UNIVERSIDAD
DE GRANADA

Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones

Programación de dispositivos móviles: App 3

What2wear

José Alberto García Collado
26513007-X
joseegc10@correo.ugr.es

2020-2021

Índice

1 Explicación de la app

What2wear es una app orientada a que el usuario tenga las prendas de su armario recogidas en la app y pueda crear conjuntos con ellas, así como compartir por redes sociales la descripción y la imagen de una prenda y las imágenes que componen un conjunto, además de poder guardarlas en la librería de su dispositivo.

En cuanto a las búsquedas disponibles, el usuario puede seleccionar las prendas que pertenecen a un determinado tipo. Existen tres tipos generales, que son ropa, complementos y calzado. Dentro de la ropa encontramos diferentes subtipos como pantalones, camisetas, sudaderas, chaquetas, entre muchos otros. Además también podemos consultar los diferentes conjuntos que hemos creado, pudiéndose realizar una búsqueda por nombre de dichos conjuntos.

También comentar que se ha generado un diagrama de clases con la ayuda de la herramienta obtenida en el siguiente **enlace**. Esta me ha generado automáticamente el diagrama de clases permitiendo hacer los cambios que se requieran. El diagrama de clases se puede encontrar en la carpeta classDiagram.

2 Componentes de la app

La app se ha construido para el sistema operativo ios, a partir de la biblioteca de Apple SwiftUI, siguiendo el paradigma recomendado por ellos, el MVVM (Model - View - ViewModel). La información del usuario se almacena en la base de datos Firestore de Firebase, pues en un futuro se tiene pensado crear una red social de este aplicación para que un usuario pueda ver las prendas y conjuntos de otros usuarios.

2.1 Model

En el modelo encontramos las estructuras de datos en las que se va a pasar la aplicación. En concreto, vamos a encontrar tres:

- Una estructura para la prenda de ropa, que va a contener el id, la descripción de la prenda, el tipo de prenda, el tipo de ropa (en caso de que el tipo de prenda sea ropa este campo va a tener el tipo de ropa), la dirección donde esta almacenada la imagen de la prenda en el Storage y el id del usuario que ha creado la prenda.
- Una estructura que representa un usuario, la cual contiene el id del usuario, su nombre, sus apellidos, su correo y la dirección en el Storage donde se encuentra la imagen de usuario. Esta estructura actualmente no se usa para nada en la aplicación, pero se encuentra ya que como se ha mencionado antes se tiene pensado construir una red social en base a esta aplicación.

- Una estructura que representa un conjunto, el cual contiene el id del conjunto, el nombre del conjunto, un array con los ids de las prendas que pertenecen al conjunto y la dirección donde se encuentra la imagen del conjunto en el Storage.

Además, tenemos dos enumerados. El primero es para los tipos generales de prendas, los cuales son:

- Ropa
- Calzado
- Complementos

También tenemos otro enumerado que representa los tipos de ropa que se admiten, los cuales son:

- Pantalones
- Camisetas
- Sudaderas
- Chaquetas
- Camisas
- Ropa interior
- Trajes
- Jerseys
- Ropa de baño
- Pijamas
- Faldas
- Vestidos

2.2 View

En cuanto a las vistas, como son demasiadas, menciono las carpetas en las que se ha dividido.

La primera es **Principal**. Esta carpeta contiene las vistas principales de la app, como es la vista de introducción y la vista principal de inicio.

La segunda es **Extra**. Esta carpeta contiene vistas que son partes pequeñas de otras vistas y que se utilizan en varias, como puede ser la vista de imagen obtenida a partir del Storage de Firebase, una vista de navegación hecha por mí, una vista que llama a la vista principal pero eliminando la barra de navegación, etc.

La tercera es **Prenda**. Esta contiene todo lo relacionado con las prendas, como es la vista que representa una carta de prenda, la vista de selección de los tipos de prendas, la vista de creación y edición de una prenda, etc.

La cuarta es **Conjunto**. Esta contiene todas las vistas que están relacionadas con los conjuntos. Estas son la de añadir un conjunto, la de guardar un conjunto una vez se ha seleccionado todas las prendas, la de buscar un conjunto a partir del nombre, la de ver un conjunto y la de la carta que representa un conjunto, entre otras.

Por último, comentar que hay un archivo de vista en el cual me he basado en código encontrado por internet. El enlace a la página donde me he basado se puede encontrar en el código, al principio del todo en los comentarios iniciales. Este archivo es NavigationBarColor y se puede encontrar en la carpeta Extra. Básicamente consiste en un modificador de vista, permitiéndome modificar el color de la barra de navegación, la cual SwiftUI por defecto solo nos deja poner en blanco.

2.3 ViewModel

Por último, tenemos las clases que nos conectan las estructuras de datos con las vistas, aportando la funcionalidad necesaria.

En primer lugar, nos encontramos con dos archivos en los cuales no se ha hecho uso de la biblioteca SwiftUI, sino de la biblioteca UIKit. Esto es debido a que SwiftUI es una biblioteca muy reciente y no tiene incorporadas todas las funcionalidades todavía.

El primer archivo se trata de **ShareSheet** y este se encarga de permitir que el usuario pueda compartir todo tipo de items por los diferentes canales que existen (redes sociales, su proia galería, etc). Este archivo tiene que pertenecer al protocolo UIViewControllerRepresentable, por lo que este protocolo exige la creación de dos métodos. El primero es makeUIViewController, en el cual vamos a devolver un UIActivityViewController inicializado con los items que le pasamos a ShareSheet cuando creamos un objeto de la clase. El segundo es updateUIViewController, el cual podemos dejar vacío ya que no necesitamos hacer nada al actualizarse la vista.

