**FakeStore App – Documentación.**

12/10/2024

Mauricio Alzate Ocampo.

Documentación técnica proyecto FakeStore, una aplicación desarrollada con Jetpack Compose para visualizar productos de una tienda en línea usando la FakeStore API.

**Introducción.**

Con esta documentación tenemos el objetivo de proporcionar a los desarrolladores una guía técnica detallada para entender, modificar y mantener el proyecto FakeStore.

**FakeStore es una aplicación que permite a los usuarios:**

* Navegar por productos destacados y categorías específicas.
* Ver detalles individuales de los productos.
* Agregar productos al carrito de comprar.

**Tecnologías principales:**

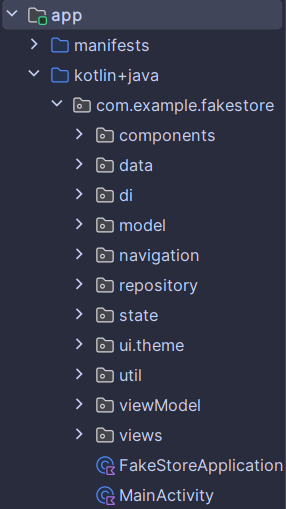
* Jetpack Compose.
* Arquitectura MVVM.
* RetroFit.
* Hilt.

**Arquitectura del proyecto:**

Se utiliza la arquitectura MVVM (Model-View-ViewModel) para una separación clara de las responsabilidades de la aplicación.

* Model: Gestión de datos (FakeStore API).
* ViewModel: Lógica de negocio y gestión de estados.
* View: Interfaz de usuario usando Jetpack Compose.

**Estructura de carpetas:**

****

* Components: Composables reutilizables.
* Data: Modelos de datos y clases relacionadas.
* Di: Configuración de Hilt para inyección de dependencias.
* Model: Definición de entidades como ProductsModel.
* Navigation: Gestión de la navegación entre pantallas.
* Repository: Lógica para consumir datos desde la FakeStore API.
* State: Gestión del estado global de la aplicación.
* Útil: Utilidades comunes como ENDPOINTS.
* ViewModel: Clases ViewModel para cada pantalla.
* Views: Pantallas principales de la aplicación.

**Dependencias:**

* RetroFit: Consumo de APIs REST.
* Hilt: Inyección de dependencias.
* Coil: Carga de imágenes.
* Kotlinx Coroutines: Manejo de operaciones asíncronas.

**Funcionalidades principales:**

* Listar productos por categorías.
* Detalles de productos.
* Carrito de compras.
* Sesión de usuario con LogOut.

**Documentación de carpetas:**

Components:

A screenshot of a computer

Description automatically generated

Esta carpeta contiene componentes reutilizables de UI. En este caso tenemos archivos para Cateogories, Products y User.

**CategoriesComponents:**

El componente CardCategories es una tarjeta que muestra el nombre de una categoría. Esta diseñado para ser reutilizable y permitir hacer click sobre la tarjeta para realizar acciones relacionadas con la categoría seleccionada.

****

**categoryName** representa el nombre de la categoría que se mostrará dentro de la tarjeta.

**onCategoryName** es una función que se ejecuta cuando el usuario da click en la tarjeta. En este caso mostrar los productos de la categoría correspondiente.

La tarjeta tiene esquinas redondeadas con un radio de 16.dp. Utiliza colores extraídos del esquema de Matherial Design.

**ProductsComponents:**

Para el caso de ProductsComponents ya tenemos más componentes reutilizables que está diseñados de igual manera para ser reutilizables en diferentes pantallas de la aplicación.

**MainTopBar:**

Es un componente que representa una barra superior para mostrar el titulo de la pantalla y proporcionar opciones de navegación.



**Title** define el texto que se mostrará como titulo en la top bar.

**showBackButton** define si se debe mostrar el botón para retroceder en la barra dependiendo de la pantalla.

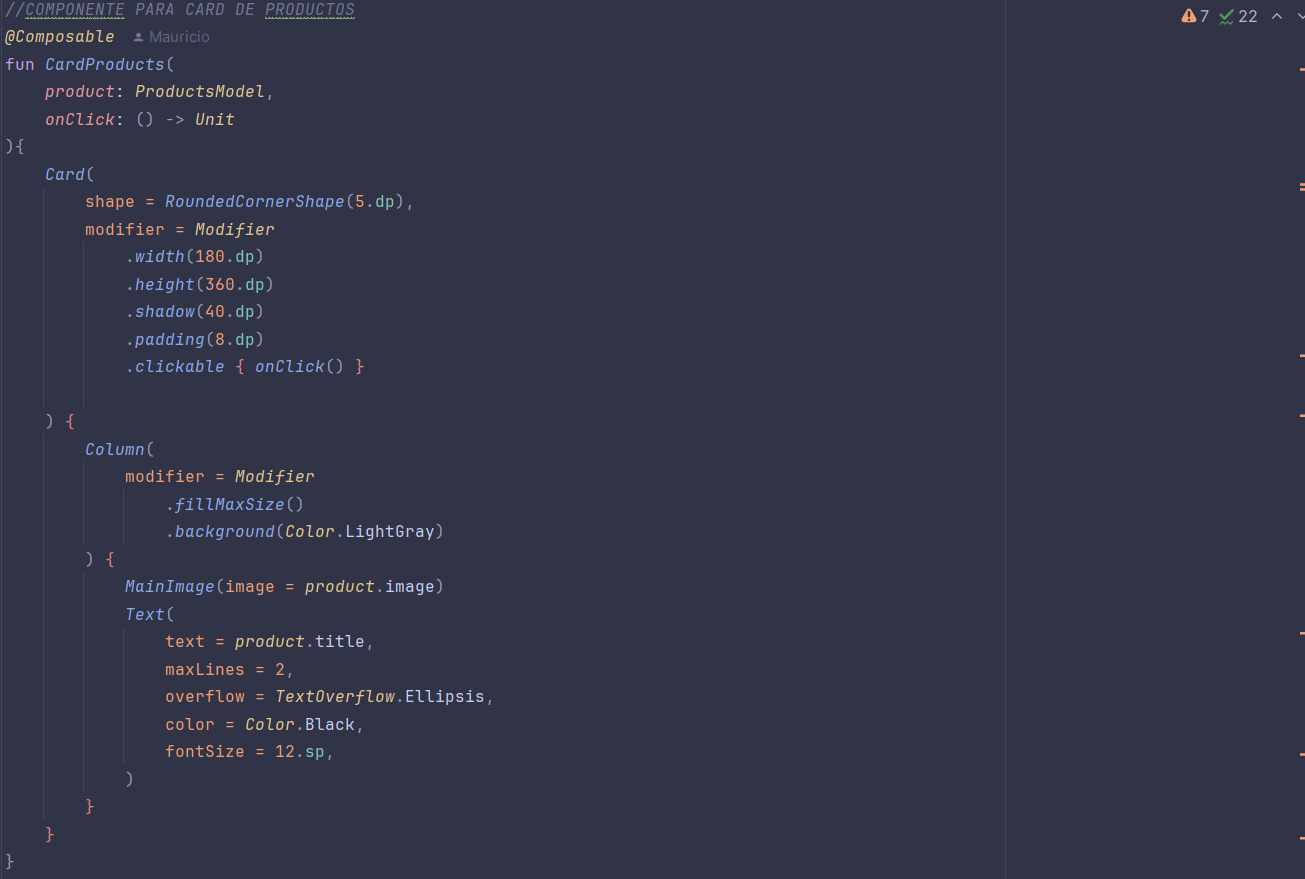
**onClickBackButton** función que se ejecuta al presionar el botón de retroceso. Es opcional dependiendo de la pantalla.

