSON da libertad y flexibilidad a la hora de almacenar elementos, porque no hace falta que todos tengan el mismo tipo de variables.

Otra diferencia importante es el lenguaje utilizado. En las bases de datos relacionales se utiliza el lenguaje SQL y en cambio, en las no relacionales, se evita usar ese lenguaje, anteponiendo su propio

[6]Tipo de archivo con una estructura de variables organizada.

lenguaje o uno alternativo en cada uno de los tipos de bases de datos. Por ejemplo, Cassandra utiliza el lenguaje CQL (*Cassandra Query Languaje*), MongoDB utiliza JSON y BigTable hace uso de GQL (*Graph Query Language*).

También se diferencian en que las NoSQL se pueden ejecutar en máquinas con pocos recursos. A diferencia de los otros sistemas, no requieren de muchos recursos de computación, por lo que es posible montarlas en dispositivos con un coste más reducido.

Las bases de datos no relacionales tienen escalabilidad horizontal. Esto es, que su estructura es distribuida y la información está en varios servidores a diferencia de las relacionales, que están centralizadas en un único servidor, lo que da lugar a la facilidad a la hora de manejar estas bases de datos con mayor cantidad de información.

Un problema de las bases de datos SQL es que tienen un punto de entrada común y eso deriva en problemas como cuellos de botella. Esto es porque cada vez que se realiza una acción, se transcribe y en algunas ocasiones, al haber sentencias realmente complejas, puede demorarse bastante tiempo. Por el contrario, las NoSQL no generan cuellos de botella tanto por su estructura distribuida como su procesamiento bajo.

Por último, las no relacionales no soportan las operaciones JOIN[7] a diferencia de las relacionales. Esto es debido a los grandes volúmenes de datos que almacenan estas y si lo que se busca no es un índice de alguna tabla, el sistema puede llegar a demorarse mucho tiempo.

1.2.4 Color

1.2.4.1 Teoría del color

En el arte de la pintura, el diseño, la fotografía, la imprenta y la televisión, la teoría del color[61][62] se denomina al conjunto de reglas básicas de la mezcla de colores para dar lugar al efecto deseado ya sea con luces o con pigmentos. Si se toma la luz, la mezcla de todos los colores da como resultado el color blanco, en cambio, si se comprenden los pigmentos de los utensilios para pintar, la mezcla de todos los colores da como resultado el negro.

1.2.4.2 Percepción biológica

La forma en la que se perciben los colores en la raza humana es a través de la retina del ojo, que es sensitiva a las longitudes de onda procedentes de nuestro entorno. La retina está compuesta por células fotorreceptoras con grandes cantidades de conos (65 % sensibles al rojo, 33 % sensibles al verde y 2 % sensibles al azul) y de bastones que recogen una pequeña parte del espectro de luz al que se le denomina espectro de luz visible. Esta información recogida viaja al cerebro en forma de impulsos eléctricos gracias al efecto fotoeléctrico y éste la decodifica.

Si el sistema tiene algunos conos que no funcionan bien, se producen irregularidades en la percepción del color. A esta anomalía se la conoce como daltonismo[63] y depende del modo, hay tres tipos.

- **Daltonismo acromático.** Es el más raro que existe. En este modelo el individuo ve en blanco y negro sin distinguir ningún color, algo que se debe a problemas de tipo neurológicos o fisiológicos en el ojo. Es una enfermedad congénita.
- **Daltonismo monocromático.** En este tipo, de los tres conos que existen en la retina, sólo uno de ellos es sensitivo, por tanto, los que lo sufren ven el mismo color en distintas intensidades.

[7] Sentencia que sirve para formar una tabla temporal para consultar datos provenientes de varias tablas a la vez.

- **Daltonismo dicromático.** Es completamente hereditaria y es uno de los casos más comunes. Los hombres tienen más probabilidades de tener este tipo de problemas. Hay tres tipos (figura 1.12).

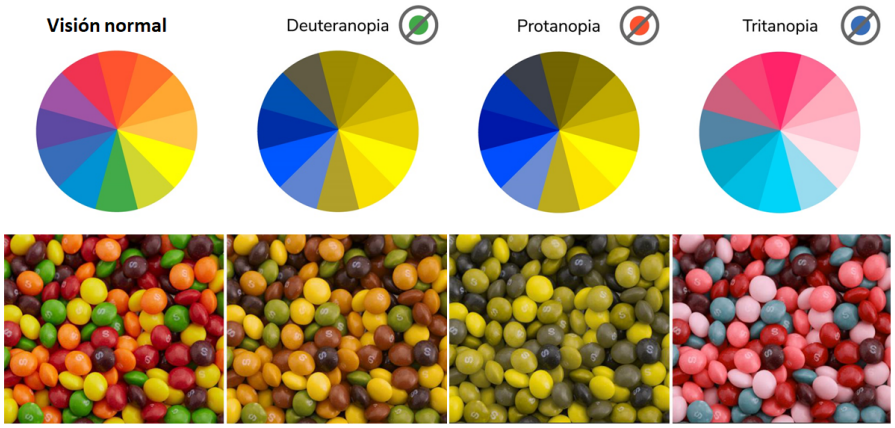

Figura 1.12: Tipos de daltonismo dicromático[10].

  - Deuteranopia. Ausencia de actividad de conos verdes.
  - Protanopia. Carencia de actividad funcional en los conos sensibles al color rojo.
  - Tritanopia. Pérdida de actividad de los conos sensibles al azul.

Es importante contemplar esta información sobre la percepción humana puesto que este trabajo tiene aplicaciones sobre las personas que padecen estas irregularidades tales como apoyarles en la decisión de conjuntar colores.

1.2.4.3 Codificación digital

Para poder digitalizar los colores se necesitan los espacios de colores[64]. Un espacio de color define un modelo de composición del color que no tiene por qué estar completo (pueden formar subconjuntos). Estos modelos no necesariamente deben tener 3 dimensiones como el RGB (*Red*, *Green*, *Blue*), sino que comprenden de una a cuatro dimensiones.

- Una dimensión: escala de grises (figura 1.13), escala Jet, etc.
- Dos dimensiones: suelen ser subespacios de modelos de tres dimensiones como el subespacio rg, xy, etc.
- Tres dimensiones: RGB (figura 1.14), HSV (figura 1.15), YI'Q', etc.
- Cuatro dimensiones: CMYK (figura 1.16).

Los espacios de color de tres dimensiones son los más extendidos. A continuación, se explican algunos de los espacios enunciados anteriormente.

1.2.4.3.1 Escala de grises

En este espacio cada color se corresponde a un valor de intensidad de gris. De esta forma, esta comprendido entre 0 y 1 en formato coma flotante o de 0 a 255 siendo un entero, aunque se puede llegar a cuantificar y definir los valores que lo representan. Estas imágenes están compuestas de sombras de grises.

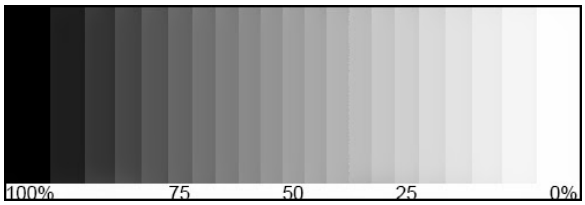

Figura 1.13: Modelo escala de grises[11].

1.2.4.3.2 RGB

Es conocido como un espacio de color aditivo respecto a los colores básicos rojo, verde y azul que ve el ojo humano con sus conos sensitivos. Si dos frecuencias de luz se unen, se suman y forman otro color que es una mezcla de ambos. De esta forma se pueden replicar muchos de los colores que ve el ser humano.

También se suele añadir un canal alfa (transparente) para añadir un efecto de opacidad al color y se denomina RGBA.

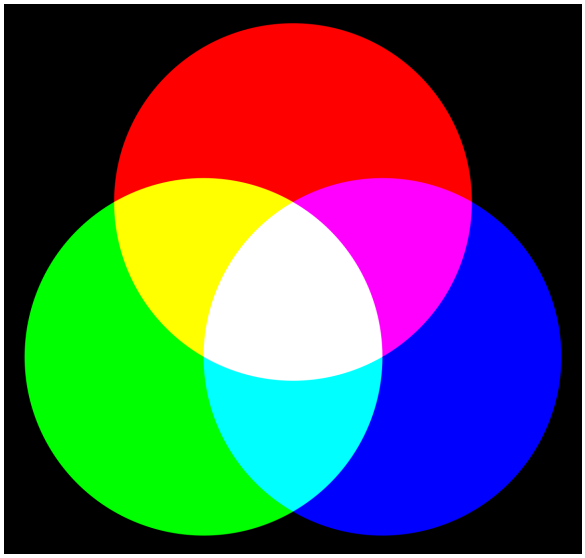

Figura 1.14: Modelo RGB[12].

1.2.4.3.3 HSV

Este modelo se representa en forma de cilindro, un cono o un cono hexagonal. Cada una de las letras se corresponde a una definición.

- Matiz (*Hue*): Establece la frecuencia dominante del color dentro del espectro visible. Es la percepción de un tipo de color en un espacio de más colores, es decir, la sensación humana de que un área parece similar a otra. Se decrementa su valor mientras se describe un movimiento de forma horaria en el cono, con el rojo en el ángulo 0.
- Saturación (*Saturation*): Es la cantidad o la pureza del color. Se puede asimilar como la mezcla con gris o blanco.
- Valor (*Value*): Denomina la intensidad de luz. Por consiguiente, es la cantidad de blanco o negro que posee un color.

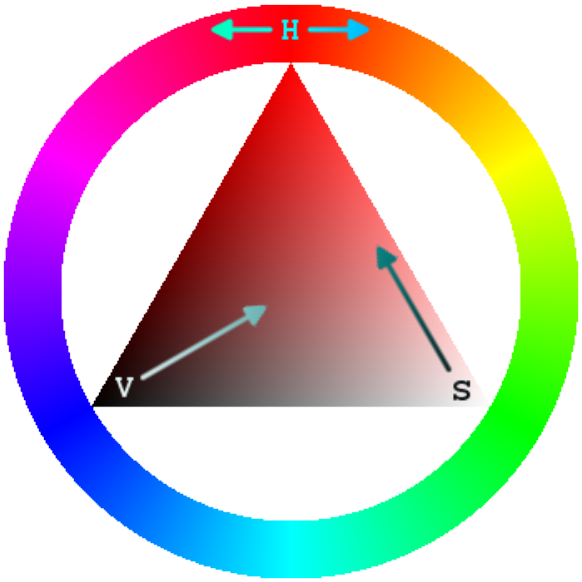

Figura 1.15: Modelo HSV[13].

1.2.4.3.4 CMYK o CMYB

CMY trabaja mediante la absorción de la luz de colores secundarios, es decir, describe un espacio sustractivo. Los colores que define son la cantidad de luz que no ha sido absorbida.

Este modelo plantea un problema y es que, al definir el negro, no llega a ser tan intenso como el definido por otros modelos, por ello se le añade el canal K (clave o *key*), que es el canal B (negro o *black*).

Este espacio es utilizado en las impresoras para poder dar más contraste al negro. De esta forma, se debe tener especial cuidado, ya que el color que se muestra en una pantalla de ordenador (codificado en RGB), no es el mismo que el impreso (codificado en CMY).

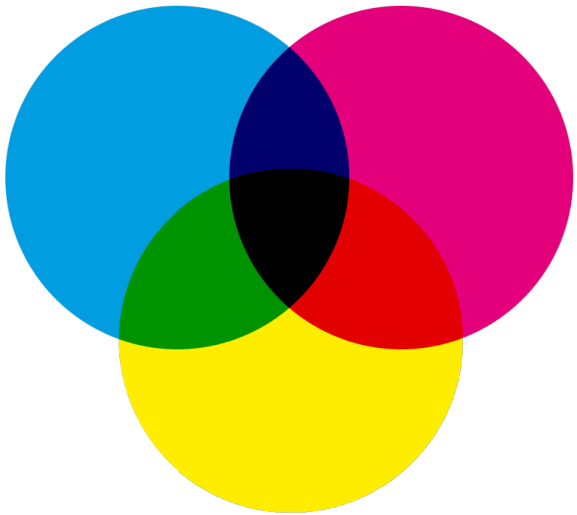

Figura 1.16: Modelo CMYK o CMYB[14].

Capítulo 2

Desarrollo

En este apartado se aborda el diseño de la aplicación y se describen los aspectos más importantes de la misma. Se detalla la utilidad y la implementación de cada uno de los bloques del proyecto, así como las relaciones entre ellos.

2.1 Diagrama de bloques

En la figura 2.1, se representan los bloques principales de la aplicación y, mediante flechas, las relaciones entre ellos.

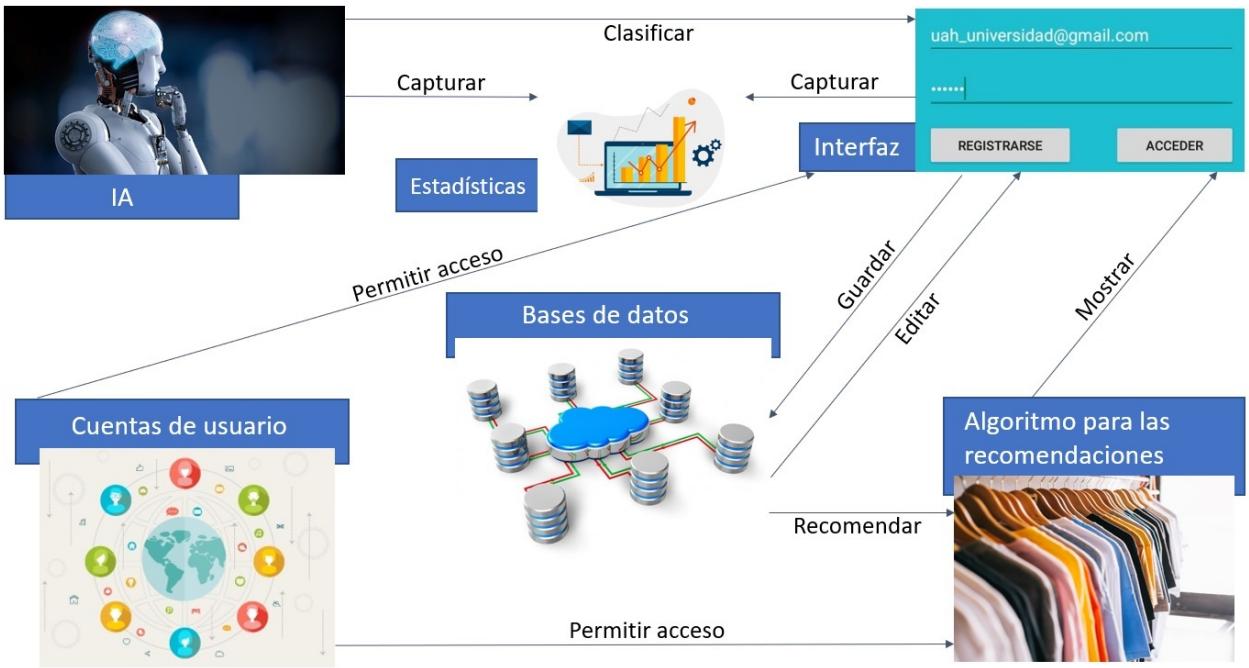

Figura 2.1: Diagrama de bloques.

Como se muestra en la figura, la aplicación se divide en seis grandes bloques:

- La interfaz, que conecta al usuario con los distintos bloques que dan funcionalidad al sistema: permite la visualización de prendas y propuestas, edición de características, etc.

- Las bases de datos que almacenan las fotos y todas las características de las prendas que se tienen en cuenta a la hora de realizar las propuestas de *outfits* posteriormente. Los datos que almacenan se pueden visualizar, editar y borrar.
- La inteligencia artificial para obtener automáticamente el tipo de prenda a partir de las imágenes.
- El algoritmo de recomendaciones que genera los *outfits* que propone al usuario a partir de las características de las prendas y del estilo, temporada, color, etc. seleccionados por el usuario. Se muestran todos los atuendos que se adecúen a los parámetros de búsqueda para que pueda elegir entre un abanico de opciones.
- El bloque de estadísticas que contabiliza los distintos eventos que ocurren en la aplicación para su análisis posterior.
- El bloque de gestión de cuentas de usuario: para registrar a los usuarios y permitir o denegar su acceso.

2.2 Bloque de interfaz de usuario

Es la vía de comunicación entre la funcionalidad de la aplicación y el usuario: se comunica con las bases de datos cuando se introducen nuevas prendas, con el algoritmo que realiza sugerencias para plasmar todos los resultados de forma agradable y con el bloque de gestión de cuentas a la hora de visualizar los datos del usuario actual.

En este apartado se van a explicar los elementos utilizados en la interfaz de usuario así como la parte de las características de la prenda, ya sea para la inserción de la misma o en la edición posterior.

Siempre que se mencione código en formato XML se refiere la parte gráfica de la aplicación.

2.2.1 Pantallas

Android Studio tiene diversidad de tipos de elementos definidos para formar aplicaciones. Desde módulos para poder escribir algo, hasta botones y barras de valoración con estrellas. Generalmente, cada uno de ellos se define con un id de variable, sus dimensiones y su posición.

La pantalla que más elementos tiene es la de *activity_add*, donde se guarda la ropa. Las que más elementos externos poseen, son la de *activity_mix*, que es la pantalla de sugerencia de atuendos (más adelante explicado en el apartado 2.2.6) y la de *activity_rate*, correspondiente a la pantalla para valorar la aplicación (apartado 2.2.4).

A continuación se enuncian los elementos más usados.

- ***EditText***: para introducir texto (código 2.1).

Código 2.1: activity_add.xml. *EditText.*

```
47              <EditText
48                  android:id="@+id/nameEditText"
49                  android:layout_width="match_parent"
50                  android:layout_height="45dp"
51                  android:hint="Prenda"
52                  app:layout_constraintStart_toStartOf="parent"
53                  app:layout_constraintTop_toTopOf="parent" />
```

- ***ImageView***: para visualización de imágenes (código 2.2).

Código 2.2: activity_add.xml. *ImageView*.

```
36              <ImageView
37                  android:id="@+id/background2"
38                  android:layout_width="475dp"
39                  android:layout_height="332dp"
40                  android:src="@mipmap/fondo"
41                  app:layout_constraintBottom_toBottomOf="@+id/linearLayout5"
42                  app:layout_constraintEnd_toEndOf="parent"
43                  app:layout_constraintStart_toStartOf="parent"
44                  app:layout_constraintTop_toTopOf="@+id/nameEditText"
45                  tools:srcCompat="@tools:sample/avatars" />
```

- ***RadioButton***: Botón para seleccionar una opción de una lista (código 2.3).

Código 2.3: activity_add.xml. *RadioButton*.

```
88                      <RadioButton
89                          android:id="@+id/measureRadioLarge"
90                          android:layout_width="wrap_content"
91                          android:layout_height="wrap_content"
92                          android:layout_weight="1"
93                          android:text="Largo" />
```

- ***CheckBox***: Botón para seleccionar varias opciones de una lista (código 2.4).

Código 2.4: activity_add.xml. *CheckBox*.

```
117                 <CheckBox
118                     android:id="@+id/seasonCBSummer"
119                     android:layout_width="wrap_content"
120                     android:layout_height="wrap_content"
121                     android:layout_weight="1"
122                     android:text="Verano" />
```

- ***Button***: Botón para llevar a cabo una acción (código 2.5).