El segundo archivo se trata de **ImagePicker** y este se encarga de proporcionar la selección de la imagen para las prendas y para el conjunto. En este caso, este archivo tiene una estructura muy fija, por lo que se ha obtenido con la ayuda de internet al ser siempre igual.

Posteriormente, tenemos un enumerado que representa el estado del usuario, es decir, si se encuentra en la vista de introducción o en la vista principal, pudiendo añadir más valores posibles en el futuro cuando se construya la red social.

La clase **DocumentViewModel** se encarga de realizar la conexión con el Storage de Firebase para obtener las imágenes. Esta contiene dos atributos, el primero para almacenar la imagen en si y el segundo para almacenar la dirección a la imagen en el Storage y el cual se recibe como parámetro en el constructor. Esta clase tiene un método load, el cual se encarga de hacer la petición al Storage del documento almacenado en la dirección que nos indica el atributo y de guardarlo en nuestra variable data para que pueda ser accedido por otras vistas.

2.3.1 PrendaViewModel

Por último, nos encontramos con la clase más importante, **PrendaViewModel**. Esta clase se encarga de realizar la conexión con la base de datos, ya sea para guardar prendas (y obtenerlas), o para crear conjuntos con estas prendas (y obtenerlos).

Los atributos de esta clase son:

- user: Para almacenar la información del usuario.
- prendas: Para almacenar una serie de prendas.
- conjuntos: Para almacenar una serie de conjuntos.

- prendasConjunto: Diccionario cuya clave es el id de un conjunto y que contiene la lista de prendas del conjunto.
- imagenes: Para almacenar una lista de imágenes.
- tiposRopa: Para almacenar los tipos de ropa que usa el usuario.

El primer método es **addPrenda**, el cual recibe el objeto del modelo PrendaModel y la imagen de la prenda. En primer lugar añade la imagen al Storage y una vez se ha añadido se añade la información de la prenda a Firestore.

El segundo método es **getTiposRopa**. En nuestra base de datos almacenamos los tipos de ropa que el usuario tiene en la aplicación. Esto es para que cuando se crea un conjunto solo se le muestre al usuario que seleccione entre los tipos de ropa que tiene. Por tanto, este método accede a la colección tiposRopa y obtiene los tipos del ropa del usuario.

Ahora tenemos dos métodos llamados **docEnPrenda**. Cambia lo que reciben como parámetro, el primero recibe un objeto QueryDocumentSnapshot el cual contiene dos atributos, el diccionario y el id. El segundo, recibe directamente el diccionario y el id. Estos dos son necesarios ya que en función de la llamada a Firestore recibimos una u otra cosa.

Después, nos encontramos con dos métodos para editar una prenda. El primero, **editPrenda**, recibe solo la prenda a modificar, actualizando la información de la prenda cuyo id recibimos dentro del objeto con la información restante en el objeto. El segundo, **editPrendaWithImage**, recibe además la nueva imagen de la prenda, actualizándola primero y actualizando el resto de información después.

El siguiente método es **deletePrenda**, el cual recibe la prenda a eliminar. En primer lugar, eliminamos el documento cuyo id recibimos y en segundo lugar eliminamos la imagen en el Storage a partir de la dirección que recibimos dentro del objeto.

El penúltimo método referente a las prendas es **busquedaPrendas**, el cual busca las prendas que son del tipo que recibimos como parámetro.

El último método referente a las prendas es **busquedaPrendasRopa**, el cual busca la ropa que es del tipo que recibimos como parámetro, es decir, la petición es la siguiente: quiero aquellas prendas que son del tipo ropa y además el tipo de ropa es igual al que recibimos como parámetro.

Ahora nos encontramos con los métodos referentes a los conjuntos. El primero es **addConjunto** y recibe la lista de prendas del conjunto y el nombre del conjunto, añadiendo esta información a la base de datos.

Después tenemos el método **docEnConjunto**, el cual recibe un diccionario y un id y crea el objeto del modelo ConjuntoModel, devolviéndolo.

El siguiente método es **getConjuntos** y obtiene todos los conjuntos que son del usuario actual.

Más tarde encontramos al método **getPrendasConjunto**, el cual recibe el id del conjunto y en primer lugar lo obtiene de la base de datos. Después, recorre los ids de las prendas que están almacenadas en el conjunto y va añadiéndolas al atributo prendasConjunto, el cual se explica arriba.

Ahora tenemos el método **busquedaConjuntos**, el cual recibe la cadena que debe contener el nombre del conjunto. Este hace una petición de todos los conjuntos del usuario y se queda con aquellos cuyo nombre contiene la cadena buscada.

Después está el método **editConjuntoImage**, el cual recibe el conjunto y su nueva imagen y se encarga de modificar la imagen actual que tiene el conjunto o de añadirla en caso de que no tenga ninguna.

El penúltimo método es **getImagenes** y recibe una lista de direcciones que son las direcciones de las prendas de un conjunto en el Storage. Este método recorre todas las direcciones y obtiene las imágenes, almacenándolas en el atributo imagenes.

Por último, tenemos el método **deleteConjunto** que recibe el id del conjunto y la dirección donde está almacenada la imagen del conjunto en el Storage, borrando tanto la imagen como la información del conjunto.