**onMenuClick** función que se ejecuta al presionar el botón de menú (icono de tres líneas horizontales)

**onClickCart** función que se ejecuta al presionar el icono del carrito de compras.

**CardProducts:**

El componente CardProducts muestra la información básica de un producto en forma de tarjeta. Este diseño incluye una imagen destacada del producto y su título. Al hacer click en la tarjeta permite navegar a los detalles del producto



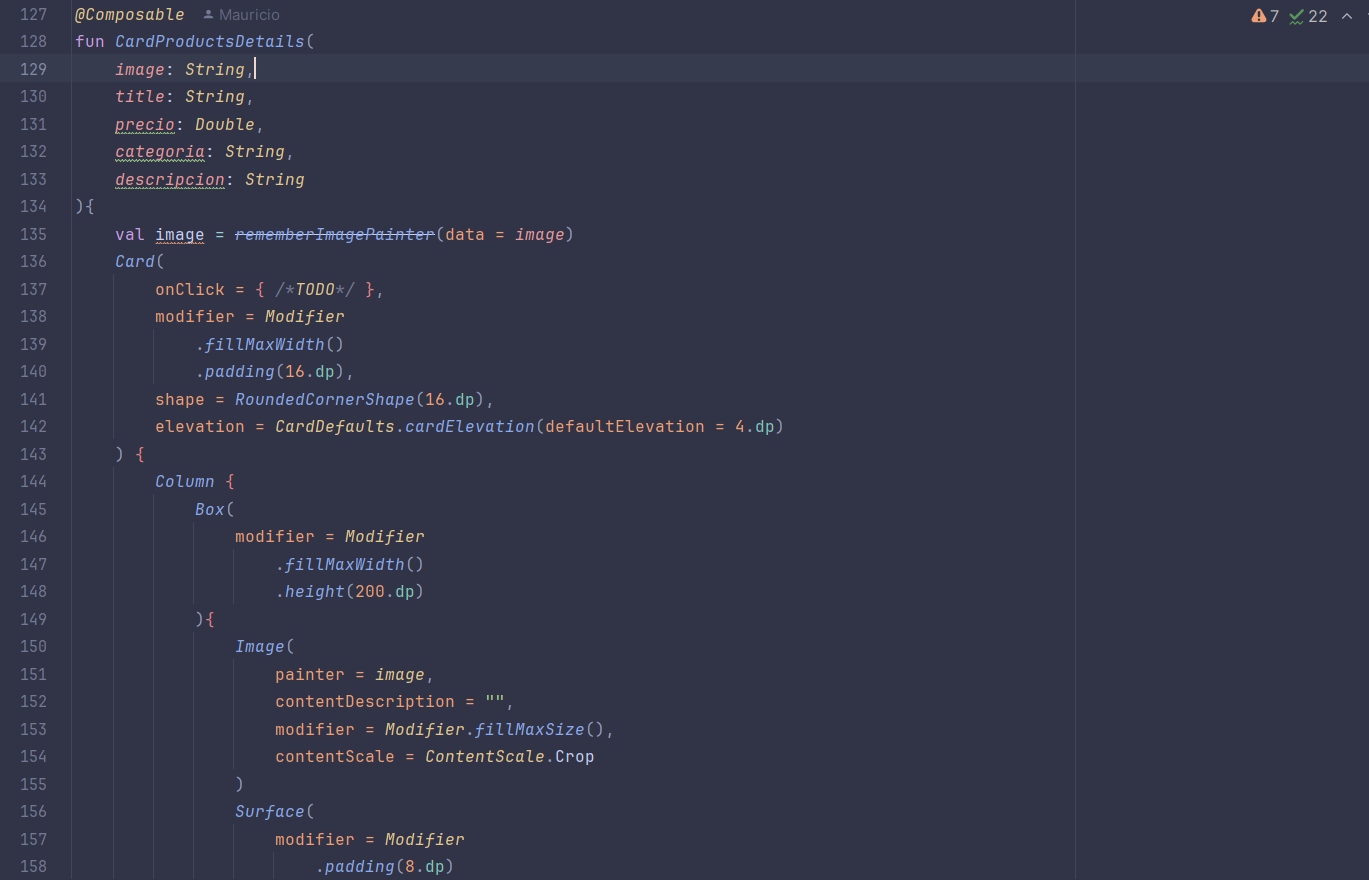
**Products** contiene los datos del producto, incluidos la imagen y el título.

**onClick** define la acción que hace la tarjeta al hacer click en ella.

La imagen de muestra en la parte superior mediante el componente MainImage. El texto aparece debajo de la imagen, adaptándose al tamaño.