Código 2.5: activity_add.xml. *Button*.

```
204                 <Button
205                     android:id="@+id/guardarButton"
206                     android:layout_width="match_parent"
207                     android:layout_height="wrap_content"
208                     android:layout_weight="1"
209                     android:background="@color/white"
210                     android:text="Guardar" />
```

Hay otros elementos usados como *ProgressBar* (punto 2.2.2), *CameraView* (punto 2.2.3) o *RatingBar* (punto 2.2.4) que se explican en los correspondientes apartados.

2.2.2 Pantallas de carga

La pantalla de carga se ha creado en una clase XML donde se define una barra de progreso y un texto para mostrar “Cargando...”.

Código 2.6: activity_loading.xml. Pantalla de carga.

```
10      <ProgressBar
11          android:id="@+id/progressBar2"
12          style="?android:attr/progressBarStyle"
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          app:layout_constraintBottom_toBottomOf="parent"
16          app:layout_constraintEnd_toEndOf="parent"
17          app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintTop_toTopOf="parent" />
19
20      <TextView
21          android:id="@+id/textView"
22          android:layout_width="wrap_content"
23          android:layout_height="wrap_content"
24          android:layout_marginTop="32dp"
25          android:text="Cargando..."
26          android:textSize="15sp"
27          app:layout_constraintEnd_toEndOf="parent"
28          app:layout_constraintStart_toStartOf="parent"
29          app:layout_constraintTop_toBottomOf="@+id/progressBar2" />
```

En cada una de las clases que se utiliza esta pantalla, se crea e inicializa un objeto *LoadingActivity* que posee los métodos `startLoadingDialog()` para visualizar la pantalla y `dismissDialog()` para ocultar la pantalla de carga. Ambos se pueden ver en el código 2.7.

Para mostrar el mensaje en la pantalla se utiliza un objeto *AlertDialog* que da las herramientas básicas para poder construir este tipo de diálogos. También se utiliza un objeto *inflater* para situar la pestaña por delante de todos los elementos que aparecen en pantalla.

Código 2.7: Clase LoadingActivity. Funciones para mostrar y ocultar la pantalla de carga.

```
7  public class LoadingActivity {
8
9      private Activity activity;
10     private AlertDialog dialog;
11
12     public LoadingActivity (Activity activity){
13         this.activity = activity;
14     }
15
16     public void startLoadingDialog() {
17         AlertDialog.Builder builder = new AlertDialog.Builder(activity);
18
19         LayoutInflater inflater = activity.getLayoutInflater();
20         builder.setView(inflater.inflate(R.layout.activity_loading, null));
21         builder.setCancelable(false);
22
23         dialog = builder.create();
24         dialog.show();
25     }
26
27     public void dismissDialog() {
28         dialog.dismiss();
29     }
30 }
```

2.2.3 Cámara

La cámara se ha implementado con un paquete externo llamado Camerakit. Se procede a su instalación según se indica en el código 2.8.

Código 2.8: build.gradle. Instalación de la cámara.

```
71     //Camera kit
72     implementation "com.wonderkiln:camerakit:0.13.1"
```

Después se usa el paquete en la clase XML con el código 2.9.

Código 2.9: activity_view.xml. Utilización de la cámara.

```
9      <com.wonderkiln.camerakit.CameraView
10         android:id="@+id/camera_view"
11         android:layout_width="match_parent"
12         android:layout_height="match_parent">
13
14     </com.wonderkiln.camerakit.CameraView>
```

Y por último se implementan los métodos que necesita en la clase java (código 2.10). Estos definen el comportamiento una vez se realice una foto (`onImage()`), un vídeo (`onVideo()`), surja algún error (`onError()`) o se produzca un evento (`onEvent()`).

Código 2.10: Clase AddActivity. Implementación de los métodos de la cámara.

```
108         cameraView.addCameraKitListener(new CameraKitEventListener() {
109             @Override
110             public void onEvent(CameraKitEvent cameraKitEvent) {
111
112             }
113
114             @Override
115             public void onError(CameraKitError cameraKitError) {
116
117             }
118
119             @Override
120             public void onImage(CameraKitImage cameraKitImage) {
121                 bitmap = cameraKitImage.getBitmap();
122                 bitmap = Bitmap.createScaledBitmap(bitmap, cameraView.getWidth(), cameraView.
                        getHeight(),false);
123                 color_int = createPaletteSync();
124                 cameraView.stop();
125
126                 detectPhoto();
127             }
128
129             @Override
130             public void onVideo(CameraKitVideo cameraKitVideo) {
131
132             }
```

2.2.4 Valoración

Es importante saber la opinión y los comentarios de los usuarios, por ello se ha implementado un botón para que la comunicación sea directa y se pueda ampliar el grupo de gente que utiliza la aplicación.

Para poder implementar la valoración hace falta un correo electrónico de Google desde donde se enviarán todos los comentarios.

Primero se instalan las dependencias del paquete Jakarta Mail (código 2.11).

Código 2.11: build.gradle. Instalación del paquete para conectarse con Gmail.

```
92      //Nota de la app
93      implementation 'com.sun.mail:android-mail:1.6.7'
94      implementation 'com.sun.mail:android-activation:1.6.7'
```

Después se implementa la parte gráfica con una barra de estrellas donde se puntúa la aplicación (código 2.12) y un cuadro de texto para poder escribir un comentario.

Código 2.12: activity_rate.xml. Barra para puntuar la aplicación.

```
22      <RatingBar
23          android:id="@+id/appRatingBar"
24          android:layout_width="wrap_content"
25          android:layout_height="wrap_content"
26          android:layout_marginTop="40dp"
27          android:theme="@style/RatingBar"
28          app:layout_constraintEnd_toEndOf="parent"
29          app:layout_constraintStart_toStartOf="parent"
30          app:layout_constraintTop_toBottomOf="@+id/textView3" />
```

Se puede observar en la línea 27 del código 2.12 que se usa el tema *RatingBar* definido en el código 2.21.

En la clase que actúa de interfaz, primero se capturan los datos que ha introducido el usuario y después se envía el mensaje (código 2.13).

Código 2.13: Clase RateActivity. Implementación para puntuar la aplicación.

```
51                  String puntuacion_str = String.valueOf(puntuacion.getRating());
52                  String messageSend = "Usuario: " + email + "\n\n"
53                          + "Puntuacion: " + puntuacion_str + "\n\n"
54                          +  mensaje.getText().toString();
55
56                  Properties properties = new Properties();
57                  properties.put("mail.smtp.auth", "true");
58                  properties.put("mail.smtp.starttls.enable", "true");
59                  properties.put("mail.smtp.host", "smtp.gmail.com");
60                  properties.put("mail.smtp.port", "587");
61
62                  Session session = Session.getInstance(properties,
63                          new javax.mail.Authenticator(){
64                              @Override
65                              protected PasswordAuthentication getPasswordAuthentication() {
66                                  return new PasswordAuthentication(from, pass);
67                              }
68                          });
69
70                  try{
71                      Message message = new MimeMessage(session);
72                      message.setFrom(new InternetAddress(from));
73                      message.setRecipients(Message.RecipientType.TO, InternetAddress.parse("app
                            .cloe@gmail.com"));
74                      message.setSubject(email + " - " + puntuacion_str);
75                      message.setText(messageSend);
76                      Transport.send(message);
77                      Toast.makeText(getApplicationContext(), "Gracias por haber dejado un
                            comentario<3", Toast.LENGTH_SHORT).show();
78                      goToHome(email);
79                  } catch (MessagingException e){
80                      throw new RuntimeException(e);
81                  }
```

Para ello, de la línea 51 a la 54 se crea el formato del cuerpo del mensaje y de la 56 a la 60 se definen las propiedades que se van a usar en el envío del mensaje (línea 70 a la 81), para lo cual se debe iniciar sesión en la cuenta de Gmail ya creada (línea 62 a la 68).

De esta forma, el usuario no necesita un correo propio para enviar un comentario. El email que se recibe tiene la siguiente estructura (figura 2.2).

Figura 2.2: Email con la valoración.

2.2.5 Mostrar imágenes

Para mostrar imágenes se ha optado por utilizar una librería externa llamada Glide y así reducir los recursos que usa la aplicación al mostrar una gran cantidad de fotos a la vez, como en la pantalla donde se visualizan las prendas digitalizadas.

En primer lugar se instalan las dependencias (código 2.14).

Código 2.14: build.gradle. Instalación de Glide.

```
75      //Glide for images
76      implementation "com.github.bumptech.glide:glide:4.11.0"
```

Después, cuando se desee utilizar esta librería se configura como en el código 2.15, usando en el XML un *ImageView*.

Código 2.15: Clase ShowViewAdapter. Uso de Glide.

```
59          Glide.with(photo.getContext())
60                  .load(Uri.parse(uri.get(position)))
61                  .placeholder(R.drawable.progress_animation)
62                  .into(photo);
63          name_view.setText(name.get(position));
```

Glide necesita un contexto, que se saca del objeto *ImageView* (método *with*), una fuente, ya sea un *bitmap* o una uri[1] (método *load*) y dónde se desea situar (método *into*).

En caso de que se desee insertar una imagen mientras se descargan los datos, se indica la uri o el *bitmap* de la foto deseada en el método `placeholder()`. A tal efecto, se ha establecido un PNG que se anima para dar la sensación de que es un GIF en el código 2.16.

[1] Identificador del archivo.

Código 2.16: progress_animation.xml. Uso de foto de carga en Glide.

```
<animated-rotate
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@mipmap/loading"
    android:pivotX="50%"
    android:pivotY="50%"/>
```

2.2.6 Características de la prenda

Este bloque se comunica con la inteligencia artificial en el momento de introducir una prenda nueva o con la base de datos en el momento de editar una prenda, que primero recupera toda la información almacenada en la nube y después, sobre esos datos, actualiza lo que se haya editado, enviando datos nuevos a la base de datos.

Gracias al diseño del bloque de características es posible ahorrar mucho tiempo y proporcionar comodidad al usuario, evitándole la necesidad de rellenar a mano toda la información de cada prenda, por ejemplo, nombre, color, etc., aunque no todos los campos se rellenan automáticamente. Esto permite que sea posible realizar una clasificación en el armario y que la aplicación pueda recomendar al usuario las prendas que registra formando un atuendo.

Se entiende como atuendo correcto un conjunto de ropa con coherencia respecto a las características que se han elegido, es decir, un conjunto de prendas suficientes para vestir completamente a una persona con una temporada y estilo determinados y con una combinación adecuada de colores según la teoría del color.

La aplicación utiliza dos métodos para rellenar los campos de la interfaz (necesarios para después realizar las recomendaciones):

- Un algoritmo basado en inteligencia artificial que discrimina el tipo de prenda que es (pantalón, camisa, etc.).
- Asignación directa por código de determinadas características dependiendo del tipo de prenda. Por ejemplo: un abrigo es una prenda de invierno, un jersey normalmente es una prenda de invierno y entretiempo, pero puede llegar a ser una prenda de verano, por ello, el usuario puede modificar las características para definir las de cada prenda concreta.

A continuación, se exploran las características elegidas para filtrar las prendas.

2.2.6.1 Características elegidas

Ciertas características se sopesaron para poder hacer la aplicación más intuitiva y agradable al usuario, evitando que tenga que escribir grandes cantidades de información sobre cada prenda que introduzca, y a su vez, que sean suficientes para poder hacer atuendos de forma correcta.

El **nombre** encabeza la prenda y sólo se utiliza para que el usuario pueda diferenciarlas. Se rellenará mediante la información detectada en la inteligencia artificial.

El **color** es fundamental para poder combinar la ropa de forma satisfactoria. Gracias a esta aplicación, una persona daltónica o que no sepa las directrices podrá hacer combinaciones realmente buenas y profesionales como se describe en la teoría del color. Esta variable es lo segundo que se detecta, y se

determina su valor mediante un algoritmo que evalúa la proximidad entre el color recogido y una serie de colores predefinidos.

La **medida** también es fundamental, no sólo para futuras actualizaciones en la previsualización de las predas, sino también para hacer las combinaciones oportunas. Por ejemplo, si se combina unos pantalones largos con unos calcetines, es recomendable que los calcetines sean cortos y no largos también. Esto vendrá regulado por la moda del momento. Se debe elegir manualmente y hay dos opciones: corto o largo.

La **estación** conviene saberla para poder recomendar atuendos calurosos o más frescos. Incluso hay ropa que puede llegar a ser polivalente. De momento se han establecido tres estaciones: verano, entretiempo (que aúna la primavera y el otoño) e invierno.

Y, por último, el **estilo** al que pertenecen los atuendos. Se han diferenciado cuatro posibles estilos que agrupan los más importantes para poder hacer una clasificación básica: deporte, trabajo, casual o de fiesta. La de deporte es el tipo de ropa que se utiliza para ejercitar el cuerpo. La de trabajo es la que se emplea en el ámbito laboral (esta clasificación es para la gente que no utiliza una ropa de trabajo ya establecida). La casual, es la que define lo que las personas usan para poder hacer las tareas del día a día más livianas como dar un paseo, comprar algo o ir a algún sitio. Y finalmente la de fiesta, que designa una actividad lúdico-festiva.

2.2.6.2 Detección del color

Primero se van a abordar los algoritmos y herramientas que hacen posible la clasificación del color.

2.2.6.3 Android Studio

En Android Studio, para poder codificar los colores se utiliza el RGB con valores en hexadecimal[2]. Pero para poder formar la paleta de colores de una imagen, primero se debe hacer una captura con el formato adecuado y a continuación se procesa mediante varias funciones definidas en un paquete llamado *Palette*[65].

2.2.6.3.1 *Palette*

El paquete *Palette*[66] tiene funcionalidades que nos permiten trabajar con paletas y perfiles de color. Estas son las principales:

- Establecer un color de texto acorde a un fondo con un color dinámico gracias a la generación automática de paletas de colores: colores que combinan entre sí (no se ha llegado a utilizar esta técnica).
- Sacar perfiles de una imagen que define los colores que la componen.

Del último punto, se pueden extraer seis principales, pero es posible llegar a un número elevado. Son los siguientes:

- Vibrante (*Vibrant*)
- Vibrante oscuro (*Vibrant dark*)

[2]Seis dígitos positivos enteros en base 16

- Vibrante claro (*Vibrant light*)
- Tenue (*Muted*)
- Tenue oscuro (*Muted dark*)
- Tenue claro (*Muted light*)

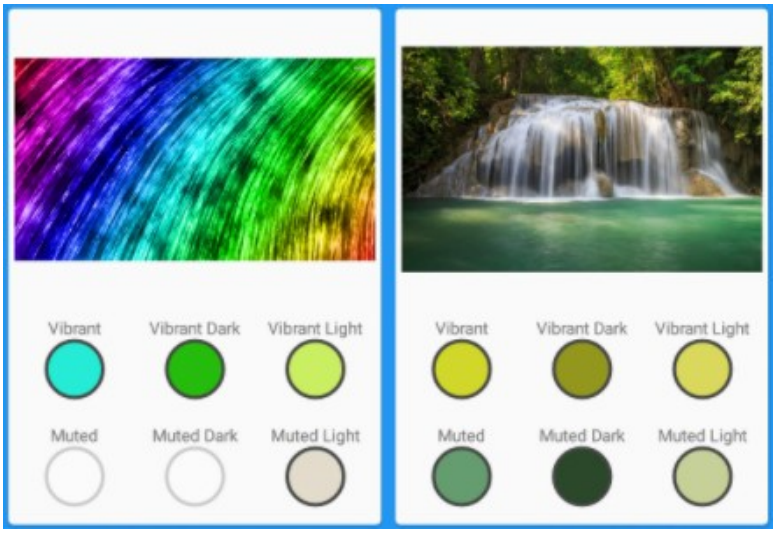

Figura 2.3: Perfiles de color de una imagen[15].

Los colores se extraen calculando la media ponderada con preferencias a la iluminación, saturación y cantidad de píxeles en la imagen (en ese orden). En general los perfiles vibrantes tienen más saturación que los tenues. Como no todas las imágenes contienen todos los perfiles de color, adicionalmente se debe proporcionar un valor predeterminado para mostrar en caso de no encontrar ningún dato.

En el trabajo, se ha abordado este apartado eligiendo una función extra que posee este paquete que se denomina `getPredominantColor()`. De este modo, se puede extraer el color que más espacio ocupe en la imagen de todos los perfiles.

2.2.6.3.2 Algoritmos para averiguar el nombre del color

Para poder combinar prendas adecuadamente es necesario conocer su color. Android no tiene una función nativa, ni siquiera en el paquete *Palette* que proporcione el nombre del color de un pixel, así que se han explorado dos opciones (distancia euclídea y delta E). En ambos casos es necesario disponer de un listado de colores básicos que se utilizarán para comparar el color de la prenda y así asignarle el nombre del color al que más se parezca.

Distancia Euclídea. La distancia euclídea o euclidiana[67] deriva del Teorema de Pitágoras (ecuación 2.1).

$$a^2 + b^2 = c^2 \tag{2.1}$$

En un espacio euclídeo de n dimensiones, la distancia entre dos puntos $A = (a_1, a_2, ..., a_n)$ y $B = (b_1, b_2, ..., n_n)$ se define de la siguiente forma (ecuación 2.2).

$$d_E(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + ... + (a_n - b_n)^2} \tag{2.2}$$

Las dimensiones que se utilizan en este caso para medir la distancia entre colores serían las componentes RGB. Por tanto, la ecuación queda de esta manera en el código (ecuación 2.3), siendo pixR, pixG, pixB las componentes del color detectado y r, g y b, las componentes del color de la lista ya definida.

$$difference = \sqrt{(pixR - r)^2 + (pixG - g)^2 + (pixB - b)^2} \tag{2.3}$$

Delta E. La expresión delta E o error delta (ΔE)[68][69] deriva de la palabra en alemán para "sensación" y significa literalmente "diferencia en sensación". Esta ecuación expresa esa percepción de color diferente cuando al ser humano se le muestran dos o más de ellos. Está basada en la distancia euclídea.

Hay muchas fórmulas para expresar este valor, pero la más común es la CIE76, que define la diferencia entre el color A y el B en un espacio CIELAB conformado por las coordenadas L* (luminosidad), a* y b* (que actúan como x e y en un espacio compuesto de 4 colores: rojo, verde, azul y amarillo) mostrado en la ecuación 2.4.

$$\Delta E = \sqrt{(L*_2 - L*_1)^2 + (a*_2 - a*_1)^2 + (b*_2 - b*_1)^2} \tag{2.4}$$

- ΔE $<$ 1 Los colores son indistinguibles.
- 2 $<$ ΔE $<$ 3 Los colores son parecidos.
- 4 $<$ ΔE $<$ 5 Los colores son bastante diferentes.
- ΔE $>$ 6 La diferencia entre los colores es obvia.

Esta ecuación se aplica principalmente en las imprentas.

2.2.6.4 Implementación

Para poder detectar el pigmento que tiene la prenda se utiliza una función de una biblioteca nativa de Android Studio que detecta el color predominante `getPredominantColor()`. En caso de que la prenda tenga exactamente la misma cantidad de dos o más colores, escogería el color por defecto pasado por parámetro, en este caso el negro.