**CardProductsDetails:**

El componente CardProductsDetails está diseñado para mostrar la información detallada de un producto en un formato de tarjeta. Podemos ver datos como la imagen, el título, el precio, la categoría y la descripción, además de incluir un botón para añadir el producto al carrito.



A screenshot of a computer program

Description automatically generatedA screen shot of a computer program

Description automatically generated

**MainImage:**

El componente MainImage tiene como objetivo mostrar una imagen en un diseño especifico, adaptándola al ancho del contenedor. Nos sirve para presentar la imágenes de los productos.

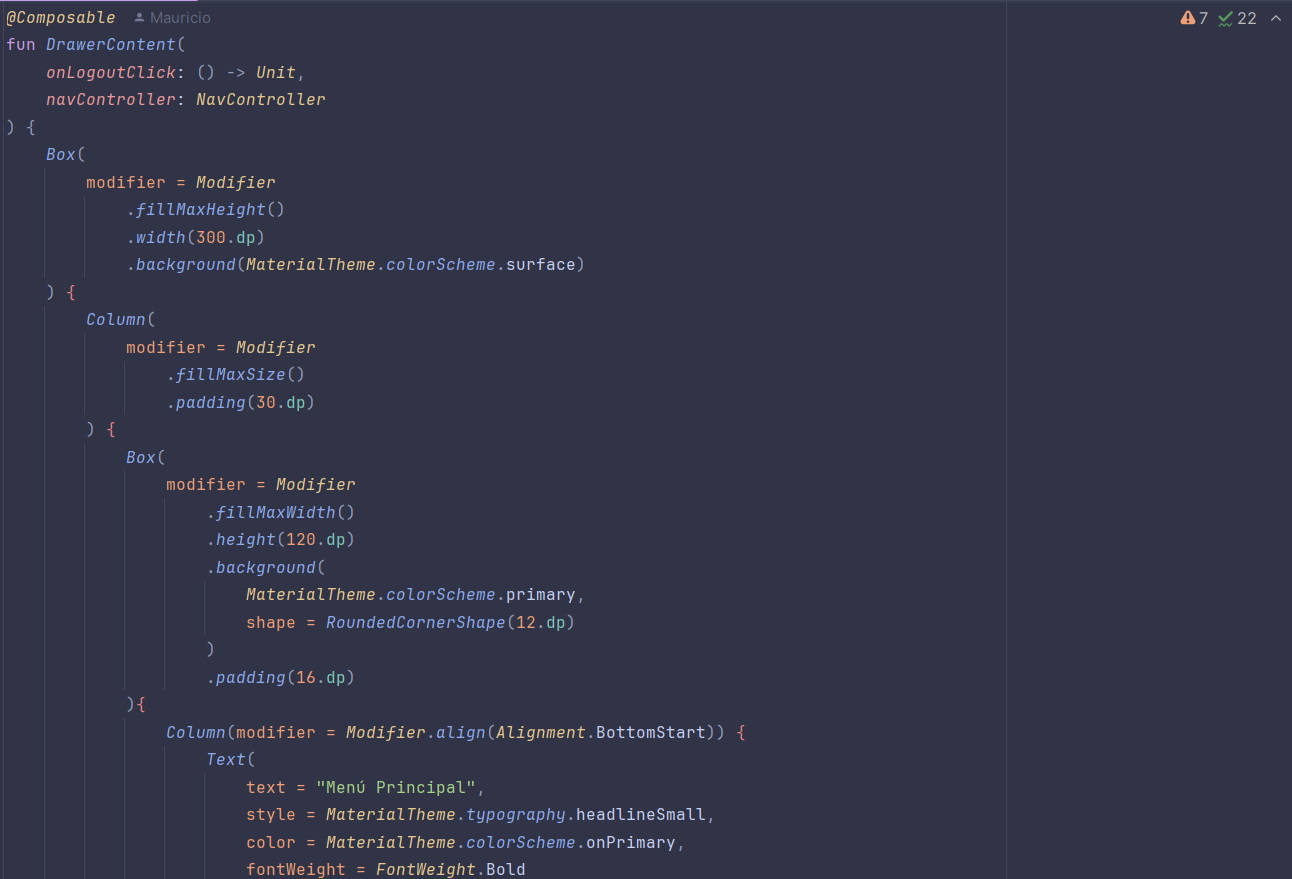
A blue background with red and white dots

Description automatically generated

**Image** representa la URL o la ruta de la imagen que se mostrará, es un parámetro obligatorio para que el componente funcione correctamente.

**DrawerContent**:

El componente DrawerContent se utiliza para construir un menú lateral interactivo en la aplicación. Este menú permite al usuario acceder a diferentes secciones de la aplicación y ofrece la opción de cerrar sesión

****

**A screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

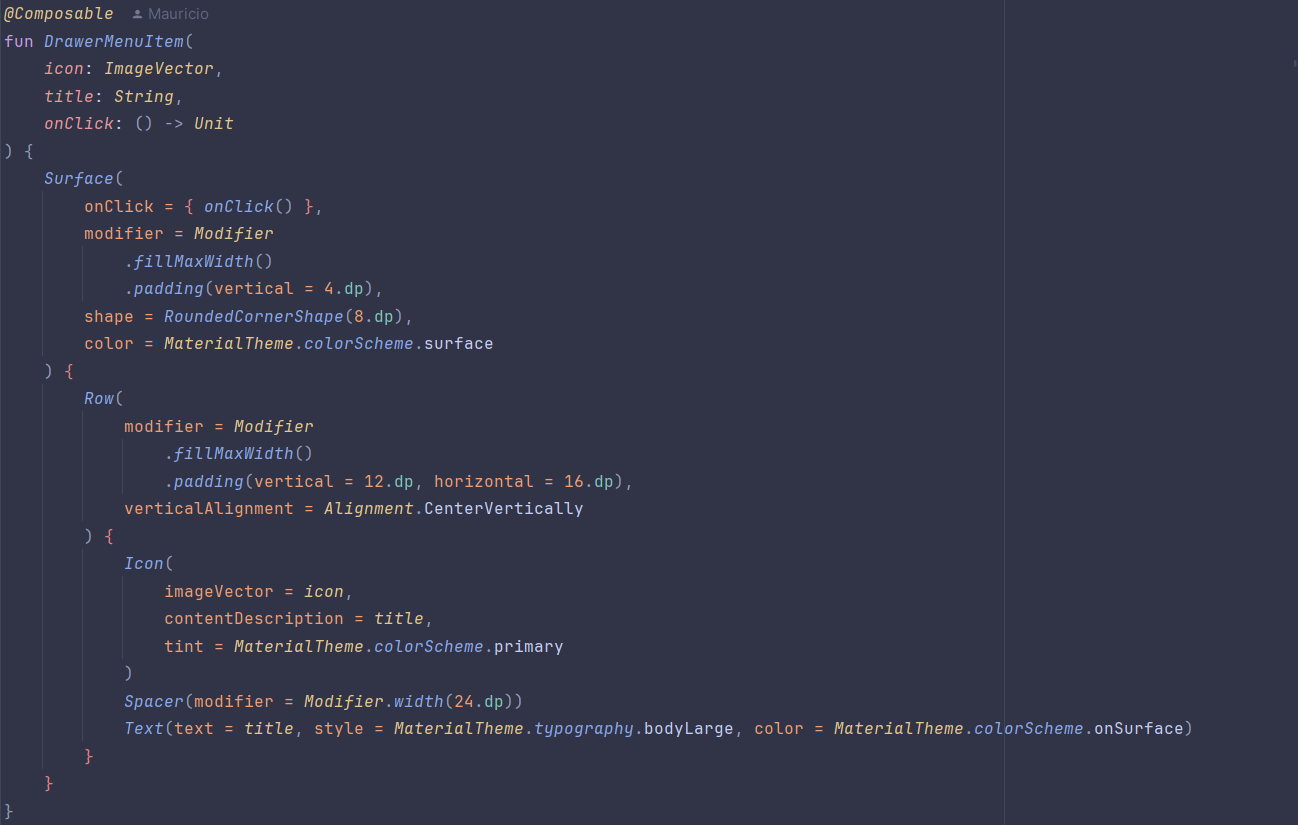
**onLogOutClick** Acción que se ejecuta al pulsar el botón Cerrar Sesión para redirigir al inicio de sesión.

**navController** Controlador de navegación utilizado para redirigir a las diferentes vistas de la aplicación según la opción seleccionada.

Las opciones del menú están representadas por componentes individuales (DrawerMenuItem) con iconos y textos que nos ayudan a navegar a rutas especificas usando navController.

**DrawerMenuItem:**

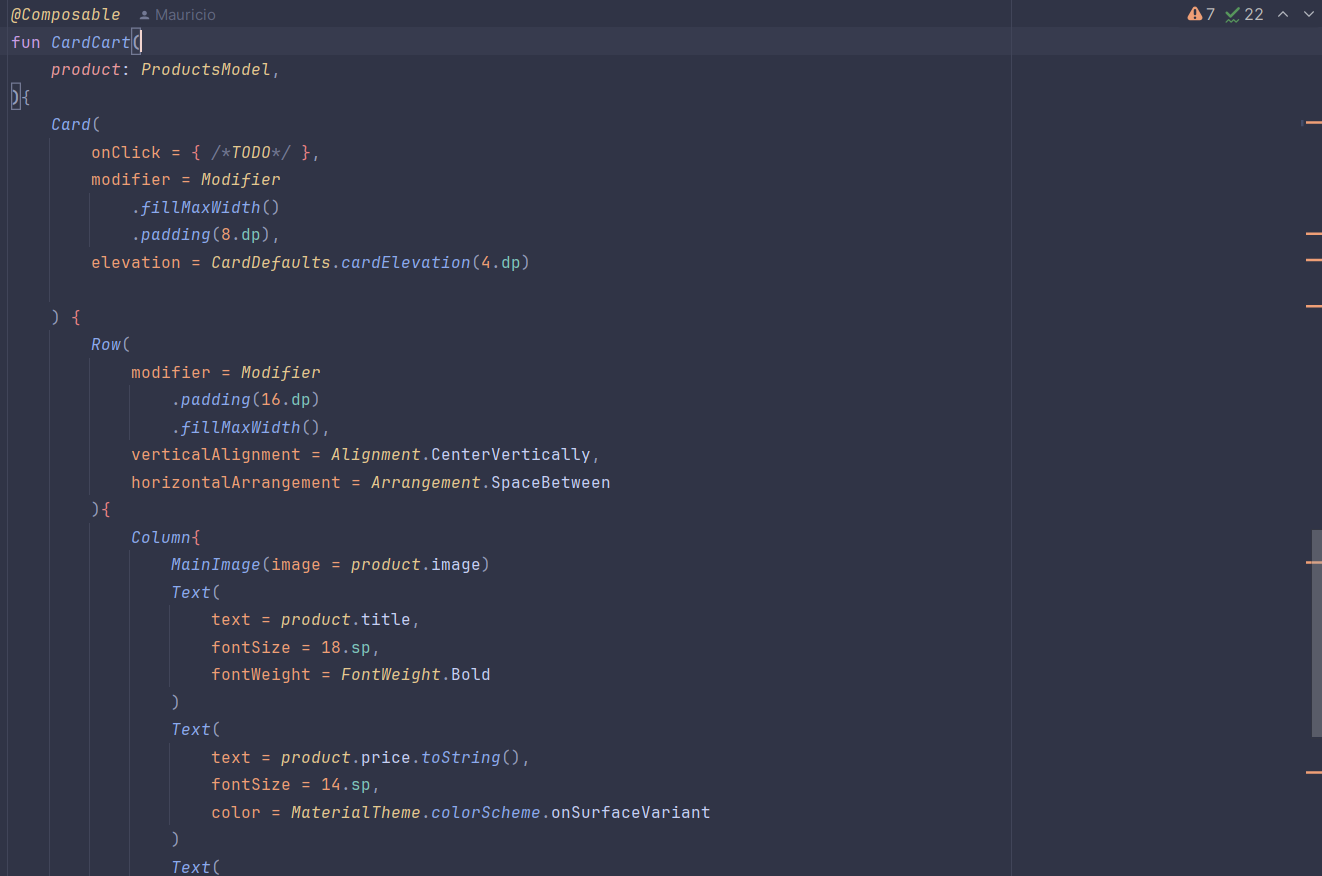
El componente DrawerMenuItem representa un elemento de menú que puede ser utilizado en un menú lateral.