Una vez se tiene el código de color en hexadecimal, se pasa a la siguiente parte. Por diseño de aplicación hay que tener especial cuidado a la hora de realizar las instantáneas. Se deben realizar de forma que no haya predominancia de otro color en la imagen, ya que, debido a las pruebas que se han realizado, si la prenda se saca desde lo suficientemente lejos como para detectar el color de fondo de forma uniforme, causará errores. La aplicación examina la foto y pone el color de fondo como el de la prenda cuando no es así.

Debido a que Android no tiene ninguna función nativa que facilite directamente el nombre del color, se ha detectado la distancia entre dos colores mediante el método de la distancia Euclídea. Este método dicta la distancia 3D de un punto en el espacio a otro punto. Y se ha implementado como en el código 2.18.

Código 2.17: Clase ColorUtils. Ecuación de la distancia euclídea.

```
312         public int computeMSE(int pixR, int pixG, int pixB) {
313             int difference = (int) (Math.sqrt(Math.pow((pixR - r),2) + Math.pow((pixG - g),2)
                    + Math.pow((pixB - b),2)));//Distancia Euclidiana
314             return difference;
315         }
```

Se han definido más de 50 colores en un *arraylist*[3], cada uno de ellos con sus componentes RGB en hexadecimal, su nombre y los colores que conjuntan armónicamente. La estructura se ve como en el código 2.18.

Código 2.18: Clase ColorUtils. Estructura de los colores.

```
25          ArrayList<ColorName> colorList = new ArrayList<ColorName>();
26          colorList.add(new ColorName("Azul alicia", 0xF0, 0xF8, 0xFF, new String[] {"Azul", "
                Cian", "Amarillo", "Morado", "Lila", "Violeta", "Rosa", "Rojo"}));
```

Así, cuando llega un dato, se va comprobando en bucle (código 2.19) qué color tiene el valor más bajo en cuanto a distancia con el dato introducido (línea 185) y se va guardando el menor de ellos (línea 187). De esta manera, cuando termina el bucle, se recoge la estructura del color con la menor distancia (línea 188).

Código 2.19: Clase ColorUtils. Bucle donde se obtiene el nombre del color.

```
183         for (ColorName c : colorList) {
184             mse = c.computeMSE(r, g, b);
185             if (mse < minMSE) {
186                 Log.d("mlData", "Color to analyze: " + c.name + "; DIF: " + mse);
187                 minMSE = mse;
188                 closestMatch = c;
189             }
190         }
```

De esta forma se obtiene el nombre del color similar que ha detectado el programa. Si bien es cierto que este método otorga un resultado muy fiable bajo buenas condiciones de distancia y enfoque, el factor iluminación es capaz de interferir en él, pues en los testeos de la aplicación se ha observado que diferentes tonos de iluminación o deslumbramientos y reflejos, afectan negativamente al resultado. Esto es debido a que no se realiza balance de blancos. Es decir, que, si se fotografía el mismo color con diferente iluminación, va a detectar diferentes colores. Este es un punto a mejorar en el futuro, para no tener que depender de la luz para dar un veredicto.

Otro factor a tener en cuenta es la detección de materiales translúcidos, metálicos o brillantes, donde tampoco detecta correctamente el color verdadero. Así que se debería trabajar más en profundidad en este apartado para poder detectar los colores sin importar el tipo de atuendo que se fotografíe.

[3]Estructura de datos parecida a los *arrays* pero que almacena los datos de forma dinámica.

2.2.6.5 Detección de la estación, el estilo y la medida

Estas dos características primeras comparten la forma en la que establecen los parámetros predefinidos para poder dar un veredicto de la variable.

Se han definido varios *arrays* donde se han señalado las diferentes prendas que pertenecen a cada grupo y después, según la prenda que haya detectado en la inteligencia artificial, se activan en la aplicación los *checkboxes*[4] apropiados.

Código 2.20: Clase AddActivity. Método para elegir de forma predeterminada la temporada.

```
363     private void chooseSeason() {
364         String[] sumArray = {"Vestido", "Traje de baño", "Collar", "Pulsera", "Corbata", "
                Esmoquin", "Gafas", "Gorra", "Sombrero", "Chaqueta", "Pulsera", "Joyería", "Cintur
                ón", "Leggings", "Polo", "Zapatos", "Bolso", "Zapatillas", "Vestido de fiesta", "
                Pantalón vaquero", "Mochila/Bolsa"};
365         String[] wintArray = {"Vestido", "Collar", "Pulsera", "Corbata", "Esmoquin", "Gafas",
                "Sombrero", "Sombrero de punto", "Jersey", "Bufanda", "Chaqueta", "Pulsera", "
                Joyería", "Cinturón", "Leggings", "Zapatos", "Bolso", "Zapatillas", "Vestido de
                fiesta", "Pantalón vaquero", "Mochila/Bolsa"};
366         String[] halftArray = {"Vestido", "Collar", "Pulsera", "Corbata", "Esmoquin", "Gafas",
                 "Gorra", "Sombrero", "Jersey", "Chaqueta", "Pulsera", "Joyería", "Cinturón", "
                Leggings", "Polo", "Zapatos", "Bolso", "Zapatillas", "Vestido de fiesta", "Pantaló
                n vaquero", "Mochila/Bolsa"};
367
368         //Summer
369         for (int i = 0; i<sumArray.length; i++) {
370             if (sumArray[i] == predict) {
371                 summer.setChecked(true);
372             }
373         }
374
375         //Winter
376         for (int i = 0; i<wintArray.length; i++) {
377             if (wintArray[i] == predict) {
378                 winter.setChecked(true);
379             }
380         }
381
382         //Halftime
383         for (int i = 0; i<halftArray.length; i++) {
384             if (halftArray[i] == predict) {
385                 halftime.setChecked(true);
386             }
387         }
388     }
```

De esta forma, cada prenda está clasificada según esas características y se evita que el usuario tenga que rellenar todo sin ninguna ayuda desde cero.

Sin embargo, la detección de la medida sí que se debe de hacer de manera completamente manual porque no es una característica intrínseca de una prenda. Por ejemplo, una chaqueta puede ser muy larga o muy corta. Además, la estación del año tampoco define nada, porque, por ejemplo, no todo en verano es ropa de corta. Por tanto, la medida es la única variable que se elige manualmente.

[4]Estructura para señalar opciones en la interfaz de la aplicación.

2.2.7 Archivos de configuración generales

Los archivos de configuración generales permiten establecer variables que son accesibles en todas las clases para poder definir módulos globales como los temas

En cada tema se pueden concretar colores o valores de variables ya existentes de clases nativas de Android, como por ejemplo si la pantalla flota o es translúcida.

Código 2.21: themes.xml. Temas definidos.

```
3       <style name="Theme.FirebaseTutorial" parent="Theme.AppCompat.Light.NoActionBar">
4           <!-- Primary brand color. -->
5           <item name="colorPrimary">@color/purple_500</item>
6           <item name="colorPrimaryVariant">@color/purple_700</item>
7           <item name="colorOnPrimary">@color/white</item>
8           <!-- Secondary brand color. -->
9           <item name="colorSecondary">@color/teal_200</item>
10          <item name="colorSecondaryVariant">@color/teal_700</item>
11          <item name="colorOnSecondary">@color/black</item>
12          <!-- Status bar color. -->
13          <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</
                item>
14          <!-- Customize your theme here. -->
15      </style>
16
17      <style name="Theme.FirebaseTutorial.TemaPopup">
18          <item name="android:windowIsFloating">true</item>
19          <item name="android:windowIsTranslucent">true</item>
20          <item name="android:windowCloseOnTouchOutside">true</item>
21      </style>
22
23      <style name="RatingBar" parent="Theme.AppCompat">
24          <item name="colorControlNormal">@color/black</item>
25          <item name="colorControlActivated">@color/white</item>
26      </style>
```

Se han utilizado tres temas en total (código 2.21). El primero (línea 3 a 15), que define el tema principal por el que se rigen todas las clases, el segundo (línea 17 a 21), que lo define para la pantalla que aparece al elegir un atuendo y el tercero (línea 23 a 26), que concreta los colores de la barra de la valoración de la aplicación para la pantalla de puntuación de la aplicación.

2.3 Bloque de bases de datos

Las bases de datos hoy en día son fundamentales para poder serializar gigantescas cantidades de información y poder hacer tanto un seguimiento, como un análisis exhaustivo de las variables recogidas. Es posible filtrar y extraer datos de interés gracias a los parámetros que se establecen para recoger las variables con las que se desea trabajar. De esta forma, se pueden saber todos los datos necesarios y que sean requeridos de forma instantánea, así como modificarlos o eliminarlos intuitivamente.

Esta parte es la principal del trabajo porque es la que se comunica con más bloques. Se encarga de enviar o almacenar los datos para luego poder recuperarlos y verlos, modificarlos o filtrarlos según lo que desee el usuario.

2.3.1 Firebase

En este trabajo se ha optado por desarrollar la base de datos con Firebase[70], una plataforma creada por Google que facilita a los desarrolladores las herramientas que necesitan para hacer posible la programación de aplicaciones de alta calidad. Haciendo uso de estas, se obtienen los aspectos básicos para que la aplicación que se desarrolla pueda crecer en cantidad de usuarios y en calidad de servicios.

Posee muchas extensiones que se utilizan para poder dar soporte a diferentes características como *Analytics* para recopilar información de los datos de uso de la aplicación o los métodos que se tienen para poder entrar en la aplicación (*Invites*), entre otros. Hay que destacar que trabaja con Tensorflow, por lo que en un futuro incluso se podría seguir el desarrollo de la aplicación con el mismo esqueleto para añadir un modelo personalizado de inteligencia artificial y así poder reconocer muchas más prendas de las que puede reconocer ahora, como se ve en el apartado de inteligencia artificial.

2.3.2 La base de datos: Firestore

Por la parte que compete al proyecto, se puede elegir entre dos tipos de bases de datos según el enfoque que se desee. Ambas están basadas en la nube y admiten sincronización en tiempo real.

El primer tipo es el **Cloud Firestore**[71], la más novedosa. Este modelo aprovecha lo mejor de su alternativa con un modelo de datos renovado. Hay que destacar que el tipo de consultas que se realizan con este modelo son mucho más ricas en contenido y rápidas, además de que se puede escalar a un nivel más alto que su predecesor. Este es el modelo que se ha utilizado.

En cambio, el otro tipo, **Realtime Database**, es la base de datos original que tiene Firebase. Es una solución de baja latencia destinada a las aplicaciones para dispositivos móviles que necesitan sincronización de estados entre varios clientes en tiempo real.

2.3.3 Definiciones

Se ha elegido una forma de serializar los datos NoSQL[72] orientada a los documentos. Los datos no son almacenados ni en tablas ni en filas, sino en documentos compuestos por conjuntos de pares clave-valor. Así que es conveniente dejar claros los conceptos básicos que se van a utilizar de ahora en adelante.

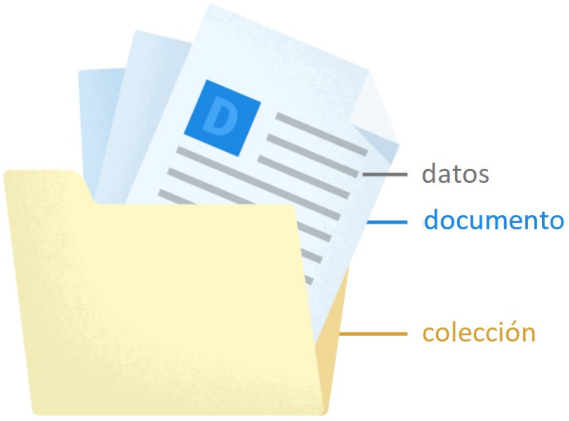

Figura 2.4: Elementos de una base de datos.

La unidad básica de almacenamiento de este tipo de bases de datos es el documento, un registro que usa pocos recursos y contiene campos con valores asignados. Por diseño, es posible utilizar diversos tipos

de datos básicos como booleanos, *strings*, números, marcas de tiempo o bien tipos de datos complejos como los mapas, para poder estructurar datos. No necesariamente debe haber un orden de variables, pueden estar compuestos por campos diferentes, aunque no está diseñado para almacenar otros documentos.

Una colección es el contenedor de los documentos. Cada documento tiene su ID único dentro de una colección que se rellena manual o automáticamente. Se debe tener especial cuidado con este tipo de estructura ya que no puede contener campos sin procesar con valores de manera directa ni otras colecciones. Además, no hace falta crear ni borrar expresamente una colección, ya que, al depender de los documentos que contiene, si se crea el primer documento de una colección, automáticamente se crea el documento junto con la colección. Y de la misma manera, si se borran todos los documentos de una colección, la colección dejará de existir también.

Por último, cabe destacar la existencia de subcolecciones, que son conjuntos de duplas colección-documento anidadas entre sí, pudiendo llegar a tener hasta 100 niveles de profundidad. De esta forma se hace posible la jerarquía de la información que contiene la base de datos.

En general, este diseño no utiliza esquemas, por lo que se tiene libertad total a la hora de estructurar la base de datos.

2.3.4 Diseño de la base de datos

Como ya se ha dicho anteriormente, el modo de almacenamiento es NoSLQ, por tanto, se ha tenido que pensar meticulosamente en el diseño[73] de esta base de datos según las reglas que establece la plataforma Firebase.

- **Los documentos tienen un límite de capacidad** (1MB). Conviene almacenar sólo la cantidad necesaria de información.
- **No se pueden extraer parcialmente los documentos**, se extrae el documento entero. Es recomendable que, si hay partes del documento que son privadas o de uso interno, no vayan en el mismo documento donde se desea recuperar la información.
- **Las *queries* son poco profundas**. Esto quiere decir que, si se recupera información de un documento y este tiene subcolecciones, únicamente se recuperará el documento sin las subcolecciones.
- **Se factura por número de lecturas y escrituras**. Hay que ser precavidos y entender este tipo de bases de datos para no causar lecturas extra y que la aplicación tenga un mal rendimiento.
- **Las *queries* encuentran documentos en las colecciones**. No es posible almacenar colecciones ni variables en colecciones ni documentos en documentos. Siempre, aunque haya subcolecciones, los datos se estructuran en documentos que contienen las colecciones.
- **Los *arrays*[5] no son un tipo de dato aceptado**. Es mejor utilizar mapas con un índice y un valor.

Una vez se tienen en cuenta todas estas reglas, más algunas recomendaciones oficiales[74], se ha decidido que la estructura con la que se va a trabajar en este TFG es la siguiente.

La figura 2.5 está dividida en varias partes. Dos estructuras principales de las que se dividen en diferentes secundarias. Esto es para que en el futuro se pueda buscar en la aplicación por *outfit* y por usuarios y así facilitar el camino para hacer la red social.

[5] Conjuntos de datos de un mismo tipo establecidos en una zona de almacenamiento contiguo.

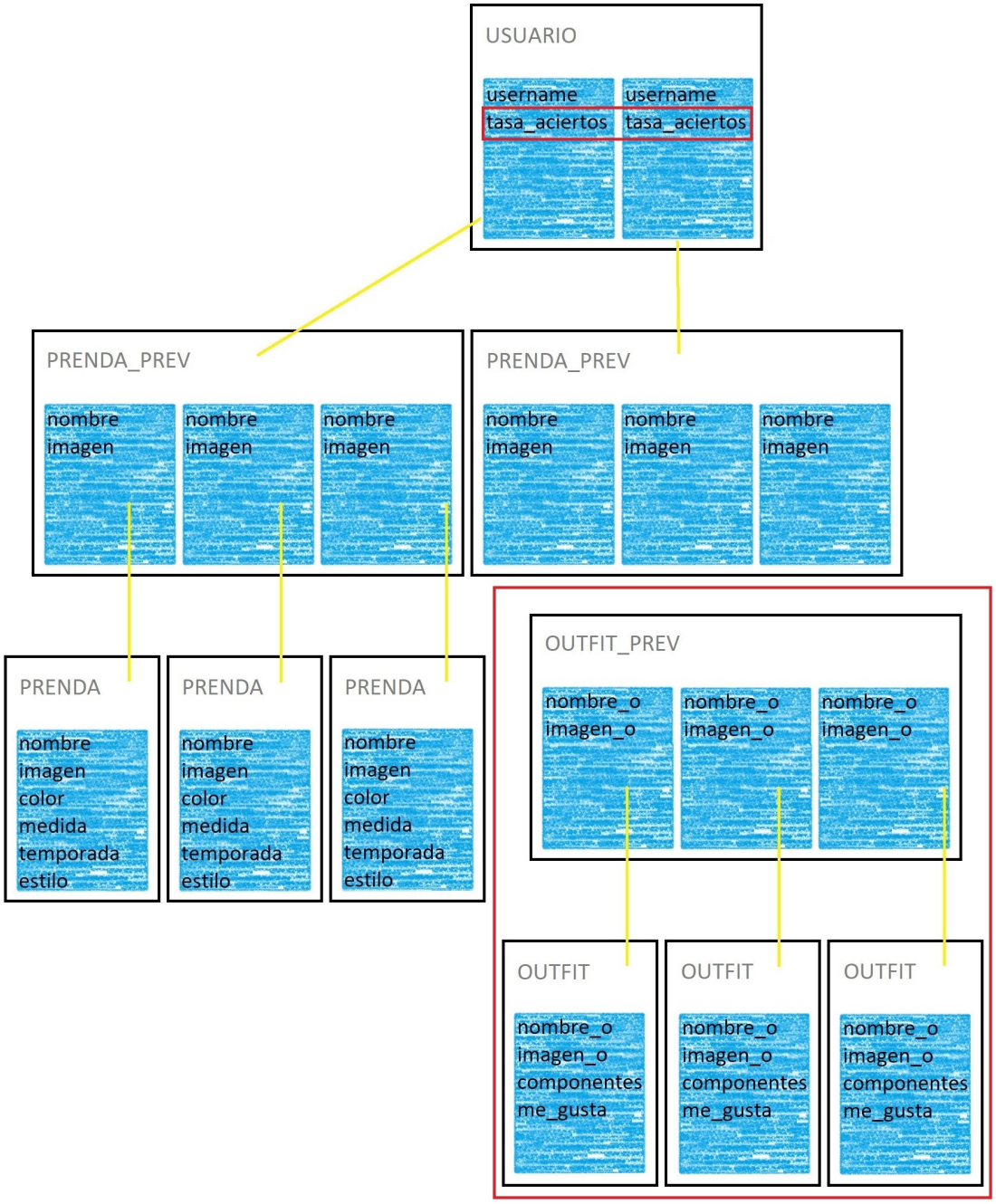

Figura 2.5: Vista simplificada de la base de datos.

También se aprecia que hay partes duplicadas en la base de datos como en el caso de `PRENDA_PREV` y `PRENDA`, el nombre y la imagen o en `OUTFIT_PREV` y `OUTFIT`, que sucede lo mismo. Esto tiene que ser así por el diseño NoSQL de la base de datos. Hay que recordar que en este diseño se deben crear documentos acordes con lo que se quiera enseñar en una página determinada de la aplicación y después mediante código extra, se mantienen esas variables iguales en los documentos para evitar inconsistencias.

La primera parte corresponde con los usuarios. En esta parte simplemente se guarda información básica sobre éstos, como puede ser el nombre de usuario y la tasa de aciertos de la aplicación a la hora de recomendarle *outfits*. El nombre de usuario aparece en la página principal de la aplicación.

Después se diferencia la subcolección de la ropa, `PRENDA_PREV`, que se descompone a su vez en el documento completo de cada uno de los atuendos `PRENDA`. En la primera colección interesa sólo poder

acceder al nombre de la prenda y a su foto para mostrarlo en la pantalla correspondiente de ropa que ha introducido en la aplicación y, si le da clic, se accede al documento inmediatamente inferior, que ya tiene toda la información que el usuario almacenó al hacer por primera vez la foto, o bien, editar el atuendo.

Y, por último, todo lo referente a los *outfits*, que se ha realizado de la misma forma que la parte anterior. Primero tiene su colección con los nombres y las instantáneas y, si se quiere más información, se otorgará la información de los diferentes atuendos que lo componen añadido a si le gusta o no.

Las partes que están con un recuadro en rojo no se han llegado a desarrollar, pero será necesario implementarlas de esa forma si se desea poder visualizar los atuendos que se han elegido a lo largo del tiempo. También se podría tener esto en cuenta para hacer recomendaciones más personalizadas, de manera que, si un usuario tiene preferencia por una prenda determinada, una armonía de colores e incluso un color determinado, que se detectase y en las recomendaciones apareciese con más frecuencia esos elementos.

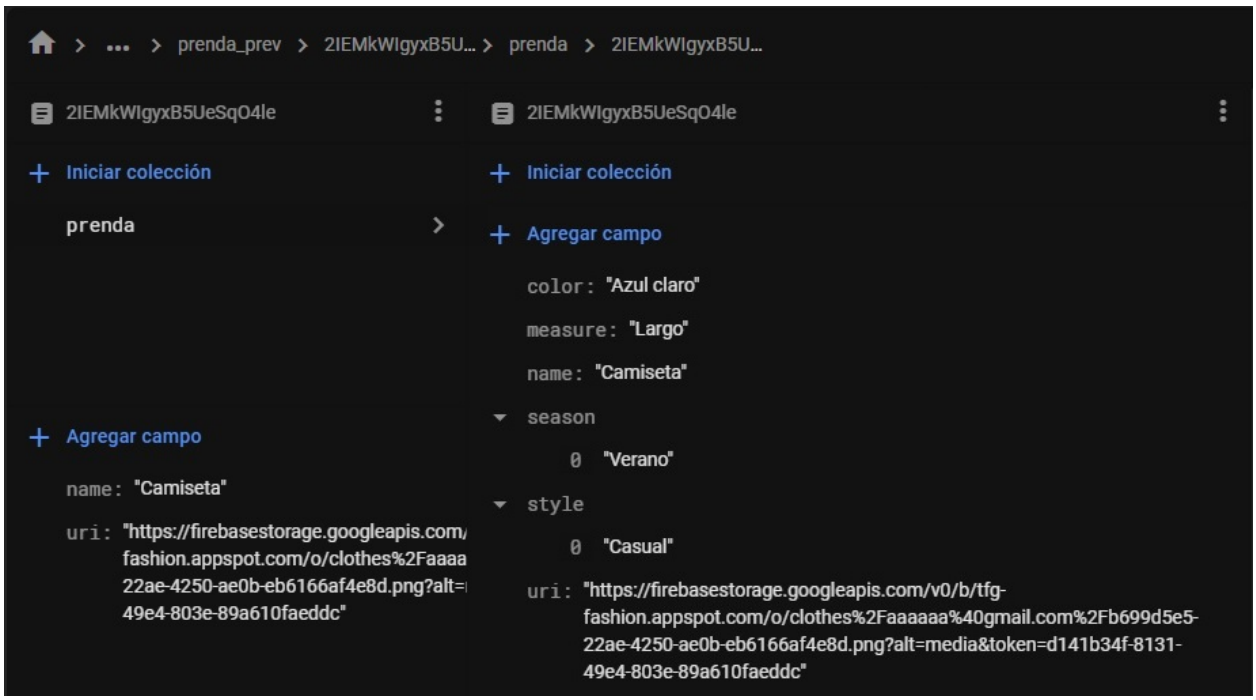

Figura 2.6: Ejemplo de cómo queda en la consola de Firebase.

En la figura 2.6 se visualiza un ejemplo en la consola de cómo se almacena una prenda en la base de datos. En la parte de arriba a la izquierda, se ve la ruta del archivo, y la pantalla se divide en dos columnas. A la derecha, el documento (con el nombre arriba), que depende de la colección prenda, con el nombre del documento en la parte superior y su visualización en la parte de abajo a la izquierda.

2.3.5 Instalación

2.3.5.1 Firebase

Este paquete se instala de forma parecida al paquete de la IA, pero exceptuando que se deben seguir unos pasos adicionales para configurar no sólo el paquete en la aplicación, sino que también hay que establecer el servidor desde el que se va a acceder a los datos de interés. Hay dos formas de instalarlo[75]: mediante Firebase Console o mediante Firebase Assistant. Se ha elegido la primera forma.

2.3.5.1.1 Agregar Firebase mediante Firebase Console

Se va a empezar por el entorno web. Primeramente, hay que crear un proyecto en Firebase Console dándole clic a 'Agregar proyecto' como se muestra en la imagen 2.7 y seguir los pasos que se indican por pantalla.

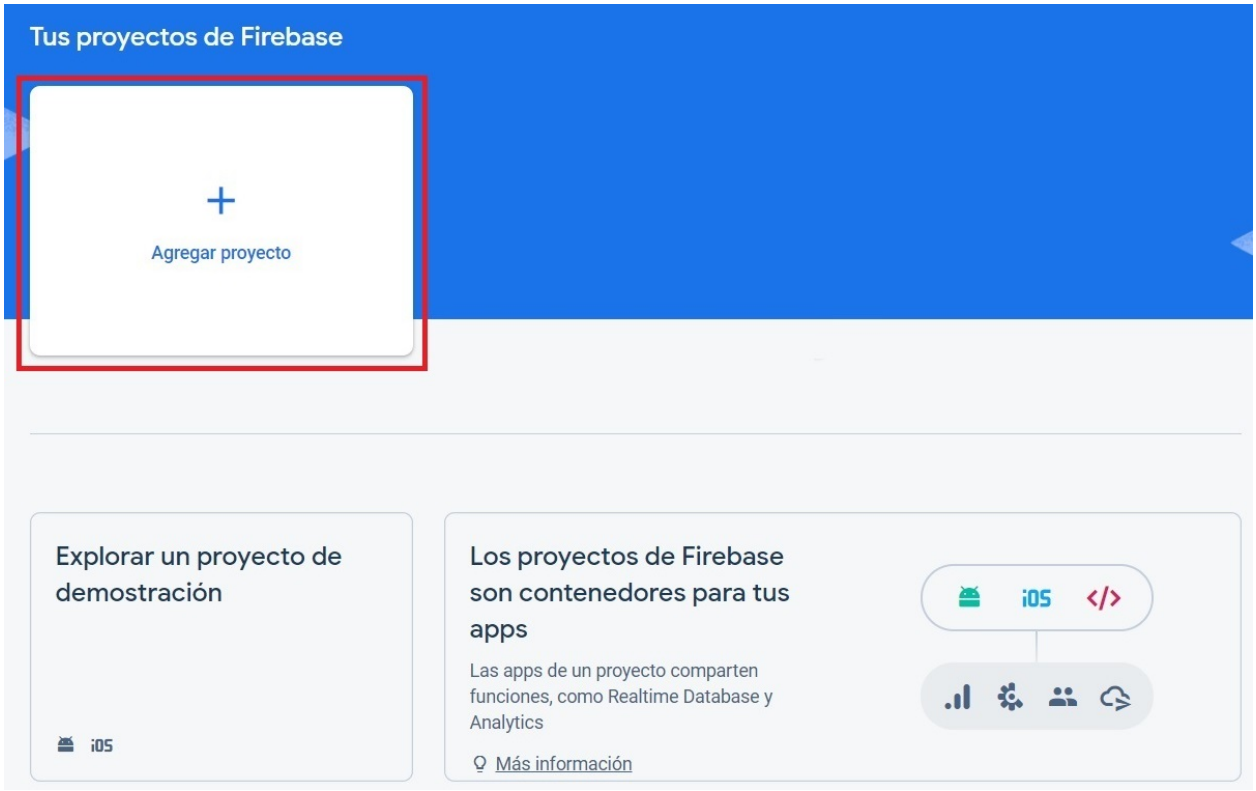

Figura 2.7: Botón agregar proyecto.

Una vez creado, el proyecto aparece en la misma pantalla que la figura 2.7 al lado del botón que se seleccionó anteriormente.

A continuación, se pedirá el modo para aplicar las reglas de seguridad de Cloud Firestore. Existen dos modos de inicio:

- **Modo de prueba.** Permite que todos los usuarios de la aplicación lean y reemplacen los datos de cualquier perfil (modo recomendable para principiantes que empiezan a usar bibliotecas cliente para dispositivos móviles y web).
- **Modo bloqueado.** Rechaza todas las lecturas y escrituras de clientes tanto de móviles como de la web (sólo pueden acceder los servidores de aplicación autenticados, que son C#, Go, Java, Node.js, PHP, Python, Ruby).

El siguiente paso es seleccionar el lugar donde se sitúa el servidor para la base de datos que se desea implementar. Esta configuración es la ubicación predeterminada de los recursos de Google Cloud Platform (GCP) del proyecto. Esta ubicación se va a utilizar para los recursos que requieran esta característica, como, por ejemplo, Cloud Storage. Si no se selecciona una ubicación, el proyecto establece una predeterminada para los recursos GCP definida durante la creación del proyecto o cuando se configuró algún servicio que lo requiera.

Con esto ya casi se ha terminado de configurar la parte del entorno web, solo queda asociarlo con la aplicación que se tiene creada en Android con anterioridad.

Figura 2.8: Página del proyecto de Firebase.

Para ello, en la página principal del proyecto, se debe clicar en el botón 'Añadir app' como indica en la figura 2.8 o bien hacer clic en el icono de Android que sale antes de ese botón.

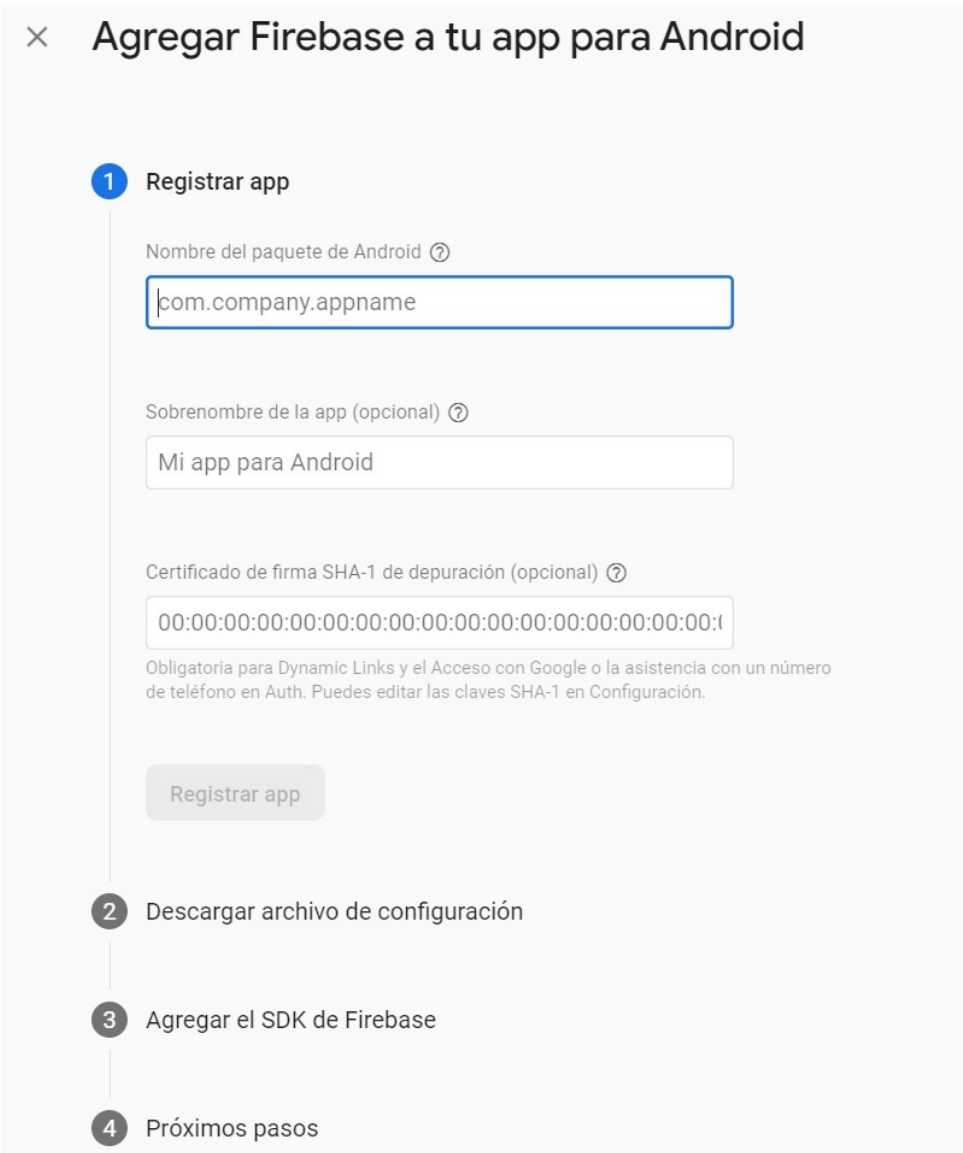

Figura 2.9: Pantalla para enlazar Firebase con Android.

La pantalla que sale después es una guía con una serie de pasos para poder agregar Firebase a la app de Android como se muestra en la figura 2.9.

Una vez se ha cumplimentado el formulario, hay que configurar el proyecto Android para poder dar por concluido este proceso.

Se incluyen las bibliotecas y servicios necesarios de Google al archivo *gradle* de nivel raíz de la aplicación y el plugin a nivel de módulo, además de añadir el SDK de Firebase para utilizar el BoM (línea 45 del código 2.22) y así tener siempre las bibliotecas compatibles actualizadas automáticamente porque no hace falta escribir la versión de cada biblioteca. En este caso se ha instalado el módulo de autenticación (línea 53) y el que recopila estadísticas (línea 49).

Código 2.22: Build.gradle (aplicación). Instalación de Firebase.

```
44      // Import the Firebase BoM
45      implementation platform('com.google.firebase:firebase-bom:27.1.0')
46
47      // Add the dependency for the Firebase SDK for Google Analytics
48      // When using the BoM, don't specify versions in Firebase dependencies
49      implementation 'com.google.firebase:firebase-analytics'
50
51      // Declare the dependency for the Firebase Authentication library
52      // When using the BoM, you don't specify versions in Firebase library dependencies
53      implementation 'com.google.firebase:firebase-auth'
```

El último paso es la sincronización y con ella se da por concluida la configuración y se puede empezar a trabajar con estas herramientas de Google.

2.3.5.1.2 Agregar Firebase mediante Firebase Assistant

Esta forma alternativa de instalación es mucho más inmediata y permite poder gestionar todo desde Android Studio. De esta manera, se registra la app con un proyecto de Firebase y se instalan los correspondientes archivos y las dependencias necesarias directamente en el proyecto de Android.

En primer lugar, hay que asegurarse que se utilizan las versiones más recientes de este entorno de desarrollo y Firebase Assistant en Ayuda > Buscar actualizaciones desde Windows.

Por otro lado, hay que abrir Firebase Assistant en Android Studio: Herramientas > Firebase. En el panel Asistente hay que elegir un producto de Firebase para agregar a la aplicación. Para ello, hay que expandir la sección y seleccionar el producto deseado. Una vez en el producto, se da clic en 'Conectarse a Firebase' y después se realizarán una serie de pasos para finalmente agregarlo. Se proporciona ejemplos, casos de uso y toda la documentación disponible.

Lo último es sincronizar la app para validar todas las dependencias y seguir con las instrucciones de configuración restantes en el panel Asistente.

2.3.5.2 Firestore

Para agregar Firestore a la aplicación, simplemente se debe poner la parte del código 2.23 en el *gradle* a nivel de aplicación y sincronizar el proyecto.

Código 2.23: Build.gradle (aplicación). Instalación de Firestore.

```
55      // Declare the dependency for the Firebase Firestore
56      implementation 'com.google.firebase:firebase-firestore'
```

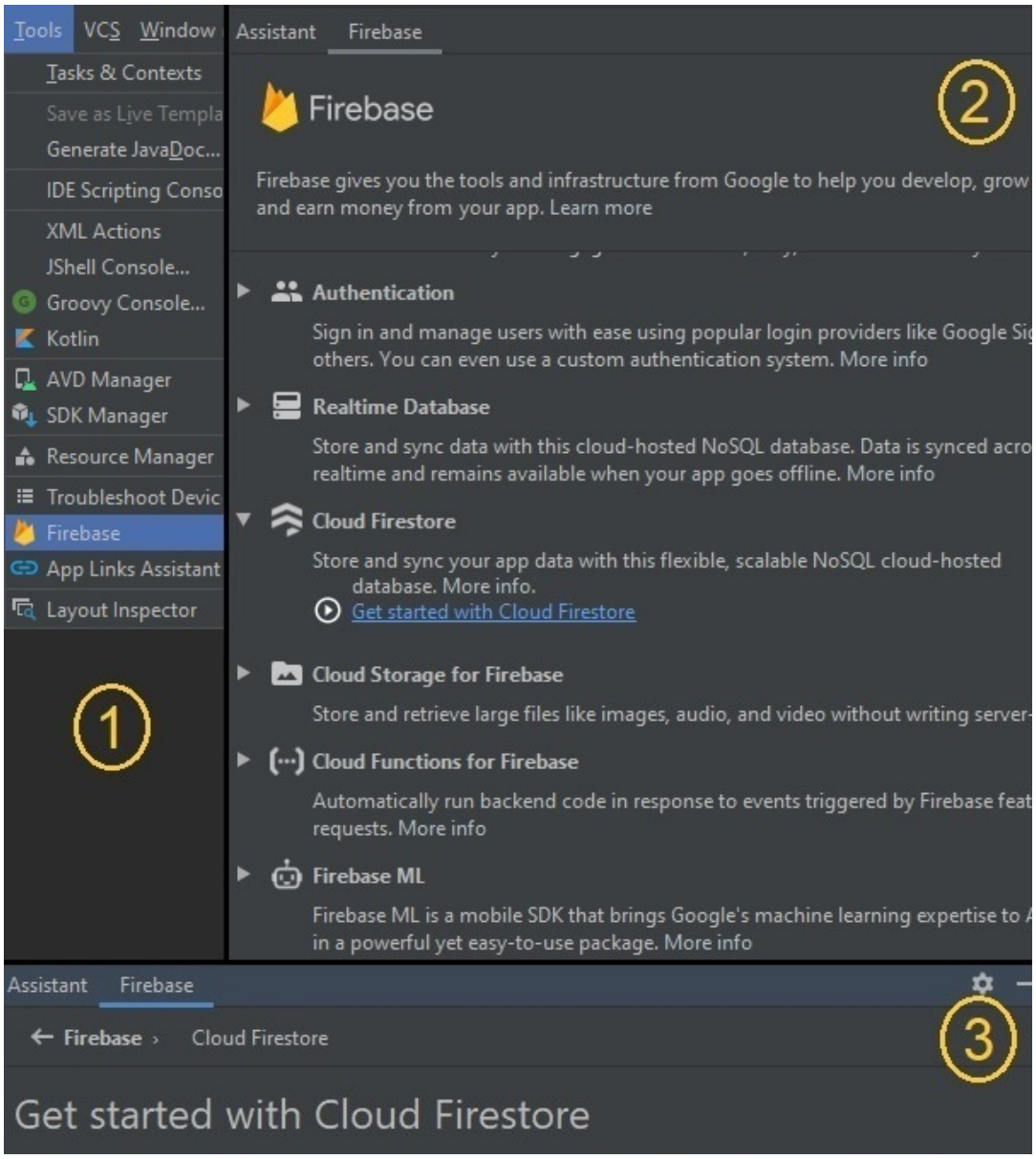

Figura 2.10: Instalación de Firebase mediante Firebase Assistant dividido en pasos. 1) Seleccionar Firebase en el menú Herramientas. 2) Elegir el producto que se desee instalar. 3) Seguir los pasos.