****

Al hacer click en un item se ejecuta la acción proporcionada por el parámetro onClick. Nos ayuda A navegar por las diferentes vistas.

**CardCart:**

El componente CardCart representa una tarjeta diseñada para mostrar información de un producto en un carrito de compras.

 A computer screen shot of a computer code

Description automatically generated

**UserComponents:**

El componente UserInfo es una tarjeta que utilizamos para mostrar la informacion asociada al usuario. Este componente es flexible ya que nos permite agregar contenido adicional de manera dinamica a traves del parametro content.

****

Data:

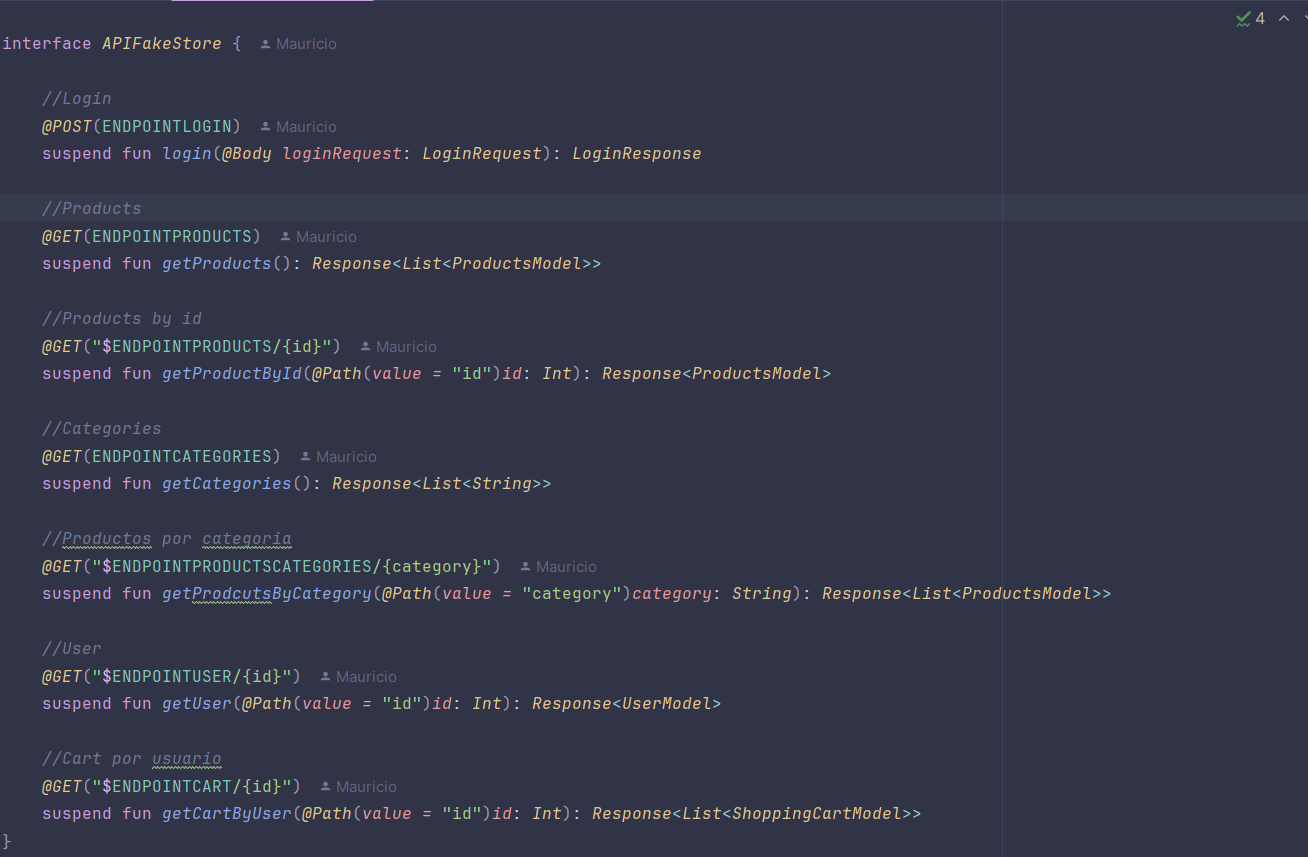
En esta carpeta tenemos el archivo que nos va a ayudar a gestionar todas las peticiones a la API FakeStore. Esta carpeta está diseñada para ser responsable de la obtención, manipulación y distribución de los datos de la aplicación.

A close up of a logo

Description automatically generated

**APIFakeStore:**

La interfaz APIFakeStore define las operaciones para interactuar con los diferentes endpoints de la API FakeStore. Utiliza Retrofit para realizar solicitudes HTTP y obtener daros relacionados con los productos, categorías, usuarios y más. Cada función de la interfaz está anotada con las solicitudes correspondientes y los parámetros necesarios para acceder a los recursos de la API.



A screen shot of a computer

Description automatically generated

A computer screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

**Di:**

La carpeta di contiene los archivos relacionados con la configuración y gestión de la inyección de dependencias en el proyecto. En esta carpeta definimos los módulos del Hilt que es la librería de inyección de dependencias utilizada en el proyecto.

A close up of a sign

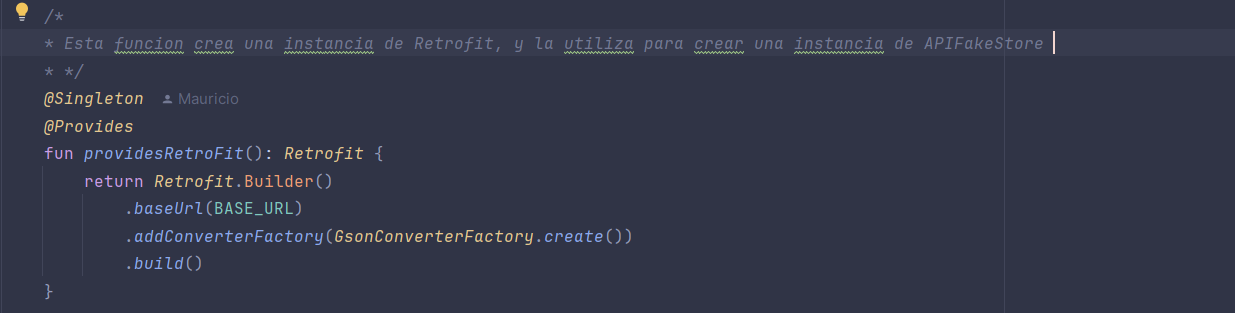
Description automatically generated

AppModule:

La clase de AppModule es un módulo de inyección de dependencias que utiliza Dagger Hilt para proporcionar instancias como RetroFit y APIFakeStore.

A screenshot of a computer program

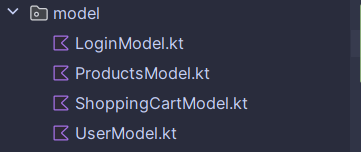
Description automatically generated





**Model:**

La carpeta contiene las clases que representan los datos o entidades de la aplicación.



**ProductsModel:**

Representa un producto dentro de la tienda. Contiene atributos como id, title, Price, description, category, image. Es utilizado para mostrar la información de los productos en la aplicación.

A blue square with white text

Description automatically generated with medium confidence

**LoginModel:**

LoginModel es utilizada para enviar los datos de inicio de sesión del usuario, como el email y la password. Esta clase es enviada como cuerpo en la solicitud de login a la API.

A blue square with white dots

Description automatically generated with medium confidence

**ShoppingCartModel:**

Representa los artículos agregados al carrito de compras del usuario. Contiene atributos como id, quantity, productId. Se utiliza para mostrar el contenido del carrito y gestionar las acciones relacionadas.

A blue square with white text

Description automatically generated with medium confidence

**UserModel:**

La clase UserModel es utilizada para representar la información detallada de un usuario dentro de la aplicación.

**A screenshot of a computer

Description automatically generated**

**Navigation:**

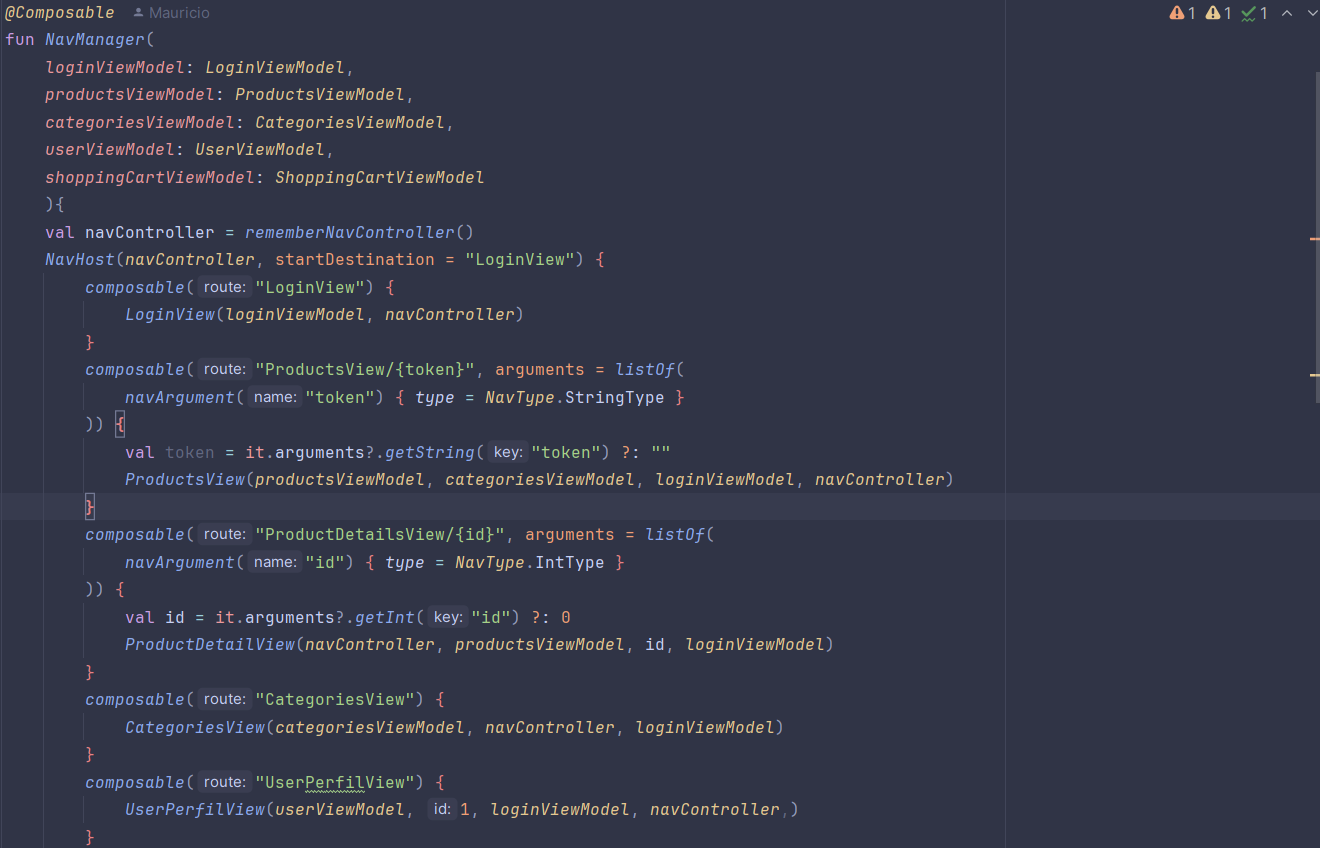
La carpeta navigation se encarga de gestionar la navegación entre las distintas pantallas o vistas de la aplicación utilizando Jetpack Navigation.

A close up of a computer screen

Description automatically generated

**NavManager:**

En este archivo definimos un gestor de navegación que utiliza el componente NavHost para manejar la navegación entre las diferentes pantallas o vistas de la aplicación. Este componente se encarga de dirigir al usuario a las pantallas correspondientes según las rutas definidas.



A screen shot of a computer program

Description automatically generated

**Repository:**

La carpeta repository tiene como función gestionar la comunicación con los datos desde APIFakeStore y proporcionar estos datos a los ViewModels.

A close up of a logo

Description automatically generated

FakeStoreRepository:



A computer screen shot of a program code

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Útil:**

El archivo Constants lo usamos para definir valores constantes que se utilizan en toda la aplicación para interactuar con la API FakeStore.

A close up of a text

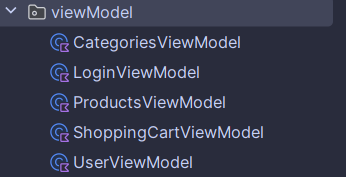
Description automatically generated

A screenshot of a computer program

Description automatically generated

**ViewModel:**

La carpeta ViewModel contiene las clases que actúan como intermediarias entre los repositorios de datos y las interfaces de usuario (vistas). Con los ViewModels podemos gestionar la lógica de negocio y el estado de las pantallas de la aplicación.



**CategoriesViewModel:**

Es una clase que gestiona los datos relacionados con las categorías y los productos por categoría en la aplicación, utilizando un repositorio (APIFakeStore) inyectado con Hilt.

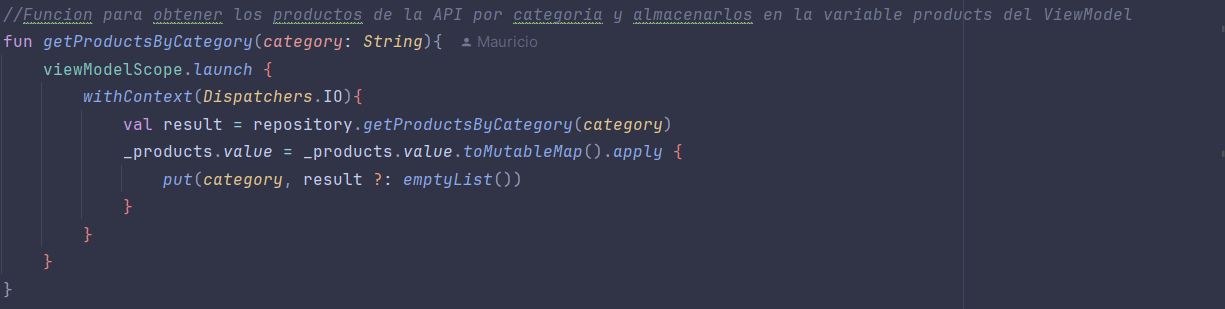
A screenshot of a computer program

Description automatically generated



A screenshot of a computer

Description automatically generated



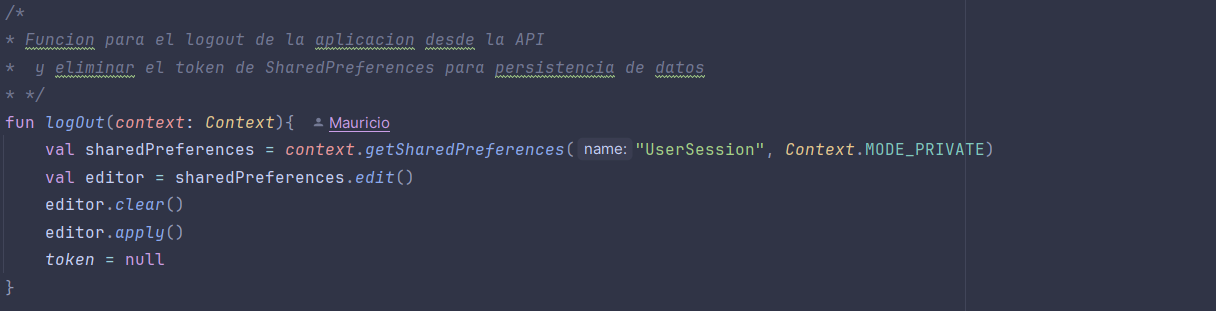
**LoginViewModel:**

El loginViewModel es una clase que gestiona la lógica de inicio de sesión en la aplicación y es responsable de interactuar con el repositorio para realizar el proceso de autenticación, también para la administración del estado de la sesión de usuario.

**A screenshot of a computer program

Description automatically generated**

****

****

**A computer screen shot of text

Description automatically generated**

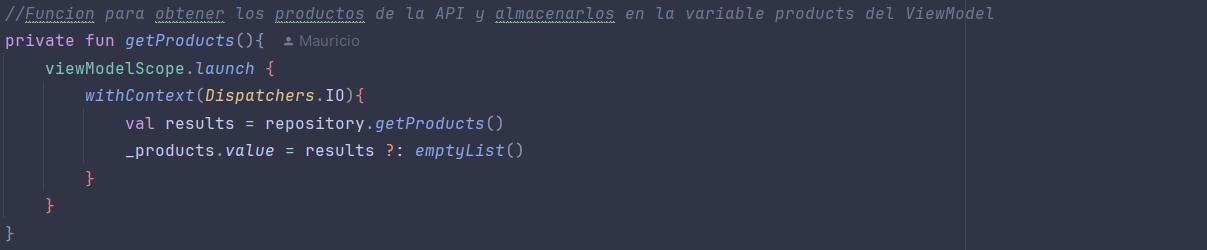
**A screenshot of a computer screen

Description automatically generated**

**ProductsViewModel:**

Se encarga de gestionar toda la lógica relacionada con los productos de la API. Y nos proporciona los datos y maneja el estado de las vistas de productos en la aplicación.

****

****

**A computer screen shot of a program

Description automatically generated**

**A computer screen shot

Description automatically generated**

**ShoppingCartViewModel:**

El shoppingCartViewModel gestiona la lógica relacionada con el carrito de compras de los usuarios.

****

**A computer screen shot

Description automatically generated**

**UserViewModel:**

Se encarga de gestionar la lógica relacionada con la obtención de información de un usuario especifico desde la API.

A screen shot of a computer

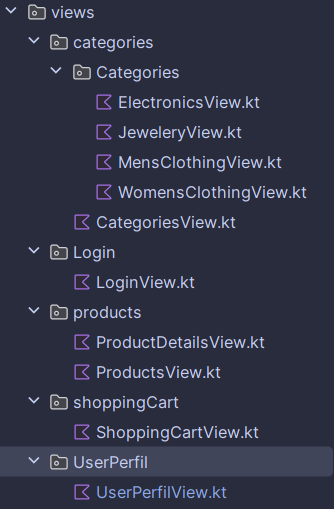
Description automatically generated

A screenshot of a computer

Description automatically generated

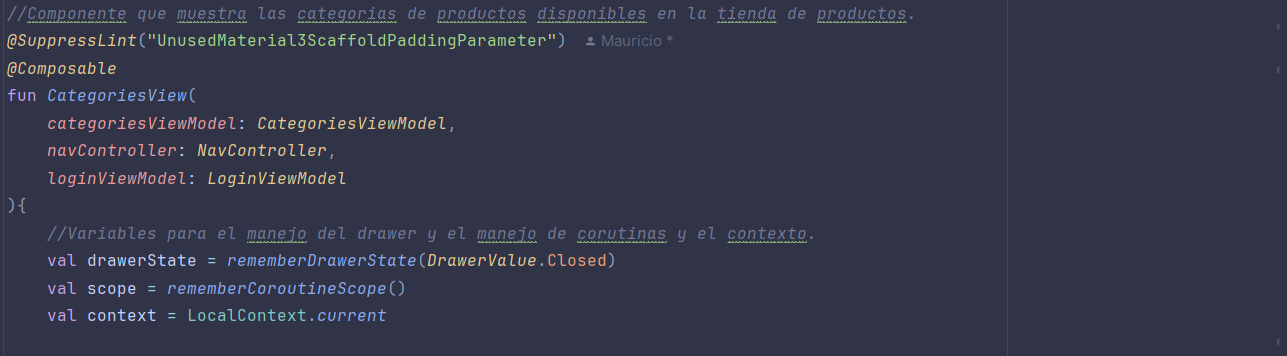
**Views:**

La carpeta views contiene los componentes y pantallas de la interfaz de usuario de la aplicación. Cada carpeta y archivo representa una vista especifica que corresponde a una funcionalidad clave de la aplicación.

****

**CategoriesView:**

El componente Categories View es la pantalla principal para mostrar las categorías disponibles obtenidas desde la API.

A screen shot of a computer screen

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

**Electronics:**

El componente Electronics está diseñado para mostrar una lista de productos de la categoría Electronics de la tienda.



**A screen shot of a computer screen

Description automatically generatedA screen shot of a computer code

Description automatically generated**

**Jewelry:**

El componente Jewelry está diseñado para mostrar una lista de productos de la categoría Jewelry de la tienda.

A screen shot of a computer

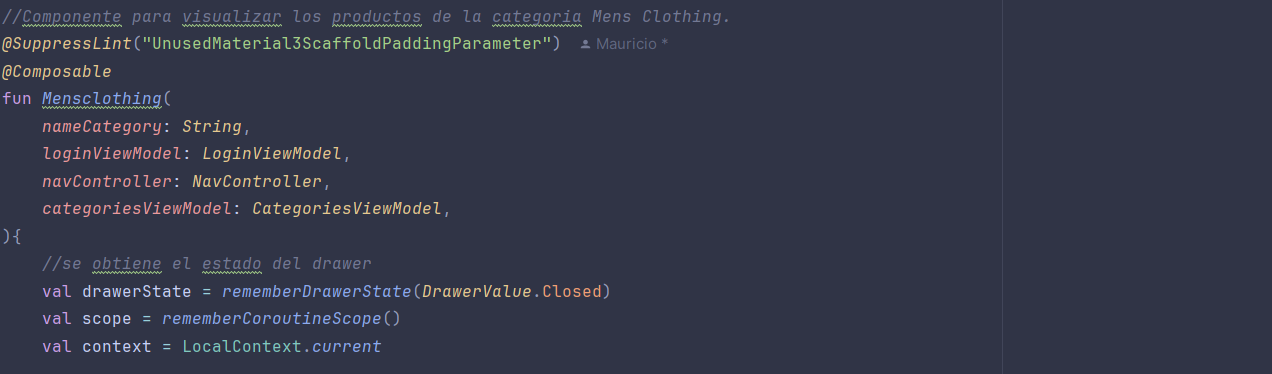
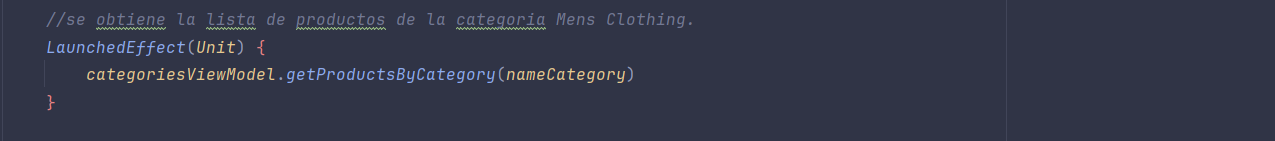
Description automatically generatedA computer screen shot of a program code

Description automatically generatedA screen shot of a computer code

Description automatically generated

**MensClothing:**

El componente **MensClothing** está diseñado para mostrar una lista de productos de la categoría **MensClothing** de la tienda.

  A blue screen with white text

Description automatically generated A screen shot of a computer program

Description automatically generated A screen shot of a computer code

Description automatically generated

**Guía de configuración y ejecución:**

Requisitos: Android Studio, Conexión a internet.

Pasos:

* Clonar el repositorio.

<https://github.com/Mauro1049/FakeStore.git>

**git clone <**[**https://github.com/Mauro1049/FakeStore.git**](https://github.com/Mauro1049/FakeStore.git)**>**

**cd FakeStore**

**./gradlew build**