2.3.6 Implementación

Hay que tener en cuenta que siempre que se accede a un documento, colección o subcolección, se debe guardar en una variable la referencia a la base de datos y a la ubicación del elemento en la base de datos asociada al proyecto sin importar si existe información o no. Además, es recomendable hacer uso de los métodos `addOnSuccessListener()` y `addOnFailureListener()` para poder informar al usuario de los resultados de las operaciones, éxito o fracaso respectivamente.

Con esta premisa, la forma de poder plasmar todo lo que se ha visto hasta ahora en este apartado en el código se expone a continuación.

En primer lugar, para hacer la validación de las credenciales cuando se entra a la aplicación, se puede iniciar sesión (código 2.25) o crear una cuenta en el caso de que sea un usuario nuevo (código 2.26). El objeto `authInstance` se declara `FirebaseAuth authInstance;` y se inicializa como en el código 2.24.

Código 2.24: Clase AuthActivity. Inicializar el objeto FirebaseAuth.

```
55          // Initialize Firebase Auth
56          authInstance = FirebaseAuth.getInstance();
```

Código 2.25: Clase AuthActivity. Iniciar sesión.

```
100                     authInstance.signInWithEmailAndPassword(email_s, pass_s)
```

Código 2.26: Clase AuthActivity. Registrarse.

```
66                      authInstance.createUserWithEmailAndPassword(email_s, pass_s)
```

Ya dentro de la aplicación, para mantener una comunicación con la base de datos, en este TFG se realizan tres operaciones distintas: envío, petición y modificación de datos. Siempre se parte de la referencia a la base de datos y la referencia para poder identificar el campo, documento o colección al que se desea acceder.

La referencia a la base de datos en este proyecto siempre se va a llamar `dbInstance` y se declara e inicializa de la siguiente forma: `FirebaseFirestore dbInstance = FirebaseFirestore.getInstance();`.

Esta referencia se puede poner de varias maneras. Como una ruta `dbInstance.document("users/"+ email + "/prenda_prev/" + docId + "/prenda/" + docId);` o como una sucesión de métodos como en el código 2.27.

Código 2.27: Clase AddActivity. Ruta de referencia.

```
324                             dbInstance.collection("users").document(email)
325                                     .collection("prenda_prev").document(docId)
326                                     .collection("prenda").document(docId)
```

2.3.6.1 Enviar datos (método *set*)

En la aplicación, se envían datos a la hora de añadir una prenda o modificarla. Hay que tener en cuenta que cuando se envían los datos, si el documento no existe, se crea, y si ya existe, se añade a la información que contiene la base de datos.

Para enviar datos, se debe incluir en la ruta de la referencia a un documento este método con el objeto que se desee almacenar.

2.3.6.2 Recibir datos (método *get*)

En caso de que se desee recibir datos, se utiliza el método *get* y se puede usar de varias maneras: Para poder recibir todos los documentos de una colección determinada, se realiza un *get* sobre una referencia a una colección; si se desea visualizar un sólo documento, entonces se debe inicializar una referencia a ese archivo y, si se desea filtrar[76] y recuperar información que coincida con ciertos parámetros definidos por el usuario, se introducen en métodos anteriores al *get*.

2.3.6.2.1 Modelos óptimos para filtrar información

Pero hay que tener especial cuidado, ya que mientras se trabajaba con esta herramienta haciendo las consultas, se observó que se debía implementar código adicional para saber cuándo terminaba de recorrer la base de datos. Esto es porque se constató que cuando se realizaban este tipo de operaciones, no se hacían de forma atómica (de una sola vez), si no que el bucle que recibía los documentos coincidentes se interrumpía, saliendo y volviendo a entrar de nuevo recuperando la posición en la que estaba. Esto es causado porque este tipo de métodos son asíncronos, es decir, se desconoce cuándo van a terminar.

Por ello la solución a la que se llegó es dividir los procesos en clases diferentes para poder darle tiempo suficiente (además se controla con pantallas de carga) y así ir pasando los datos de una clase a otra.

También serviría en una futura actualización para construir una aplicación con una interfaz de usuario más intuitiva. De esta manera, se enseñarían las prendas del usuario directamente en esa pantalla eliminando los botones, y así, cuando se desee añadir alguna prenda o que la aplicación te recomiende algún atuendo, esos botones estarán accesibles mediante un menú desplegable.

2.3.6.3 Modificar datos (método *update*)

Para poder mantener los campos duplicados con valores correctos y no crear inconsistencias, se ha utilizado la escritura en bloque[77], que se encarga de hacer un número determinado de escrituras de forma que, en el caso de que falle alguna, todo volverá al estado anterior. De esta manera, todas las operaciones se realizan correctamente.

Código 2.28: Clase ShowActivity. Escritura en lotes.

```
287         WriteBatch batch = dbInstance.batch();
288
289         DocumentReference prenda = dbInstance.collection("users").document(email)
290                                              .collection("prenda_prev").document(id)
291                                              .collection("prenda").document(id);
292
293         DocumentReference prenda_prev = dbInstance.collection("users").document(email)
294                                                   .collection("prenda_prev").document(id);
295
296         if (field.equals("name")) {
297             Log.d("editData", "Campo - " + field + ": " + value);
298             batch.update(prenda_prev, field, value);
299         }
300
301         batch.update (prenda, field, value);
302
303         batch.commit().addOnCompleteListener(new OnCompleteListener<Void>() {
```

En el caso de la clase donde se editan los datos de las prendas (código 2.28), se crean dos referencias: una a `prenda_prev` y otra a `prenda`, que llevan a los documentos seleccionados de las colecciones con los mismos nombres. Previamente, en la línea 287 se ha definido un objeto `WriteBatch` que será el encargado de realizar la operación.

Para poder indicar las operaciones simples, se utiliza `batch.update()`, un método que recibe tres variables por parámetro. En este caso, la instancia al documento en la base de datos, el nombre del campo de la base de datos y el valor que se desea establecer. Entonces, en caso de que se edite el nombre de la

prenda, se realizan dos escrituras, una en `prenda_prev` y otra en `prenda`. Y para hacer la operación atómica, simplemente se realiza con `batch.commit()`, que se le asocian los métodos correspondientes para notificar del éxito o del fracaso.

2.4 Bloque Inteligencia Artificial

La inteligencia artificial es una de las partes más atractivas en este trabajo de fin de grado. Como se ha mencionado anteriormente, este proyecto se basa fundamentalmente en el uso de la IA para el reconocimiento de prendas mediante fotografías a través del móvil y la posterior organización de un grupo de atuendos que puedan interactuar entre ellos formando conjuntos adaptados y personalizados a la manera de vestir del usuario. En este apartado se explora la serie de procesos que sigue el *software* para ser capaz de procesar una fotografía y devolver la serie de metadatos necesarios para que se pueda serializar en una base de datos. Esto ayuda a que, a la hora de guardar la información de la prenda, el usuario tenga a su disposición una serie de propuestas para agilizar el proceso de guardado y que no se vuelva una tarea repetitiva y tediosa.

Para desarrollar este punto, se ha utilizado el kit de aprendizaje automático de Google (Kit ML)[78] que proporciona las herramientas necesarias para poder procesar imágenes y reconocer atuendos en las instantáneas que se realicen en la cámara. Esta interfaz de programación de aplicaciones (API) facilita la tarea que se abarca en este apartado: identificar de forma precisa la ropa fotografiada. Para ello, este paquete cuenta con un modelo preconfigurado, entrenado y optimizado por el cual procesa la captura y es capaz de indicar, con altos niveles de seguridad, qué prenda es la que corresponde con la instantánea.

Se ha elegido utilizar el modelo de inteligencia artificial estándar para poder hacer la clasificación de las prendas de las fotografías porque el número de clases que puede distinguir es suficiente.

Este bloque ayuda a completar la información del nombre de la prenda cuando se desea añadir una nueva.

2.4.1 Descripción de ML Kit

ML kit es una herramienta que lleva la experiencia de Google del aprendizaje automático directo a las aplicaciones de Android e iOS. Permite poder tener en el móvil potentes modelos de aprendizaje automático (ML o *machine learning*), capaces de procesar miles de datos y poder dar una solución rápida y precisa.

Es un paquete fácil de usar y no sólo está destinado al etiquetado de imágenes, sino que también puede llegar a hacer otras tareas tales como la detección de caras, el reconocimiento de texto, el seguimiento de objetos en tiempo real y muchas más opciones. Pero como anteriormente se mencionó, se ha elegido el etiquetado de imágenes para realizar el reconocimiento de la ropa.

Se trata de una librería de apoyo que realiza la clasificación con el modelo básico de inteligencia artificial de Google[79] el cual diferencia más de 400 tipos de objetos de toda índole. También puede llegar a diferenciar tejidos, pero este aspecto se implementará en futuras versiones.

2.4.2 Instalación

El kit de ML de Google se puede instalar de dos formas diferentes: como un paquete que forma parte de la aplicación o como una herramienta externa que depende de los servicios de Google Play. Este paquete es necesario para poder desarrollar correctamente la aplicación.

Label index	Label text
0	Team
1	Bonfire
2	Comics
3	Himalayan
4	Iceberg
5	Bento
7	Sink
8	Toy
9	Statue
10	Cheeseburger
11	Tractor
12	Sled
13	Aquarium
14	Circus
16	Sitting
17	Beard
18	Bridge
19	Tights
20	Bird
21	Rafting
22	Park
24	Factory
25	Graduation
422	Glasses
423	Car
424	Aircraft
425	Hand
426	Rodeo
427	Canyon
428	Meal
429	Softball
430	Alcohol
431	Bride
432	Swamp
433	Pie
434	Bag
435	Joker
436	Supervillain
437	Army
438	Canoe
439	Selfie
440	Rickshaw
441	Barn
442	Archery
443	Aerospace engineering
445	Storm
446	Helmet

Figura 2.11: Lista de algunos de los objetos que puede diferenciar ML kit.

En la primera configuración, la aplicación ocupa 3MB más, pues permite que no haga falta conectarse a internet en ningún momento y puede llegar a procesar las imágenes de una forma más rápida. Esta forma se corresponde con la línea 63 del código 2.29.

La segunda forma depende de los servicios de la App Store de Google, siendo imprescindible una conexión a internet para poder utilizar la aplicación. La parte del procesado de imágenes sufre un pequeño retraso de tiempo, teniendo que esperar a los resultados, aunque ocupa menos espacio de almacenamiento del dispositivo. Este es el método que se ha elegido y se puede ver en la línea 64 del código 2.29.

Código 2.29: Build.gradle (aplicación). Instalación del kit ML de Google.

```
61      //ML
62      // Use this dependency to bundle the model with your app
63      //implementation 'com.google.mlkit:image-labeling:17.0.3'
64      implementation 'com.google.android.gms:play-services-mlkit-image-labeling:16.0.3'
```

2.4.3 Implementación

Para poder realizar la clasificación de la ropa de forma exitosa, hay que llevar a cabo una serie de pasos que facilitan el reconocimiento posterior.

En primer lugar, hay que asegurarse de que la imagen está correctamente posicionada, ya sea ayudándose del propio giroscopio del teléfono o bien con una librería adicional que indica los grados exactos de rotación de la instantánea. Una vez la aplicación tenga la foto con la inclinación de la misma, se debe almacenar en un Bitarray o Bitbuffer para después convertirla a InputImage como en el código 2.30.

Código 2.30: Clase AddActivity. Transformación de Bitarray a InputImage.

```
165          InputImage image = InputImage.fromBitmap(bitmap, 90);
```

A continuación, se crea un objeto *ImageLabeler* donde se indica el modelo que se utiliza para analizar la instantánea guardada anteriormente. En este caso, el que viene por defecto, indicado como `ImageLabelerOptions.DEFAULT_OPTIONS`.

Código 2.31: Clase AddActivity. Objeto ImageLabeler.

```
172                      ImageLabeler labeler = ImageLabeling.getClient(ImageLabelerOptions.
                             DEFAULT_OPTIONS);
```

Dicho objeto pasa a la API de Google gracias a un método donde se debe describir el procedimiento que se lleva a cabo si la operación de búsqueda de objetos ha tenido éxito o ha fracasado.

Código 2.32: Clase AddActivity. LLamada al kit.

```
174                      labeler.process(image)
```

Si ha sido exitosa, se recibirá una dupla de objetos conformada por el nombre del objeto encontrado y el grado de confianza en que, efectivamente, es ese objeto. Aunque en este caso, sólo se recoge el nombre de la prenda, como se puede observar en la línea 181 del código 2.33.

Código 2.33: Clase AddActivity. Éxito en la obtención de la clasificación.

```
175                             .addOnSuccessListener(new OnSuccessListener<List<ImageLabel>>() {
176                                 @Override
177                                 public void onSuccess(List<ImageLabel> labels) {
178                                     for (ImageLabel label : labels) {
179                                         String text = label.getText();
180                                         float confidence = label.getConfidence();
181                                         predict = translate(text);
182                                         Log.d("mlData", text + "->" + confidence);
183                                         break;
184                                     }
185                                     loadAlert.dismissDialog();
186                                     Toast.makeText(AddActivity.this, predict, Toast.
                                            LENGTH_SHORT).show();
187
188                                     //Subir foto
189                                     uploadPhotoView();
190
191                                     // Luego se muestra y se rellena el formulario
192                                     setContentView(R.layout.activity_add);
193                                     setup();//photo
194                                 }
195                             })
```

Se puede ver en el código 2.33 que hay un bucle *for*. Esto se debe a que se realiza un barrido entre las diferentes etiquetas de las que dispone el modelo de inteligencia artificial con el que se trabaja. Estas etiquetas las devuelve ordenadas de mayor a menor certeza. Por eso, se utiliza el *break* (línea 183) para poder guardar en una variable el componente del que mayor certidumbre tenga la IA. A su vez, cogido ese dato, se vuelve a filtrar traduciéndose al español en caso de ser una prenda (código 2.35).

Código 2.34: Clase AddActivity. Fracaso en la obtención de la clasificación.

```
184                             .addOnFailureListener(new OnFailureListener() {
185                                 @Override
186                                 public void onFailure(@NonNull Exception e) {
187                                     Log.d("mlData", "FAIL");
188
189                                     //Firebase Analytics event - Fail IA
190                                     FirebaseAnalytics mFirebaseAnalytics = FirebaseAnalytics.
                                            getInstance(AddActivity.this);
191                                     Bundle bundle = new Bundle();
192                                     bundle.putString(FirebaseAnalytics.Param.ITEM_ID, "IA");
193                                     bundle.putString(FirebaseAnalytics.Param.ITEM_NAME, "Fallo
                                            ");
194                                     bundle.putString(FirebaseAnalytics.Param.CONTENT_TYPE, "
                                            respuesta_IA");
195                                     mFirebaseAnalytics.logEvent(FirebaseAnalytics.Event.
                                            SELECT_CONTENT, bundle);
196
197                                     goToHome();
198                                 }
199                             });
```

En caso contrario se imprime por pantalla un mensaje con el error ocurrido y se mostrará “No encontrado” en el nombre de la prenda (código 2.34).

Código 2.35: Clase AddActivity. Función para traducir todo lo relacionado con ropa.

```
480     public String translate(String something){
481         String esp;
482 
483         switch(something){
484             case "Dress": esp = "Vestido"; break;
485             case "Swimwear": esp = "Traje de baño"; break;
486             case "Necklace": esp = "Collar"; break;
487             case "Bracelet": esp = "Pulsera"; break;
488             case "Tie": esp = "Corbata"; break;
489             case "Tuxedo": esp = "Esmoquin"; break;
490             case "Goggles": esp = "Gafas"; break;
491             case "Cap": esp = "Gorra"; break;
492             case "Hat": esp = "Sombrero"; break;
493             case "Beanie": esp = "Sombrero de punto"; break;
494             case "Jersey": esp = "Jersey"; break;
495             case "Scarf": esp = "Bufanda"; break;
496             case "Jacket": esp = "Chaqueta"; break;
497             case "Bangle": esp = "Pulsera"; break;
498             case "Jewellery": esp = "Joyería"; break;
499             case "Strap": esp = "Cinturón"; break;
500             case "Leggings": esp = "Leggings"; break;
501             case "Blazer": esp = "Chaqueta"; break;
502             case "Polo": esp = "Polo"; break;
503             case "Shoe": esp = "Zapatos"; break;
504             case "Handbag": esp = "Bolso"; break;
505             case "Sneakers": esp = "Zapatillas"; break;
506             case "Gown": esp = "Vestido de fiesta"; break;
507             case "Jeans": esp = "Pantalón vaquero"; break;
508             case "Glasses": esp = "Gafas"; break;
509             case "Bag": esp = "Mochila/Bolsa"; break;
510             case "Shorts": esp = "Pantalón"; break;
511             default: esp = "No encontrado";
512         }
513 
514         return esp;
515     }
```

De esta forma indica lo que ha detectado y lo muestra en pantalla por escrito. Si el usuario no está de acuerdo con la prenda que ha detectado, la puede modificar.

En un futuro se desea poder hacer un modelo de inteligencia artificial personalizado capaz de discernir entre mayor cantidad de prendas incluyendo colores, tejidos o incluso conjuntos complejos. Este modelo será entrenado y optimizado para que sea capaz de procesar las imágenes en un espacio de tiempo relativamente corto.

2.5 Bloque sobre las recomendaciones de atuendos

En este bloque se abarca todo lo referente a las recomendaciones que elabora la aplicación con las prendas que se tienen digitalizadas.

Se relaciona de forma indirecta con la base de datos, como se trató en el apartado 2.3.6.2.1. Realmente, trabaja con datos ya almacenados previamente en estructuras y pasados por la clase anterior. De esta forma, la información no es consultada en la base de datos en ese momento determinado, sino que se consultó con anterioridad y se obtienen los datos de la clase anterior.

En primer lugar se presentarán ciertos conceptos que es necesario conocer para entender cómo se forman los *outfits*.

2.5.1 Armonía

El concepto de armonía enlaza los sentimientos humanos con la percepción de la visión humana. Por ejemplo, si se ve un color azul, posiblemente evoque tristeza por asemejarlo al agua y a los días de lluvia.

Estos conceptos se utilizan mucho en la publicidad para poder interactuar con el cliente de forma pasiva y evocarle una sensación deseada.

2.5.1.1 Paletas de color

Los colores armónicos designan un conjunto de colores que producen un esquema o paleta de color[80] sensible al mismo sentido. Es decir, un conjunto de colores no armónicos puede llegar a hacer sentir incómodo al ser humano. Estos esquemas de colores se pueden analizar fácilmente en el círculo cromático. Se exponen algunos de ellos a continuación.

- **Monocromático.** Usa varias intensidades de un mismo color (figura 2.12).

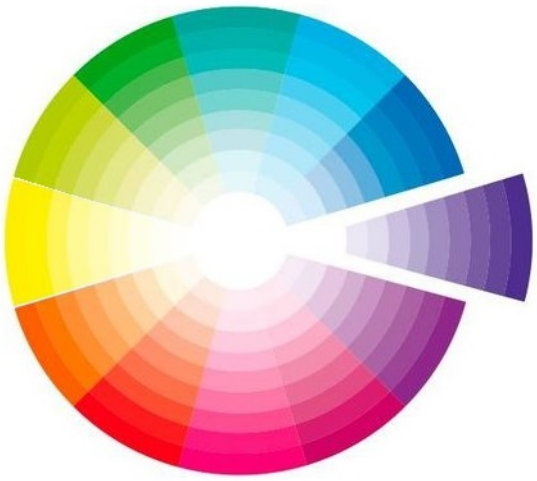

Figura 2.12: Paleta monocromática.

- **Colores complementarios.** Combina el color opuesto en el círculo. Por ejemplo, en la figura 2.13, los opuestos son el amarillo y el morado.

- **Colores análogos.** Mezcla colores vecinos del círculo cromático (figura 2.14).

Figura 2.13: Paleta complementaria[16].

Figura 2.14: Paleta análoga[17].

2.5.2 Implementación

Se parte con todos los atuendos de la base de datos del usuario en un *arraylist* y por otro lado se obtiene la fecha actual (código 2.36) transformándolo en una cadena de texto para saber la temporada, además del estilo que el usuario ha elegido previamente.

Código 2.36: Clase MixActivity. Obtención del dato de la estación.

```
73          //Get info for recommendations
74          Calendar timestamp = Calendar.getInstance();
75          SimpleDateFormat month_format = new SimpleDateFormat("MM");
76          String season = getSeason(month_format.format(timestamp.getTime()));
```

Una vez se obtienen esos dos parámetros y toda la información de la base de datos, se va filtrando como se observa en el código 2.37. Primero se filtra por temporada (línea 249) y después por estilo (línea 256) formando un array con las prendas que cumplen con los parámetros establecidos.

Código 2.37: Clase MixActivity. Filtro de la ropa.

```
244             List<String> season = prendas.get(i).getSeason(); //La estación es un array
245             List<String> style = prendas.get(i).getStyle(); //El estilo es un array
246             boolean esDelEstilo = false;
247             boolean esDeTemporada = false;
248
249             for (int j = 0; j < season.size(); j++) { //Si no tienen la misma estación, se
                    quita de la lista
250                 if (season.get(j).equals(actually_season)) {
251                     esDeTemporada = true;
252                     break;
253                 }
254             }
255
256             for (int j = 0; j < style.size(); j++) { //Si no tienen el mismo estilo, se quita
                    de la lista
257                 if (style.get(j).equals(wanted_style)){
258                     esDelEstilo = true;
259                     break;
260                 }
261             }
262
263             //Eliminar si no cumple con los requisitos
264             Log.d(TAG, "Temporada " + esDeTemporada + " y estilo " + esDelEstilo);
265             if ((!esDeTemporada) || (!esDelEstilo)) {
266                 Log.d(TAG, "ELIMINAR ---");
267                 indices.add(i);
268             }
```

Después de tener las prendas filtradas, se deben dividir en partes de arriba, partes de abajo y prendas únicas (definidas por *arrays* mostrado en el código 2.38). Las prendas únicas directamente se añaden a las recomendaciones (vestidos, monos, etc) y las prendas de arriba y de abajo, se observan en duplas comparando el dato del color que poseen y comprobando si se pueden conjuntar o no.

Código 2.38: Clase MixActivity. *Arrays* de la ropa para establecer qué tipo de parte es.

```
301         String[] nombrePrenda_unaPieza = new String[]{"Vestido", "Mono", "Vestido de fiesta"};
302         String[] nombrePrenda_piezaArriba = new String[]{"Jersey", "Polo", "Camisa", "Camiseta
                ", "Blusa", "Top", "Sudadera"};
303         String[] nombrePrenda_piezaAbajo = new String[]{"Leggings", "Pantalón vaquero", "
                Pantalón", "Falda", "Falda larga", "Falda corta"};
```

Primero se divide el color de ambas prendas en colores sin ningún adjetivo, es decir, que, por ejemplo, el color "Azul claro" pasará a llamarse "Azul". Esto se realiza mediante el método `split()` de la clase *String*.

Después, se realiza un bucle recorriendo los colores definidos en busca del nombre completo del color de la prenda de arriba. Cuando se ha encontrado este, se recorre el *array* que tiene asociado con los colores armónicos. Si lo encuentra, se puede conjuntar la prenda, y, en caso contrario, ignora ese atuendo.

Para poder mostrar al usuario los atuendos de la forma más cómoda e intuitiva posible, se ha utilizado una librería llamada *CardStackView*[81].

2.5.2.1 Librería para mostrar los atuendos: *CardStackView*

La librería *CardStackView* permite visualizar los atuendos sugeridos en forma de cartas que se pueden deslizar hacia fuera de la pantalla en cualquier dirección y tener un efecto en el transcurso de la actividad.

En este caso se ha utilizado para que se pueda desplazar la carta horizontalmente. Si se desplaza a la derecha es que no le gusta la recomendación al usuario y si se desplazas a la izquierda, se ha elegido esa recomendación y aparece un *pop up*[6] con el nombre de los elementos elegidos y un botón para poder ir al menú principal.

Para poder implementar esta librería por la parte del desarrollador, primero se deben realizar los *imports* correspondientes del código 2.39.

Código 2.39: Build.gradle (aplicación). Importar librería *CardStackView*.

```
79      //Cartas
80      implementation 'androidx.recyclerview:recyclerview:1.1.0'
81      implementation 'com.google.android.material:material:1.0.0'
82      implementation 'com.yuyakaido.android:card-stack-view:2.3.4'
```

Después, se debe disponer de una clase principal donde ajustar todos los parámetros de la librería (es muy flexible y se puede personalizar de varias maneras), un adaptador para la ventana (dibuja cada carta), y un adaptador para las cartas (tiene el control de la baza de cartas). Esto hay que implementarlo junto con las clases de las estructuras de las cartas.

2.5.2.2 Explicación detallada de la variable del color

Se ha decidido trabajar con el nombre del color y no con el valor en hexadecimal por varias razones:

- En caso de que el método `getPredominantColor()` no obtenga el color correcto, el usuario no tiene que definir uno nuevo en hexadecimal, sino que con el nombre es suficiente.
- Para poder definir los colores que se deben conjuntar, si se hiciese con el formato numérico, se tendría que hacer una aproximación primero a uno de los colores definidos y después definir un rango de colores para designar los diferentes matices de un color. De esta manera se deberían agrupar los colores por rangos y a veces no se ve de forma precisa la barrera de cuando se pasa de un pigmento a otro.
- Con este método, se aislan colores y determina fácilmente los claros, los oscuros, los pálidos, etc. De esta forma se pueden llegar a tratar de forma diferente y así crear excepciones en los conjuntos y dotar de una mayor complejidad a las recomendaciones.
- Al tener todo codificado por nombres, el mantenimiento del código y de las armonías se realiza de forma fácil y rápida.

[6] Ventana emergente que aparece por encima de la pantalla que se tiene abierta tapando parcial o totalmente el contenido de esta.

2.6 Bloque de estadísticas

A lo largo de la ejecución de la aplicación se recopila información detallada sobre el impacto que tiene en los usuarios con métricas del rendimiento y actividad. Gracias a esto es posible analizar los datos y realizar los ajustes necesarios para poder mejorar la calidad de la aplicación.

Existen eventos ya establecidos por defecto que recogen información general. Si se desea recoger una mayor cantidad de estadísticas, se deben definir otros eventos que registren esa información.

2.6.1 Instalación

Se establecen eventos en algunas clases de la aplicación cuando lleva a cabo ciertas funciones para poder recopilar métricas incluyendo las líneas 47 a la 49 del código 2.22.

La plataforma tiene algunos eventos por defecto sobre el proyecto y las acciones que realizan los usuarios, como por ejemplo *screen_view* o *first_open*.

Para añadir uno nuevo, se debe incluir el evento que se desea registrar en un *bundle*[7] con el siguiente contenido.

- `ITEM_ID`: ID del ítem almacenado. Se mostrará en la consola.
- `ITEM_NAME`: Nombre del ítem almacenado.
- `CONTENT_TYPE`: Nombre de la sección de eventos.

Una vez recopilada toda información, se envía con la referencia a *analytics* y el nombre del paquete que lo contiene.

2.6.2 Eventos registrados

Se han hecho eventos asociados a `SELECT_CONTENT` para registrar las partes más susceptibles a errores del código.

2.6.2.1 Evento acierto_IA

Se registra cada vez que no hay cambios entre la clasificación de la inteligencia artificial y lo que recoge del formulario que rellena el usuario.

Código 2.40: Clase AddActivity. Evento acierto_IA.

```
330                                         //Firebase Analytics event - fallo/acierto IA
331                                         FirebaseAnalytics mFirebaseAnalytics =
                                                FirebaseAnalytics.getInstance(AddActivity.
                                                this);
332                                         Bundle bundle = new Bundle();
333                                         if (name_s.equals(predict)) {
334                                             bundle.putString(FirebaseAnalytics.Param.
                                                    ITEM_ID, "prenda");
335                                             bundle.putString(FirebaseAnalytics.Param.
                                                    ITEM_NAME, predict);
336                                             bundle.putString(FirebaseAnalytics.Param.
                                                    CONTENT_TYPE, "acierto_IA");
```

[7]Estructura de elementos que designa un conjunto de ficheros.

El código 2.40 se sitúa en el método `addOnSuccessListener()` cuando se guarda la prenda.

2.6.2.2 Evento fallo_IA

Es una continuación a la comparación (*if*) del punto 2.6.2.1. Se ha creado este evento para saber cuándo el usuario ha cambiado el nombre de la prenda debido a que no se ha realizado una buena clasificación.

Código 2.41: Clase AddActivity. Evento fallo_IA.

```
337                                                 } else {
338                                                     bundle.putString(FirebaseAnalytics.Param.
                                                            ITEM_ID, predict);
339                                                     bundle.putString(FirebaseAnalytics.Param.
                                                            ITEM_NAME, name_s);
340                                                     bundle.putString(FirebaseAnalytics.Param.
                                                            CONTENT_TYPE, "fallo_IA");
341                                                 }
342                                                 mFirebaseAnalytics.logEvent(FirebaseAnalytics.
                                                        Event.SELECT_CONTENT, bundle);
```

El código 2.41 se sitúa en el método `addOnSuccessListener()` cuando se almacena los datos de la ropa.

2.6.2.3 Evento respuesta_IA

Este evento registra cada vez que hay un error en la lectura del modelo de inteligencia artificial para analizar la fotografía de la ropa. Las causas del error pueden ser por una señal débil de la conexión a internet.

Código 2.42: Clase AddActivity. Evento respuesta_IA.

```
189                                 //Firebase Analytics event - Fail IA
190                                 FirebaseAnalytics mFirebaseAnalytics = FirebaseAnalytics.
                                        getInstance(AddActivity.this);
191                                 Bundle bundle = new Bundle();
192                                 bundle.putString(FirebaseAnalytics.Param.ITEM_ID, "IA");
193                                 bundle.putString(FirebaseAnalytics.Param.ITEM_NAME, "Fallo
                                        ");
194                                 bundle.putString(FirebaseAnalytics.Param.CONTENT_TYPE, "
                                        respuesta_IA");
195                                 mFirebaseAnalytics.logEvent(FirebaseAnalytics.Event.
                                        SELECT_CONTENT, bundle);
```

El código 2.42 se sitúa en el método `addOnFailureListener()` cuando se ejecuta el *labeler*, como se observa en el código 2.34.

2.6.3 Análisis de los eventos

En el corto período de pruebas (menos de una semana) en el que se publicó la aplicación a un grupo reducido de personas (6 aproximadamente), se pudo recoger la siguiente información. No es representativa,

ya que, algunas métricas no estaban desde el principio y los usuarios no hicieron un uso exhaustivo de la aplicación.

2.6.3.1 Eventos predefinidos

Los eventos que vienen por defecto[82] son los contemplados en la figura 2.15.

Eventos existentes

Nombre del evento	Recuento	% de cambio	Usuarios	% de cambio
app_remove	6	↑ 500,0 %	6	↑ 500,0 %
first_open	9	↑ 800,0 %	9	↑ 800,0 %
screen_view	2.438	↑7.287,9 %	11	↑1.000,0 %
select_content	541	↑5.911,1 %	11	↑1.000,0 %
session_start	74	↑2.366,7 %	11	↑1.000,0 %

Figura 2.15: Estadísticas de todos los eventos.

Estos eventos informan de las siguientes actividades.

- *app_remove*. Número de veces que se ha desinstalado la aplicación en total.
- *first_open*. Número de veces que se ha abierto la aplicación por primera vez en un dispositivo.
- *screen_view*. Clases que más actividad tienen (detalladas en la figura 2.16).
- *select_content*. Eventos personalizados.
- *session_start*. Cantidad de usuarios que han interactuado con la aplicación.

Con estos datos se puede analizar el interés general de la gente en este tipo de aplicaciones, además de ver las clases más accedidas, lo que permite una visión de la actividad que tienen los usuarios.

Normalmente se espera una mayor actividad en el menú principal (*HomeActivity*). La segunda clase más accedida debería ser la pantalla de las sugerencias de atuendos (*MixActivity*) junto con la selección del estilo (*ChooseActivity*), ya que se tiene un número finito de prendas que registrar, pero a partir de estas, se pueden hacer un número infinito de combinaciones. Y la tercera sería añadir prendas (*AddActivity*), ya que no se suelen visualizar mucho las prendas ya digitalizadas.

2.6.3.2 Eventos personalizados

Estos eventos entran dentro de la sección `SELECT_CONTENT`. Hay eventos específicos del acceso a las diferentes clases (para poder examinar las que mayor interés tienen) y del análisis del rendimiento de la inteligencia artificial.

En este caso se ha examinado la clase de inicio de sesión (*InitScreen*), la de las sugerencias de prendas (*MixActivity*) y la de la edición de prendas (*ShowActivity*). El orden de actividad esperado es el mismo que en el de la figura 2.17 pero con diferentes porcentajes.

Interacción de los usuarios > Clase de pantalla

Clase de pantalla	% del total		Tiempo medio	
AuthActivity	41,36 %	↓ 35,6 %	0 mi...5 s	↑ 10,7 %
AddActivity	23,74 %	↑ 14,8 %	0 mi...7 s	↑ 124,9 %
HomeActivity	11,48 %	↑ 43... %	0 mi...4 s	↑ 648,9 %
ShowAllCl...Activity	6,33 %	-	0 mi...5 s	-
MixActivity	6,27 %	-	0 mi...7 s	-
ShowActivity	4,29 %	-	0 mi...8 s	-
ChooseActivity	3,25 %	-	0 mi...3 s	-
RateActivity	1,84 %	-	0 mi...9 s	-

Figura 2.16: Estadísticas de *screen_view*.

Content_type

Nombre	Recuento	%
InitScreen	466	86,14 %
MixActivity	35	6,47 %
ShowActivity	18	3,33 %

Figura 2.17: Estadísticas de los eventos personalizados sobre las clases.

Gracias los datos de la figura 2.18, se tiene la información de que se produjeron tres respuestas no exitosas de la inteligencia artificial y con respecto a las respuestas exitosas, tres aciertos y dieciséis fallos en la clasificación. Coincide con lo que cabe esperar.

fallo_IA	16	2,96 %
acierto_IA	3	0,55 %
respuesta_IA	3	0,55 %

Figura 2.18: Estadísticas de los eventos personalizados sobre la inteligencia artificial.

Los dieciséis fallos se han producido por haber detectado las siguientes prendas (figura 2.19).

Los fallos que indica en la tabla, el 87,5 % se producen por no detectar la prenda. Todavía no hay datos suficientes para evaluar la causa de ese fallo. Se puede producir a nivel de la aplicación, por no tener un modelo adecuado de inteligencia artificial o a nivel de usuario, tomando la foto de una forma incorrecta. Pero cabe esperar esa cantidad de veces que no se ha detectado nada porque el modelo utilizado de inteligencia artificial no tiene registradas todas las prendas.

2.7 Bloque de cuentas de usuario

La gestión de las cuentas de usuario se realiza mediante Firebase y es fundamental para saber en todo momento qué usuario está utilizando la aplicación y comunicar a la interfaz o al algoritmo de las recomendaciones los datos de qué usuario visualizar.

Item_id fallo_IA

Nombre	Recuento	%
No encontrado	14	87,5 %
Sombrero	1	6,25 %
Vestido	1	6,25 %

Figura 2.19: Estadísticas del evento fallo_IA.

Todas las consultas a la base de datos se hacen por usuario, pero las cuentas de usuario no se almacenan directamente en Firestore, es un módulo a parte.

Cada vez que un usuario se da de alta, se apunta en un listado accesible desde la consola de Firebase (figura 2.20).

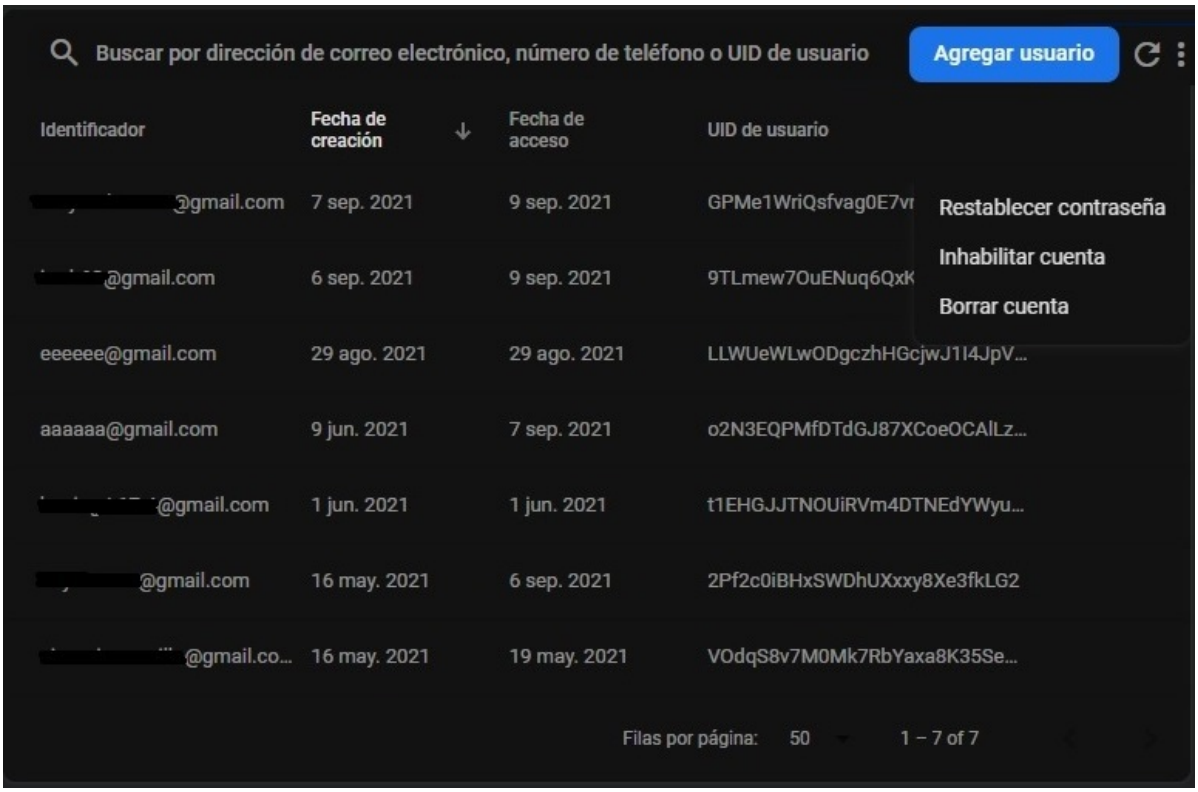

Figura 2.20: Página *Authentication* de la consola de Firebase web.

Además, es posible acceder a unas opciones básicas:

- **Copiar el UID de usuario** (cadena de caracteres que lo identifica en la base de datos).
- **Restablecer la contraseña**, para lo que se envía un correo electrónico como el de la figura 2.21.
- **Inhabilitar la cuenta.** Así se impide el ingreso del usuario a la aplicación por el tiempo que se desee
- **Borrar la cuenta.** No se eliminan la información asociada al usuario en la base de datos.

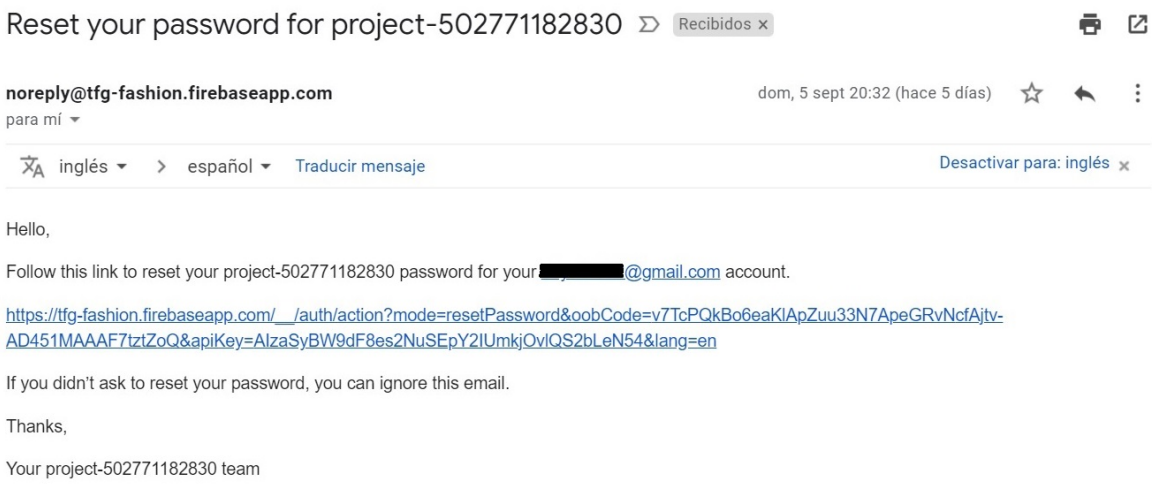

Figura 2.21: Correo de recuperación de cuenta de Firebase.

Capítulo 3

Conclusiones y líneas futuras

En este capítulo se pueden encontrar las conclusiones del desarrollo de este TFG, así como las tareas que han quedado pendientes para futuras versiones.

3.1 Conclusiones

En este proyecto se ha diseñado y desarrollado una aplicación para móviles Android para gestionar y proponer vestuario.

La aplicación realiza la propuesta de conjuntos a partir de fotografías de las prendas que el usuario ha introducido en la base de datos. Tiene como ventaja sobre sus competidores la usabilidad, ya que la introducción de las características necesarias para realizar las propuestas no tiene que ser manual: un módulo basado en inteligencia artificial averigua el tipo de prenda (pantalón, chaqueta, etc.), y el resto de las características se rellenan automáticamente a partir de la foto (color) y del tipo de prenda (ocasión y temporada). Este rasgo distintivo hace que sea mucho más fácil introducir la información y que a la hora de proponer *outfits* pueda tener en cuenta la temporada (si es verano, invierno o entretiempo), una buena coordinación de los colores y que la ocasión sea adecuada (trabajo, deporte, etc.).

La aplicación ha sido totalmente diseñada en este TFG, y en el desarrollo, además de la implementación de la interfaz de usuario, se ha incluido: la inteligencia artificial, la base de datos, se realizan estadísticas sobre el uso de las distintas partes de la aplicación y su tasa de acierto, hay un módulo de gestión de usuarios, etc.

Es un proyecto que no está finalizado, como se verá en el siguiente apartado, pero ya es totalmente funcional, habiéndose realizado una pequeña fase de evaluación con un número reducido de usuarios. La realimentación fue muy positiva, recibiendo también sugerencias que se han incorporado ya al código o al listado de tareas futuras que se presenta a continuación.

3.2 Líneas futuras

Las tecnologías que se han utilizado a lo largo de este amplio trabajo sugieren muchas formas de trabajar en el futuro. Lo que se ha desarrollado puede llegar a crecer de muchas maneras en varios ámbitos de la ciencia de una forma contundente, porque es un paso intermedio en la búsqueda de una sociedad moderna que usa la tecnología para realizar las tareas más repetitivas y tediosas del día a día y así poder tener un mayor tiempo libre más optimizado.

A lo largo de este proyecto se han pensado muchas vías para realizar una extensión del trabajo. Se pueden encontrar dos grandes grupos donde ampliarlo y por otro lado concretar aspectos más pequeños.

En referencia a los *outfits*, se podría mejorar exponencialmente si se creasen por medio de inteligencia artificial. Así captaría la esencia del usuario: la aplicación le propondría atuendos basándose en los gustos personales y de esa forma, tendría una tasa de aciertos mucho mayor. A diferencia de la versión actual, que combina los atuendos según normas generales vistas en el apartado 2.5.1. Por otro lado, también se plantea un diseño de aplicación en la que el usuario pudiese elegir los atuendos por adelantado. Por ejemplo, si tiene una fecha marcada en el calendario, podría decidir lo que ponerse días antes a la fecha y no tendría que hacer un uso excesivo diario de la aplicación. Este contenido extra, si se combina con el anterior, se lograría implementar no sólo que el cliente pudiese plantear manualmente los atuendos, sino que lo realizase de forma automática. Así, tendría la posibilidad de elegir el número de modelos por día para que, en caso de que un día saliese a correr, después a trabajar y después a pasear, entonces la aplicación debería aconsejar 3 modelos al usuario: uno de deporte, otro formal y otro casual. Otra posibilidad de expansión sería que el usuario tuviese acceso a información de sus estadísticas y lo que llevó puesto los días anteriores para que sea posible realizar un seguimiento personal de su estilo.

También, queda pendiente la gestión de la variable 'medida' en las prendas, que ahora no es utilizada, pero que en un futuro se podrá utilizar para poder ajustar y dotar de complejidad las sugerencias que genera la aplicación.

Otra idea es incluir una renderización en 3D a tiempo real de la ropa que recomienda la aplicación. De esta manera, el usuario podrá tener a su disposición un espejo de realidad aumentada para poder verse a sí mismo con la ropa puesta (figura 3.1) y no tener que estarse probando todas las combinaciones por sí mismo para visualizar cómo le quedaría.

Figura 3.1: Realidad aumentada con ropa[18].

Otra posible ampliación consiste en el establecimiento de ajustes de usuario para ofrecer una experiencia más personalizada a cada uno de ellos. Los ajustes que se deberían insertar para dar datos si se implementa la IA para las sugerencias de *outfits* serían los colores y las prendas favoritas junto con el número de recomendaciones base que se desea al día. Así, se podría favorecer hacer combinaciones con el

color o la prenda favorita de esa persona, para ofrecer una mejor experiencia al usuario y, con respecto al número de atuendos, para no tener que estar indicando todos los días el número de veces que se va a cambiar, sino establecer un número y a partir de ahí, si un día de la semana es diferente, no habría ningún problema. También se puede mejorar la experiencia de usuario ocultando alguna clasificación de las prendas para los casos en los que se tenga un uniforme del trabajo o incluso personalizar las categorías y que no se titulen formal, deporte, casual o fiesta.

La implantación de una red social será un paso importante dado el factor social de la indumentaria: compartir los *outfits* con allegados, con desconocidos e inspirar a otras personas, además de implementar la opción de buscar usuarios, chatear con ellos y ver sus prendas y sus atuendos. Por otra parte, se podría llegar a aceptar compras y así saber la procedencia de las prendas en la propia aplicación, accediendo al stock de webs de tiendas cercanas al usuario. También se contempla añadir un sistema de búsqueda de vestimenta de forma local y así ayudar a que el armario virtual esté más organizado al gusto de la persona.

Y una última actualización debería ser el refinamiento del reconocimiento del color. Hacer la lectura de tal forma que ni el material ni la iluminación fuesen componentes que inducen a error. Esto se lograría aplicar mediante filtros de balance de blancos (WB o *White Balance*)[83] para que en todo momento hiciese la detección del color verdadero. Este filtro se encargaría de ajustar los niveles de los colores básicos RGB para que la parte más brillante de la imagen se corresponda con el blanco y la que menos, el negro.

Bibliografía

[1] "Imagen logo Android," https://1000marcas.net/android-logo/ [Último acceso 29/septiembre/2021].

[2] "Imagen logo Apache," https://pnghut.com/png/mbmzDkN3NF/apache-software-foundation-react-http-server-computer-facebook-license-creative-web-material-transparent-png [Último acceso 29/septiembre/2021].

[3] "Imagen G1," https://i.blogs.es/d758f8/tmobile-g1-2/1366_2000.jpg [Último acceso 29/septiembre/2021].

[4] "Imagen Alan Turing," https://www.nationalgeographic.com.es/medio/2019/05/30/alan-mathison-turing_d8b50689.jpg [Último acceso 29/septiembre/2021].

[5] "Imagen máquina Turing," http://www.morfonet.cl/secciones/informe/ia%20imagenes/image01.gif [Último acceso 29/septiembre/2021].

[6] "Imagen Dartmouth," https://ichi.pro/assets/images/max/724/0*8MW8iP2QC_WNhmiW [Último acceso 29/septiembre/2021].

[7] "Imagen test Turing," https://i0.wp.com/www.ungeekencolombia.com/wp-content/uploads/2014/06/01-Turing-Robot.jpg?fit=600%2C450&ssl=1 [Último acceso 29/septiembre/2021].

[8] "Imagen ramas inteligencia artificial (traducida)," https://mse238blog.stanford.edu/wp-content/uploads/2017/08/AI-Graphic-NEW-768x532.jpg [Último acceso 29/septiembre/2021].

[9] "Imagen base de datos," https://cdn.statically.io/img/miro.medium.com/max/768/1*G23y70FA_6FUs1kcoJ4pIA.png [Último acceso 29/septiembre/2021].

[10] "Imagen tipos de daltonismo," https://optometristas.org/sites/default/files/daltonimso-tratamiento-asociacion-espanola-de-optometristas-unidos.png [Último acceso 29/septiembre/2021].

[11] "Imagen escala de grises," http://www.jggweb.com/wp-content/escalagrises.jpg [Último acceso 29/septiembre/2021].

[12] "Imagen espacio RGB," https://upload.wikimedia.org/wikipedia/commons/thumb/e/e0/Synthese%2B.svg/330px-Synthese%2B.svg.png [Último acceso 29/septiembre/2021].

[13] "Imagen espacio HSV," https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Triangulo_HSV.png/330px-Triangulo_HSV.png [Último acceso 29/septiembre/2021].

[14] "Imagen espacio CMYK," https://upload.wikimedia.org/wikipedia/commons/thumb/6/69/CMY%E6%B7%B7%E8%89%B2%E6%A8%A1%E5%BC%8F.png/330px-CMY%E6%B7%B7%E8%89%B2%E6%A8%A1%E5%BC%8F.png [Último acceso 29/septiembre/2021].

[15] "Imagen perfiles de color," https://i.stack.imgur.com/frFnq.jpg [Último acceso 02/octubre/2021].

[16] "Imagen paleta monocromática," https://scontent-mad1-1.xx.fbcdn.net/v/t1.6435-9/70609762_1617779581685961_8227880466833735680_n.jpg?_nc_cat=102&ccb=1-5&_nc_sid=9267fe&_nc_ohc=PcjGmXEwhuIAX85Vc-w&_nc_ht=scontent-mad1-1.xx&oh=56fa000310572fd0f79cbdc65309ac84&oe=617AE71B [Último acceso 29/septiembre/2021].

[17] "Imagen paleta análoga," https://scontent-mad1-1.xx.fbcdn.net/v/t1.6435-9/71063847_1624860567644529_2538989703944732672_n.jpg?_nc_cat=111&ccb=1-5&_nc_sid=9267fe&_nc_ohc=HQUJTZpfOdgAX_DXuG5&_nc_ht=scontent-mad1-1.xx&oh=9197d5be0e514334866c345d09b4c6f5&oe=617956CC [Último acceso 29/septiembre/2021].

[18] "Imagen vestuario en realidad aumentada," https://i0.wp.com/futuristiqsolutions.com/wp-content/uploads/2020/09/AR_Retail.jpg?w=1024&ssl=1 [Último acceso 29/septiembre/2021].

[19] "Aplicación cloth." http://www.clothapp.com/ [Último acceso 20/octubre/2020].

[20] "Aplicación wishi." https://www.wishi.me/ [Último acceso 20/octubre/2020].

[21] "Aplicación stylebook." https://apps.apple.com/es/app/stylebook/id335709058 [Último acceso 20/octubre/2020].

[22] "Aplicación stylicius." https://play.google.com/store/apps/details?id=com.fabu.stylicious [Último acceso 20/octubre/2020].

[23] "Aplicación stylicius." http://styliciousapp.com/ [Último acceso 20/octubre/2020].

[24] "Aplicación asap54." https://www.acceseo.com/asap54-encuentra-ropa-de-tu-estilo-con-una-fotografia.html [Último acceso 20/octubre/2020].

[25] "Aplicación combyne." https://play.google.com/store/apps/details?id=com.combyne.app [Último acceso 20/octubre/2020].

[26] "Aplicación qué vestir." https://play.google.com/store/apps/details?id=com.quevestir [Último acceso 20/octubre/2020].

[27] "Aplicación smart closet." https://play.google.com/store/apps/details?id=com.rkk.closet [Último acceso 20/octubre/2020].

[28] "Aplicación dress it." https://play.google.com/store/apps/details?id=com.DressIt [Último acceso 20/octubre/2020].

[29] "Aplicación 21 buttons." https://apps.apple.com/es/app/21-buttons-red-social-de-moda/id1071402389 [Último acceso 20/octubre/2020].

[30] "Aplicación iwarda." https://tuamc.tv/actualidad/decasa/a-tu-medida-ep-10-iwarda [Último acceso 20/octubre/2020].

[31] "La historia detrás de android, el sistema operativo móvil que transformó el mundo," https://www.iproup.com/coffee-break/16758-la-historia-de-android-el-sistema-operativo-que-cambio-el-mundo [Último acceso 25/agosto/2021].

[32] "¿Qué es apache? Descripción completa del servidor web Apache," https://www.hostinger.es/tutoriales/que-es-apache/ [Último acceso 25/agosto/2021].

[33] "Android cumple nueve años: así fue su primera versión y su primer móvil," https://www.xatakandroid.com/mercado/android-cumple-nueve-anos-asi-fue-su-primera-version-y-su-primer-movil [Último acceso 25/agosto/2021].

[34] "Definición de Android," https://definicion.de/android/ [Último acceso 25/agosto/2021].

[35] "Características," https://androidos.readthedocs.io/en/latest/data/caracteristicas/ [Último acceso 25/agosto/2021].

[36] "¿Qué es la inteligencia artificial y para qué sirve?" https://gestion.pe/tecnologia/inteligencia-artificial-historia-origen-funciona-aplicaciones-categorias-tipos-riesgos-nnda-nnlt-249002-noticia/?ref=gesr [Último acceso 11/agosto/2021].

[37] "Inteligencia artificial," https://www.monografias.com/trabajos82/inteligenciaartificial/inteligenciaartificial.shtml [Último acceso 11/agosto/2021].

[38] G. V. Guillén, "Aristóteles y la automatización de la lógica," *Universidad Pedagógica Nacional*, p. 18, 2001, https://revistas.udea.edu.co/index.php/estudios_de_filosofia/article/view/335673/pdf.

[39] "¿Qué aportó a la ciencia Alan Turing?" https://www.lavanguardia.com/historiayvida/historia-contemporanea/20180611/47312986353/que-aporto-a-la-ciencia-alan-turing.html [Último acceso 11/agosto/2021].

[40] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem," *London Mathematical Society*, p. 31, November 1936, https://londmathsoc.onlinelibrary.wiley.com/doi/epdf/10.1112/plms/s2-42.1.230.

[41] "Artificial intelligence (ai) coined at Dartmouth," https://250.dartmouth.edu/highlights/artificial-intelligence-ai-coined-dartmouth [Último acceso 11/agosto/2021].

[42] "Dartmouth workshop: the birthplace of ai," https://medium.com/rla-academy/dartmouth-workshop-the-birthplace-of-ai-34c533afe992 [Último acceso 11/agosto/2021].

[43] N. R. C. S. J. McCarthy, M. L. Minsky, "A proposal for the Dartmouth summer research project on artificial intelligence," *Dartmouth University*, p. 13, August 1955, http://jmc.stanford.edu/articles/dartmouth/dartmouth.pdf.

[44] "Breve historia de las redes neuronales artificiales," https://www.aprendemachinelearning.com/breve-historia-de-las-redes-neuronales-artificiales/ [Último acceso 11/agosto/2021].

[45] "Bot Eliza." http://deixilabs.com/eliza.html [Último acceso 11/agosto/2021].

[46] "¿Quién es eliza? Conoce al primer chatbot," https://soldai.com/eliza-el-primer-chatbot/ [Último acceso 11/agosto/2021].

[47] T. Winograd, "Procedure as a representation for data in a computer program for understanding natural language," Master's thesis, Massachusetts Institute of Technology, 1971.

[48] "Web oficial del proyecto," https://hci.stanford.edu/winograd/shrdlu/ [Último acceso 11/agosto/2021].

[49] "¿Eres un robot? Aprueba el test de Turing," https://umhsapiens.com/eres-un-robot-aprueba-el-test-de-turing/ [Último acceso 13/agosto/2021].

[50] "Las ramas de la inteligencia artificial(i). tipos de inteligencia artificial suave." https://www.avansis.es/inteligencia-artificial/las-ramas-de-la-inteligencia-artificial-tipos-de-inteligencia-artificial-suave/ [Último acceso 20/agosto/2021].

[51] "Conoce las 10 principales ramas de la inteligencia artificial," https://ovrmind.com/ramas-inteligencia-artificial/ [Último acceso 20/agosto/2021].

[52] "¿Qué es la visión artificial? su aplicación en robótica industrial," https://atlas-robots.com/que-es-la-vision-artificial/ [Último acceso 20/agosto/2021].

[53] "Ventajas y desventajas de la inteligencia artificial en empresas." https://nexusintegra.io/es/ventajas-y-desventajas-de-la-inteligencia-artificial/ [Último acceso 13/agosto/2021].

[54] "Ventajas y desventajas de la inteligencia artificia." https://www.winecta.com/ventajas-desventajas-inteligencia-artificial/ [Último acceso 13/agosto/2021].

[55] "Inteligencia artificial: Definición, tipos y aplicaciones," https://protecciondatos-lopd.com/empresas/inteligencia-artificial/ [Último acceso 13/agosto/2021].

[56] "Temas de base de datos," https://www.oracle.com/es/database/what-is-database/ [Último acceso 13/agosto/2021].

[57] "Bases de datos: definición, elementos y tipos," https://www.everisschool.com/blog/bases-de-datos-definicion-elementos-tipos/ [Último acceso 13/agosto/2021].

[58] "Base de datos," https://www.ticportal.es/glosario-tic/base-datos-database [Último acceso 13/agosto/2021].

[59] "Bases de datos NoSQL. Qué son y tipos que nos podemos encontrar," https://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf [Último acceso 13/agosto/2021].

[60] "Estas son las principales bases de datos nosql," https://www.nextu.com/blog/bases-datos-nosql/ [Último acceso 13/agosto/2021].

[61] "¿En qué consiste la teoría del color?" https://dical.es/blog/diseno/en-que-consiste-la-teoria-del-color [Último acceso 21/agosto/2021].

[62] "La teoría del color según Newton, Goethe, Turner y otros grandes artistas," https://www.ttamayo.com/2019/07/la-teoria-del-color/ [Último acceso 21/agosto/2021].

[63] "¿Qué tipos de daltonismo existen?" https://www.clinicagonzalezcostea.es/que-tipos-de-daltonismo-existen/ [Último acceso 21/agosto/2021].

[64] "Espacios de color," http://bibing.us.es/proyectos/abreproy/11875/fichero/Proyecto+Fin+de+Carrera%252F3.Espacios+de+color.pdf [Último acceso 21/agosto/2021].

[65] "Cómo seleccionar colores con la API de Palette," https://developer.android.com/training/material/palette-colors [Último acceso 21/agosto/2021].

[66] "Colores: Librería palette, note = https://www.sgoliver.net/blog/colores-libreria-palette/ [Último acceso 29/septiembre/2021],."

[67] "Distancia euclidiana: concepto, fórmula, cálculo, ejemplo," https://www.lifeder.com/distancia-euclidiana/ [Último acceso 21/agosto/2021].

[68] "Qué es e (delta e)," https://www.microgamma.com/calibracion_perfilacion_monitores/delta_e.php [Último acceso 21/agosto/2021].

[69] "¿Cómo sé si mi perfil de color es correcto?" http://www.jpereira.net/rough-profiler/validar-perfil-color-icc-delta-e [Último acceso 21/agosto/2021].

[70] "Documentación. descripción general de firebase," https://firebase.google.com/docs/build [Último acceso 03/septiembre/2021].

[71] "Cloud firestore," https://firebase.google.com/docs/firestore?hl=es [Último acceso 29/agosto/2021].

[72] "¿qué es una base de datos nosql? ¿cómo se estructura cloud firestore? | conozcamos cloud firestore ," https://youtu.be/v_hR4K4auoQ [Último acceso 15/agosto/2021].

[73] "How to structure your data | get to know cloud firestore 5," https://youtu.be/haMOUb3KVSo [Último acceso 15/agosto/2021].

[74] "How do queries work in cloud firestore? | get to know cloud firestore 2," https://youtu.be/Ofux_4c94FI [Último acceso 15/agosto/2021].

[75] "Agrega firebase al proyecto de android," https://firebase.google.com/docs/android/setup?hl=es [Último acceso 28/agosto/2021].

[76] "Realiza consultas simples y compuestas en cloud firestore," https://firebase.google.com/docs/firestore/query-data/queries?hl=es [Último acceso 29/agosto/2021].

[77] "Transacciones y escrituras en lotes," https://firebase.google.com/docs/firestore/manage-data/transactions?hl=es [Último acceso 28/agosto/2021].

[78] "Label images with ml kit on android," https://developers.google.com/ml-kit/vision/image-labeling/android#java [Último acceso 18/mayo/2021].

[79] "Ml kit image labeling: labels for default model," https://developers.google.com/ml-kit/vision/image-labeling/label-map [Último acceso 18/mayo/2021].

[80] "Los esquemas de colores," https://s3.accesoperu.com/wp6/includes/htmlarea/mezclador/ayuda/ec.htm [Último acceso 21/agosto/2021].

[81] "Cardstackview. tinder like swipeable card view for android," https://github.com/yuyakaido/CardStackView/blob/master/README.md [Último acceso 29/agosto/2021].

[82] "[ga4] eventos recopilados automáticamente," https://support.google.com/firebase/answer/9234069?hl=es-419 [Último acceso 19/septiembre/2021].

[83] "Balance de blancos: Qué es y cómo se usa," https://www.dzoom.org.es/el-balance-de-blancos/ [Último acceso 21/agosto/2021].

[84] "Conoce las fases de un proyecto de desarrollo de software," https://www.occamagenciadigital.com/blog/conoce-las-fases-de-un-proyecto-de-desarrollo-de-software [Último acceso 12/septiembre/2021].

[85] "Planes de precios," https://firebase.google.com/pricing?hl=es [Último acceso 12/septiembre/2021].

Apéndice A

Pliego de condiciones

El trabajo de este proyecto se ha realizado con una serie de elementos necesarios para su desempeño. Se describen a continuación.

Requerimientos de *hardware*

- Ordenador con los siguientes requerimientos mínimos:

  Arquitectura de 64 bits, segunda generación de Intel Core o una AMD con soporte de *Windows Hypervisor*

  8GB de RAM

  8GB de espacio libre de memoria

  Resolución de 1280 x 800

- Móvil con los siguientes requerimientos mínimos:

  Android 6.0

  2GB de RAM

  60MB de espacio libre de memoria

  Conexión a internet

Requerimientos de *software*

- Computadora con sistema operativo Windows 8/10, MacOS 10.14 o Linux
- Entorno de desarrollo Android Studio 4.1
- Navegador para acceder a la consola del proyecto

Apéndice B

Presupuesto

Se enseñan los diferentes costes que se deben asumir para implementar este sistema. Se dividen en costes de personal, *hardware* y *software*.

B.1 Coste de mano de obra

Este apartado comprende los gastos del personal para desarrollar el proyecto. Las tareas que se deben realizar son las siguientes[84].

1. Análisis del sistema y requisitos. Se analizan los aspectos que necesita el proyecto.
2. Diseño de la arquitectura. Se plantea el funcionamiento general del sistema.
3. Diseño de la base de datos. Se forma la estructura que debe cumplir la base de datos.
4. Desarrollo del código. En este paso se lleva a cabo la escritura del código.
5. Pruebas. Se llevan a cabo evaluaciones de posibles fallos e inconformidades entre los usuarios y el cliente.
6. Mantenimiento y cuidado. Se realizan actualizaciones para mantener el código seguro y sin vulnerabilidades.
7. Documentación. Se hace una documentación del proyecto que contiene la explicación detallada de sus componentes y cómo está hecho.
8. Diseño de usabilidad. Con estudios psicológicos y de ergonomía, se elabora un sistema fácil de utilizar.

Estas tareas están asociadas a cuatro perfiles profesionales con diferentes costes como se observa en la tabla B.1. Todos los datos son aproximados.

1. Analista programador. Combinación de dos roles. Se encarga de llegar a un acuerdo con el cliente sobre lo que desea y sintetizarlo en una base de datos de forma coherente.
2. Programador de Android. Especialista en programar aplicaciones en Android.
3. Analista de datos. Determina qué datos debe contener la base de datos además de estructurarla.

Tabla B.1: Costes de personal.

Puesto	Horas	Coste / hora (€)	Coste total (€)
Analista programador	100	60,34	6034
Programador Android	500	60,02	18006
Analista de datos (*Data scientist*)	200	63,21	12642
Administrador de datos	300	64,52	19356
Total			56038

Tabla B.2: Horas totales disponibles.

Duración del proyecto (meses)	4
Días laborales / mes	20
Horas / día	8
Horas totales	640

4. Administrador de datos. Mantiene la seguridad de la base de datos y conoce cómo se utilizan los datos en los diferentes procesos de la empresa.

En estos costes se tiene en cuenta el tiempo empleado en realizar el proyecto, prácticamente unos cuatro meses (tabla B.2).

B.2 Coste *software*

Estos costes recogen los gastos de licencias. Puesto que Android es un sistema *Open Source*, la plataforma y el entorno de programación no requieren costes adicionales. Además, para minimizarlos aún más, se desarrollará en Linux, ya que el IDE es compatible con él.

El servicio de Firebase (que contiene ML kit) ofrece varios planes[85], entre ellos, se ha elegido el gratuito donde los pagos se realizan mensualmente sólo si se llega a un umbral (20000 operaciones de lectura y 50000 de escritura al día).

B.3 Coste *hardware*

Los costes de *hardware* abarcan las máquinas y los dispositivos necesarios para poder desarrollar este proyecto. Para minimizar costes, en este caso, en vez de utilizar ordenador de sobremesa, se ha optado por un portátil a precio asequible. Se pueden observar los gastos desplegados en la tabla B.3.

B.4 Coste total

Por último, se va a hacer un recuento de todos los gastos contemplados anteriormente junto con gastos variables detallados en la tabla B.4.

Tabla B.3: Costes de *hardware*.

Producto	Coste (€) / ud.	Coste total (€)
Ordenador portátil	350	1750
Móvil	150	750
Cable micro USB/tipo C	4	20
Total		504

Tabla B.4: Costes totales.

Descripción	Coste total (€)
Costes de mano de obra	56038
Costes de *hardware*	504
Costes de *software*	0
Costes extra	400
Gastos generales	0
IVA	11957,82
Total	68899,82

- Costes extra. Se asocian a gastos imprevistos durante el desarrollo del proyecto como el mantenimiento del *hardware*.
- Gastos generales. Se incluyen gastos de propósito general como los asociados a la oficina o al desplazamiento. Para abaratar costes, no se tendrá una oficina, sino que se teletrabajará.
- IVA. Es el impuesto sobre el valor añadido. En España es un 21 % del valor total del producto.

La inversión ascendería hasta los 68900 € aproximadamente.

Apéndice C

Manual de usuario

C.1 Introducción

Este es un manual de usuario general que tiene como objetivo explicar el funcionamiento de la aplicación desarrollada. No se necesita ningún conocimiento para seguir esta guía.

Se corresponde con la versión 0.0.3 de la aplicación (última versión).

Para poder instalarla, se debe tener el archivo APK con la última versión de la aplicación y hacer clic sobre él para que Android la instale.

C.2 Manual

C.2.1 Registrarse o iniciar sesión

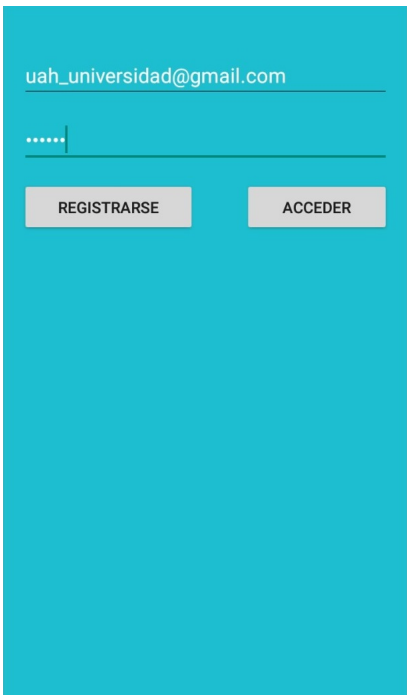

Figura C.1: Pantalla para iniciar sesión o registrarse.

La pantalla que aparece al abrir la aplicación es la de la figura C.1.

Se puede entrar de dos formas. En caso de que sea un usuario nuevo, se debe registrar escribiendo una dirección de correo y la contraseña para la aplicación. Si ya se ha registrado, tiene que presionar el botón "Acceder".

C.2.2 Menú principal

Una vez en la pantalla principal, si el usuario es nuevo, debe pulsar "Añadir prendas" para registrar la ropa del armario.

Figura C.2: Pantalla principal de un usuario nuevo.

C.2.3 Añadir prendas

En esta pantalla se muestra la vista de la cámara y hay que hacer una foto a la prenda que se desee digitalizar, preferiblemente en un fondo uniforme en el que la prenda destaque y se vea lo más cerca posible. Cuando el usuario tenga la prenda centrada y cuadriculada, se debe presionar el botón "Detectar".

Si surge algún problema, como que la aplicación no pueda establecer una conexión con la inteligencia artificial, la foto no se guarda y se retrocede al menú principal (figura C.2). Por el contrario, si no hay ningún imprevisto (ha podido establecer la conexión satisfactoriamente), aparece la pantalla de la figura C.4.

En esta pantalla se indican los datos de la ropa detectada. En caso de que algún dato esté mal introducido, se puede corregir manualmente. Una vez insertados todos los datos, se hace clic en "Guardar" e y la aplicación muestra por pantalla el resultado de la operación (si ha tenido éxito al insertar la prenda en la base de datos o ha fracasado), mientras vuelve a la pantalla principal (figura C.2).

Figura C.3: Visualización de la cámara para digitalizar la prenda.

Figura C.4: Insertar prenda.

C.2.4 Visualizar y editar prendas

Para tener una vista general de todas las prendas que se han introducido en la aplicación, desde el menú principal (figura C.2) se debe presionar el botón “Ver mis prendas”. Si no se ha digitalizado ningún tipo

de ropa, muestra la pantalla en blanco, pero si se han insertado prendas, aparecen las instantáneas con los correspondientes nombres de cada una (figura C.5).

Figura C.5: Listado de ropa.

Si se selecciona una prenda, se muestran todos los datos introducidos para ella (figura C.6) y se permite modificar toda la información menos la foto. También es posible eliminar la prenda presionando el botón con el símbolo de una papelera. Para volver a la pantalla principal pulsa el botón de atrás del dispositivo donde está instalada la aplicación.

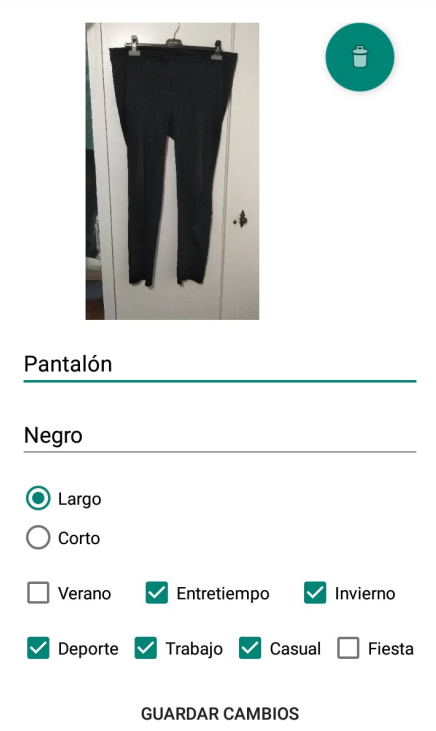

Figura C.6: Detalles de la prenda seleccionada.

C.2.5 Sugerencias de atuendos

La pantalla principal, una vez que se ha introducido un número de prendas suficiente como para hacer combinaciones satisfactorias (se necesitan partes de arriba y de abajo que concuerden con el estilo y la temporada), tiene el aspecto de la figura C.7. Se diferencia de la figura C.2 porque al acceso al botón Recomiéndame un look.[es]tá habilitado.

Figura C.7: Pantalla principal de un usuario con prendas digitalizadas.

A continuación, se da clic al botón "Recomiéndame un *look*", lo que redirige a la pantalla de la figura C.8.

Figura C.8: Pantalla de elección del estilo.

En esta pantalla hay que elegir entre los cuatro estilos disponibles, que son trabajo, deporte, casual y fiesta. Cuando se selecciona, aparece la siguiente pantalla (figura C.9) en la que, si no se encuentran prendas con el filtro indicado, se muestra la página con una figura que lo indica. Pero en cambio, si se tiene al menos un atuendo con ese criterio, se forman las recomendaciones y se ven en la pantalla de la figura C.9.

Figura C.9: Pantalla de elección del atuendo.

En esta interfaz se permite interactuar con la ropa y decidir si se desea elegir el atuendo recomendado, en cuyo caso hay que desplazar la imagen a la izquierda, o no, que entonces hay que desplazar la imagen a la derecha (figura C.10).

Figura C.10: Pantalla de elección del atuendo. A la izquierda se acepta y a la derecha de rechaza la prenda.

Si se rechaza una prenda, se pasa a la siguiente imagen. Si se acepta, entonces aparece una pantalla tipo *popup* (figura C.11) donde se puede ver el nombre de cada prenda y volver a la pantalla de inicio si se da clic en "Volver". Está hecho en tandas de cinco sugerencias.

Figura C.11: Pantalla de confirmación de elección del atuendo.

C.2.6 Valorar la aplicación

En la pantalla principal (figuras C.7 o C.2) existe la opción "Deja un comentario". Si se selecciona, muestra la pantalla de la figura C.12, donde se puede puntuar la aplicación y dejar un comentario (opcional) indicando posibles mejoras, lo que ha gustado o no, etc.

Al presionar el botón "Enviar", se vuelve al menú principal (figuras C.7 o C.2) con un mensaje de agradecimiento por haber comentado la aplicación.

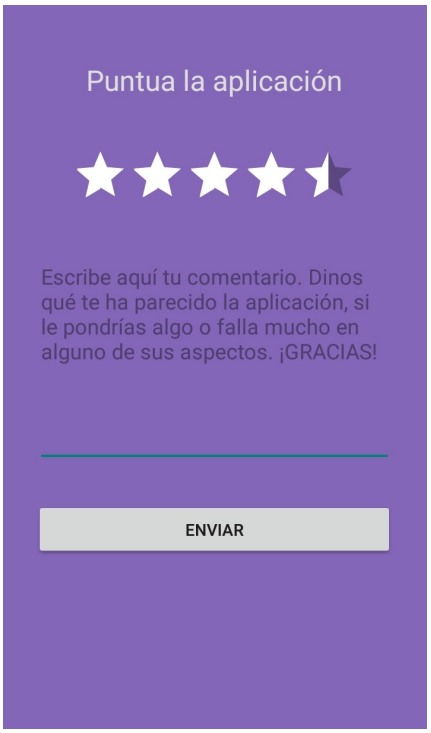

Figura C.12: Pantalla de valoración.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá