



Proyecto de final de ciclo

Zoo Not Logic

Autor: Iván Tortosa Gimeno

Curso: 2023 - 2024

Ciclo: Grado superior de DAM

Centro: IES Dr. Lluís Simarro

Tutor: Rafael Vidal Semper

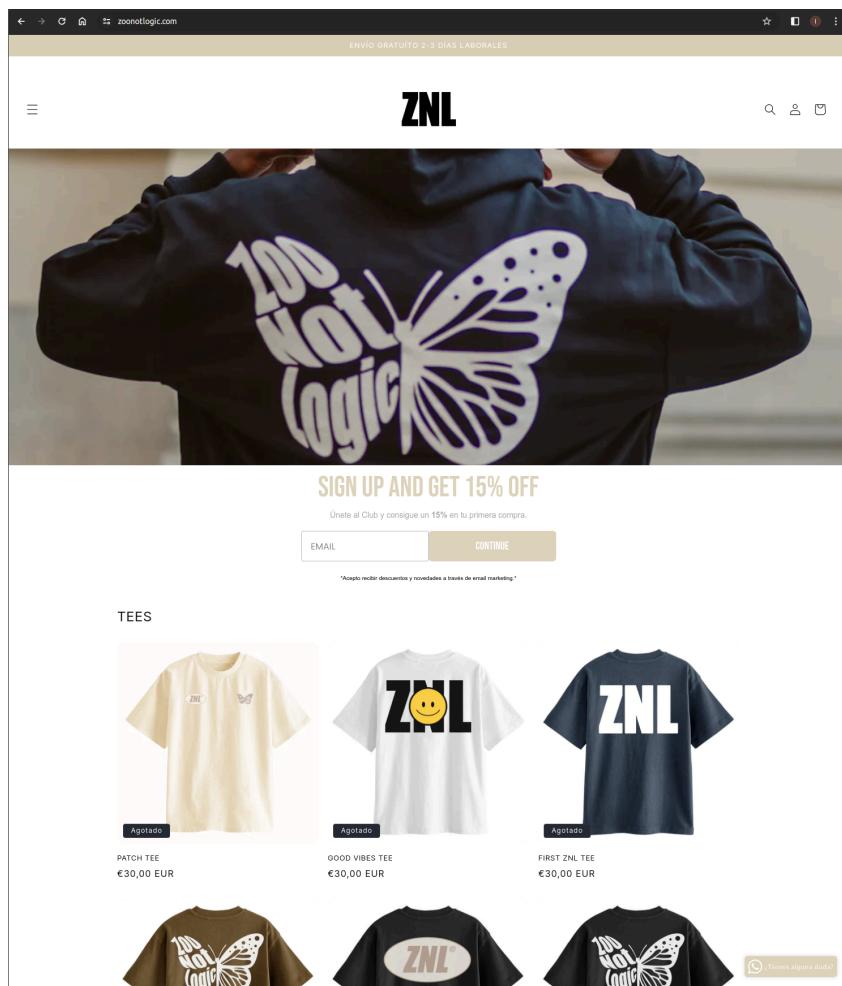
Índice

1.	Introducción.....	3
2.	Objetivos.....	4
3.	Desarrollo de la aplicación.....	5
3.1.	Análisis.....	5
3.2.	Diseño.....	6
	LoginScreen.....	6
	SignupScreen.....	7
	HomeScreen.....	8
	DetailScreen.....	9
	CartScreen.....	11
	ProfileScreen.....	13
	OrderDetailScreen.....	14
	AdminScreen.....	14
	ProductScreen.....	15
	OrdersScreen.....	16
3.3.	Implementación.....	16
	MVVM y Clean Architecture.....	16
	Inyección de dependencias con Dagger Hilt.....	18
4.	Conclusiones.....	21

1. Introducción

Este documento presenta un análisis del proceso de desarrollo de una aplicación Android creada a partir de una página web, utilizando Jetpack Compose, una arquitectura MVVM (Model-View-ViewModel) y siguiendo los principios de Clean Code. La aplicación está destinada a la compra de productos y gestión de una tienda de ropa, tiene como objetivo principal proporcionar una experiencia de compra fluida, y ofrecer herramientas para la gestión de productos a los administradores de la tienda.

Es importante destacar que la elección de Jetpack Compose como tecnología de interfaz de usuario proporciona ventajas en términos de rendimiento, flexibilidad y mantenibilidad. La arquitectura MVVM se ha adoptado para garantizar una separación de responsabilidades ya que facilita la escalabilidad de la aplicación. Además, seguir los principios de Clean Code garantiza que el código sea fácil de entender, mantener y ampliar en el futuro, lo que permite añadir nuevas funcionalidades de manera eficiente y sin comprometer la calidad del código existente.



2. Objetivos

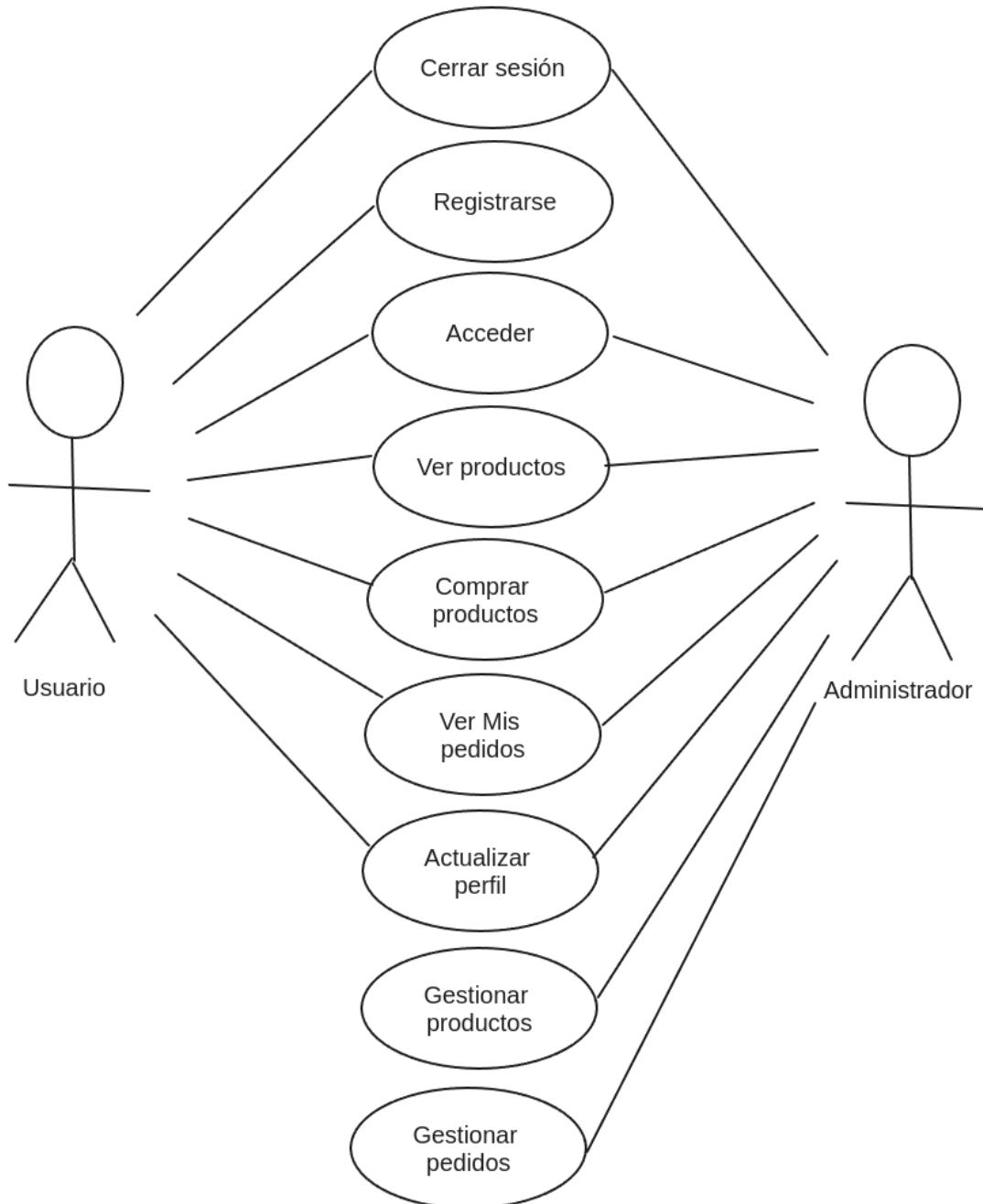
Los principales objetivos del desarrollo de esta aplicación son:

1. Interfaz de usuario intuitiva: Diseñar una interfaz de usuario intuitiva para facilitar su uso por parte de los usuarios.
2. Funcionalidad de inicio de sesión seguro: Se utilizara Firebase Auth para gestionar la autenticación de usuarios. Se implementará la autenticación mediante email y contraseña.
3. Compra de productos: Los usuarios podrán navegar por la pantalla de home la cuál contendrá una sección de nueva colección y otra con todo el catálogo de productos.
4. Gestión de perfil: Cada usuario tendrá su propio perfil donde podrán ver, editar su información y cerrar la sesión de la aplicación.
5. Gestión de productos: Si el usuario es administrador, se habilitará un botón para acceder a la gestión de productos. Donde podrá ver el estado del stock, crear productos nuevos y modificar los existentes.
6. Gestión de pedidos: Los usuarios pueden acceder a “Mis pedidos” en su perfil para ver una lista con todos sus pedidos, ver el estado y entrar la pantalla de detalle. Los administradores tienen acceso a todos los pedidos y pueden modificar el estado en el momento que sea enviado.
7. Integración de pagos: Se utilizará Stripe como plataforma para gestionar los pagos dentro de la aplicación.

3. Desarrollo del proyecto

3.1 Análisis

Diagrama de los principales casos de uso



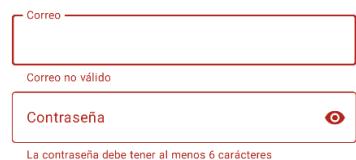
3.2 Diseño

LoginScreen

La pantalla de login (LoginScreen) es la primera que se muestra al iniciar la aplicación. Permite autenticarse en Firebase Authentication mediante el correo y la contraseña para acceder a la pantalla principal (HomeScreen). También proporciona un enlace para acceder a la pantalla de registro (SignupScreen) en el caso de que no tengas usuario. Además, verifica si el usuario ya ha iniciado sesión anteriormente y, de ser así, redirige directamente a la pantalla principal (HomeScreen).



Incluye validaciones para asegurar que el correo esté correctamente formateado y la contraseña tenga al menos 6 caracteres.



Si las credenciales no son válidas, se muestra un Snackbar con el error correspondiente.



SignupScreen

La pantalla (SingupScreen) es para el registro de usuarios, en el caso de completar el registro se accede directamente a la pantalla principal (HomeScreen), en caso de cancelar se vuelve a la pantalla de login (LoginScreen). Se solicitan tres campos obligatorios: email, contraseña y nombre.



Cada uno tiene su validación, el email que esté correctamente formateado, la contraseña debe tener al menos 6 caracteres, y el nombre debe tener al menos 4 caracteres.

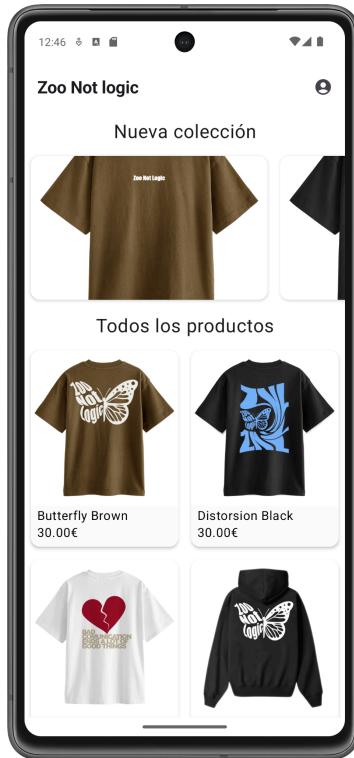


Los apellidos y el teléfono son opcionales ya que en la pantalla de perfil se puede modificar. Cuando se pulsa el botón de registrar si el email ya está en uso por otro usuario, se muestra un Snackbar con el error correspondiente.



HomeScreen

La pantalla principal (HomeScreen) es la primera que se muestra después de iniciar la sesión. En esta pantalla, se presenta una sección de “Nueva colección” y otra con todos los productos disponibles. Al pulsar en cada producto, se accede a la pantalla de detalle (DetailScreen), donde se muestra información detallada del producto seleccionado. Además, en la barra superior se encuentra un ícono que permite acceder a la pantalla de perfil (ProfileScreen).



Cuando se añaden productos al carrito aparece una Card donde nos muestra información de la cantidad de productos añadidos y el precio total. Si pulsamos navegamos a la pantalla del carrito (CartScreen).



En el caso de iniciar sesión como administrador se añaden 2 íconos en la barra superior, uno para acceder a la gestión de los productos y otro para acceder a la pantalla de gestión de pedidos



DetailScreen

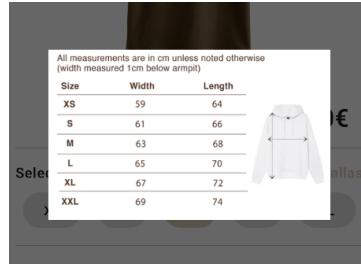
La pantalla de detalles del producto (DetailScreen) donde se muestra una vista detallada del producto. En esta pantalla tenemos un slider para mostrar las imágenes del producto, el precio del producto, un selector de tallas y una descripción del producto. Hasta que no se selecciona una talla no se habilita el botón de “añadir al carrito”.



Hasta que no se selecciona una talla no se habilita el botón de “añadir al carrito”.



Si pulsamos en el botón de “Guía de tallas” nos muestra una imagen con las medidas de cada talla.

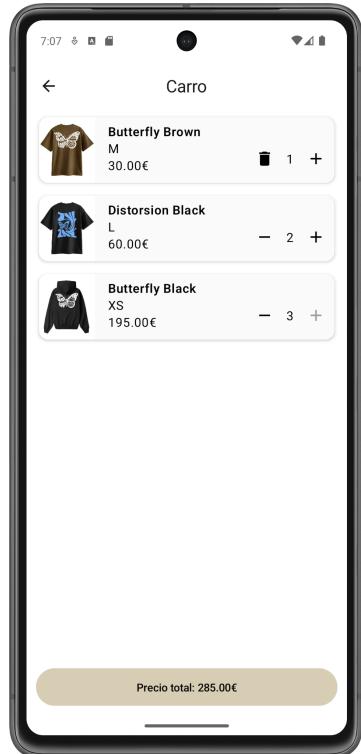


También se nos muestra un Snackbar para informar que hemos añadido el producto al carrito y otro cuando llegamos al límite de productos que se pueden comprar de la misma talla.



CartScreen

La pantalla de carrito (CartScreen) permite a los usuarios modificar la cantidad de productos que desean comprar. Si la cantidad de un producto es uno, se muestra un icono de papelera para eliminarlo o un icono de más para añadir más cantidad de ese producto. Cuando la cantidad es dos, el icono de papelera se convierte en un icono de menos para disminuir la cantidad. Si se alcanza el límite máximo de tres unidades de un mismo producto, el botón para añadir más se deshabilita.



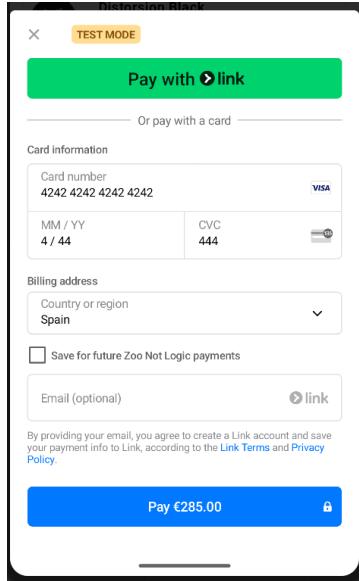
En la parte inferior de la pantalla, se muestra un botón que indica el precio total a pagar. Si se eliminan todos los productos se muestra un mensaje “El carro está vacío” y se oculta el botón de pagar. Al pulsarlo, aparece un BottomSheet para que insertemos los datos para el envío. Si esos datos han sido previamente llenados en la pantalla de perfil, se completarán automáticamente.



Si algún campo de los datos de envío está vacío, el botón de “Pagar ahora” se deshabilita.



Cuando se pulsa el botón “Pagar ahora”, se activa la pasarela de pago. En este caso, se ha utilizado la librería de Stripe, la cual proporciona un PaymentSheet para introducir los datos de la tarjeta de crédito. Stripe se encarga de realizar las comprobaciones necesarias y devuelve el estado del pago, indicando si ha sido completado, cancelado o rechazado. Si el pago es completado, se guarda el pedido en la base de datos, se eliminan los productos del carro y se envía al usuario a la pantalla principal (HomeScreen).



ProfileScreen

La pantalla de perfil (ProfileScreen) permite a los usuarios modificar sus datos personales y dirección de entrega. En la parte superior, hay un ícono de guardado que permite guardar los datos si se realizan modificaciones. Además, en esta pantalla se pueden ver los pedidos realizados. su estado y acceder a la pantalla de detalle de pedido.

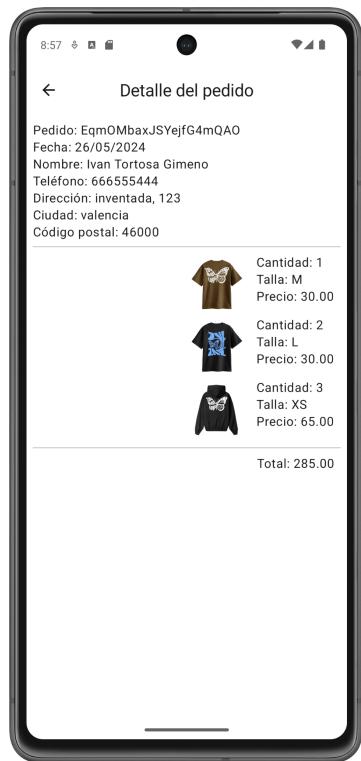


En la parte inferior del contenido de la pantalla, se encuentra el botón para cerrar la sesión. Este botón hace el logout en Firebase Authentication y redirige a la pantalla de inicio de sesión (LoginScreen).



OrderDetailScreen

La pantalla de detalle del pedido (OrderDetailScreen) proporciona una vista detallada de cada pedido realizado. En esta pantalla se muestra información relevante, como el número de pedido, la fecha de compra, el nombre del cliente, el teléfono, la dirección de envío y los productos comprados. Para cada producto, se muestra su foto, cantidad, talla, precio unitario. En la parte inferior se muestra el precio total del pedido.

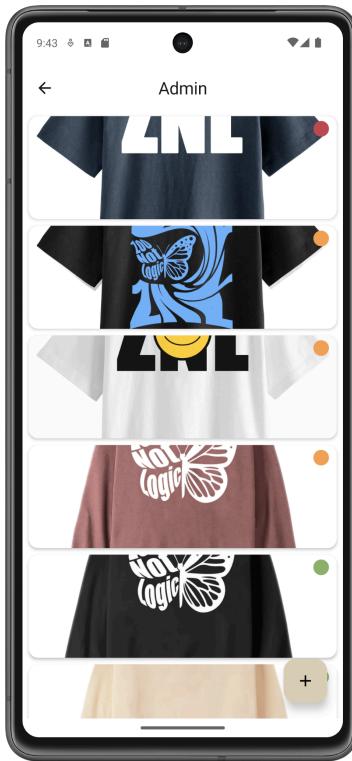


AdminScreen

La pantalla de administración de productos (AdminScreen) es accesible al pulsar la llave inglesa en la pantalla de inicio (HomeScreen) cuando el usuario es administrador. En esta pantalla se muestran todos los productos disponibles. Cada producto tiene un indicador en la

parte superior derecha que muestra el estado de su stock. Un punto rojo indica que alguna de las tallas tiene una cantidad de stock inferior a cinco, un punto naranja indica que alguna de las tallas tiene una cantidad de stock inferior a diez, y un punto verde indica que el stock de todas las tallas es superior a diez. La lista de productos está ordenada en función del stock, primero los que tienen el punto rojo, luego los que tienen el punto naranja y por último los que tienen el punto verde.

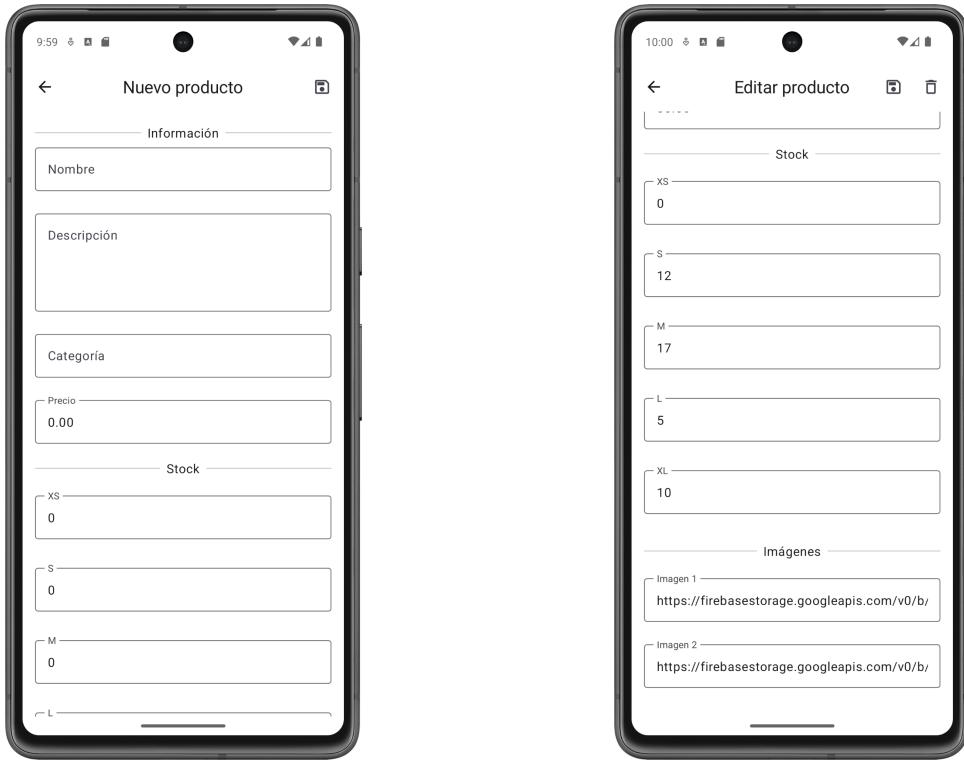
En la parte inferior de la pantalla se encuentra un Floating Action Button que permite crear nuevos productos. Al pulsar en la imagen de un producto, se puede editar el producto seleccionado. Tanto el crear un nuevo producto como el editar un producto existente, te envía a la pantalla de crear producto (ProductScreen). En el primer caso, el formulario está vacío y se crea un nuevo producto, mientras que en el segundo caso, el formulario se rellena con los datos del producto existente.



ProductScreen

La pantalla de creación y edición de productos (ProductScreen) es accesible solo para usuarios administradores. Cuando creas un nuevo producto, en la TopBar solo aparece el ícono de guardar, mientras que al editar un producto existente, aparecen los iconos de guardar y eliminar.

En esta pantalla, se puede editar la información del producto, incluyendo su nombre, descripción, categoría, precio, stock e imágenes.



Cuando se pulsa el botón de eliminar, se muestra un diálogo de confirmación que solicita al administrador que confirme la eliminación o la cancele.



OrdersScreen

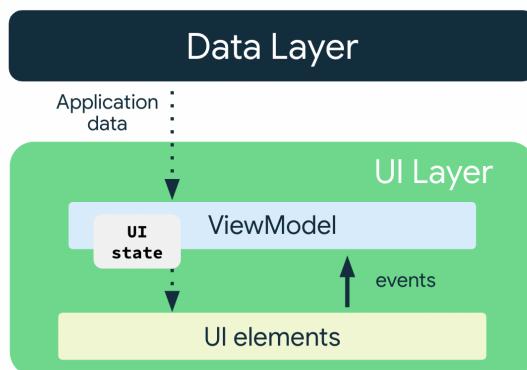
La pantalla de gestión de pedidos realizados por los usuarios (OrdersScreen), se accede a través de la pantalla de inicio (HomeScreen) al pulsar en el ícono de la bolsa de la compra cuando eres administrador. En esta pantalla se muestra para cada pedido el número de pedido, la fecha, el estado y el precio total. Si se pulsa en un pedido, se redirige a la pantalla de detalle de pedido (OrderDetailScreen). Además, aparece un botón que permite cambiar el estado del pedido a "Enviado" una vez que ha sido enviado. Esto permite a los usuarios comprobar el estado de sus pedidos.



3.3 Implementación

MVVM y Clean Architecture

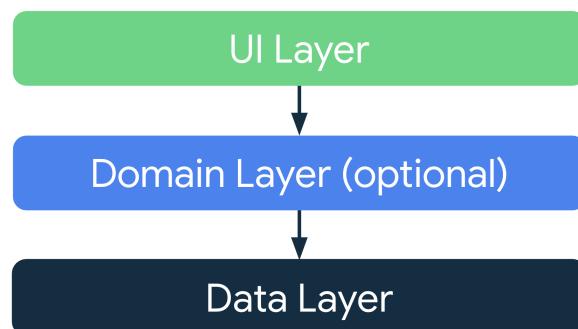
MVVM es un patrón de diseño utilizado en el desarrollo de aplicaciones que separa los componentes de la interfaz de usuario de la lógica de negocio. Se compone de tres partes principales:



- **Model (Modelo)**: Representa los datos y la lógica de negocio de la aplicación. El modelo contiene clases que encapsulan los datos y las operaciones relacionadas con estos datos.

- View (Vista): Es la capa de presentación de la aplicación que se encarga de mostrar la interfaz al usuario final. La vista no debe contener lógica de negocio.
- ViewModel: Actúa como un intermediario entre la vista y el modelo. Se encarga de manejar la lógica de presentación y de exponer los datos y estados necesarios para la vista. El ViewModel también se encarga de manejar las interacciones del usuario y de comunicarse con el modelo para obtener o modificar los datos según sea necesario.

Clean Architecture es un estilo de diseño de software que promueve la creación de sistemas bien estructurados, modularidad y altamente mantenibles. Se basa en los principios de separación de preocupaciones, y consta de varias capas con responsabilidades bien definidas:



- Capa de Presentación (UI Layer): Contiene la lógica relacionada con la interfaz de usuario y la interacción con el usuario. Esta capa se comunica con las capas inferiores a través de abstracciones.
- Capa de Dominio (Domain Layer): Contiene la lógica de negocio central de la aplicación. Esta capa es independiente de cualquier framework o biblioteca y encapsula las reglas de negocio y la lógica de procesamiento de datos.
- Capa de Datos: Gestiona las fuentes de datos y su acceso. Esta capa se encarga de interactuar con bases de datos, servicios web, APIs y otros sistemas externos.

La estructura del proyecto es la siguiente:

Data Layer (Capa de datos)

- di (dependency Injection): Configuración de las dependencias con Dagger Hilt.
- mapper: Transforma los DTO's en modelos de dominio y viceversa.

- remote: Contiene las implementaciones de los repositorios y los DTO's
 - DTO's (Data Transfer Objects): Representaciones de los datos que se intercambian con las fuentes remotas.
 - Repositories: Implementaciones de los repositorios encargados de obtener y almacenar los datos.

Domain Layer (Capa de dominio)

- model: Clases que representan los datos y son utilizadas en toda la aplicación.

Navigation (Navegación)

- NavigationGraph: Define el grafo de navegación de la aplicación, que incluye las rutas y cómo se relacionan entre sí.
- Routes: Enumera las rutas disponibles en la aplicación.

Presentation Layer (Capa de presentación)

- composables: Composables que representan componentes de la pantalla.
- screen: Es el Composable que representa una pantalla completa de la aplicación.
- state: Representa el estado de la UI para cada pantalla.
- ViewModel: Manejan la lógica de presentación y exponen los estados de la UI.

Inyección de dependencias con Dagger Hilt

Hilt es una biblioteca de inyección de dependencias estable para Android que te permite reducir el código estándar utilizado en la inyección de dependencias manual en tu proyecto. Hilt ofrece una manera estándar de inyectar dependencias en tu aplicación proporcionando contenedores a cada componente de Android de tu proyecto y administrando el ciclo de vida de los contenedores automáticamente. Para ello, se aprovecha de la popular biblioteca de inyección de dependencias Dagger.

- Agregar el plugin de Hilt en el archivo “build.gradle” del proyecto.

```
// Dagger hilt
alias(libs.plugins.daggerHiltAndroid) apply false
```

- Agregar dependencias al archivo “build.gradle” del módulo app

```

plugins {
    alias(libs.plugins.daggerHiltAndroid)
}

dependencies {
    // Dagger Hilt
    implementation(libs.dagger.hilt.android)
    kapt(libs.dagger.hilt.android.compiler)
    implementation(libs.hilt.navigation.compose)
}

```

- Agregar librerías al archivo libs.versions.toml

```

[versions]
daggerHilt = "2.46"
hilt = "1.0.0"

[libraries]
#Dagger hilt
dagger-hilt-android = { group = "com.google.dagger", name =
"hilt-android", version.ref = "daggerHilt" }
dagger-hilt-android-compiler = { group = "com.google.dagger",
name = "hilt-android-compiler", version.ref = "daggerHilt" }
hilt-navigation-compose = { group = "androidx.hilt", name =
"hilt-navigation-compose", version.ref = "hilt" }

[plugins]
#Dagger hilt
daggerHiltAndroid = { id = "com.google.dagger.hilt.android",
version.ref = "daggerHilt" }

```

- Para agregar un contenedor que esté vinculado con el ciclo de vida de la app, es necesario anotar la clase Application con `@HiltAndroidApp`.

```

@HiltAndroidApp
class ZooNotLogicApp : Application()

```

- Inyectar dependencias en una Activity.

```

@AndroidEntryPoint
class MainActivity : ComponentActivity() {

```

- Crear un módulo de Hilt para proporcionar dependencias.

```

@Module

```

```

@InstallIn(SingletonComponent::class)
object FirebaseModule {

    @Singleton
    @Provides
    fun provideFirebaseAuth(): FirebaseAuth =
        FirebaseAuth.getInstance()

    @Singleton
    @Provides
    fun provideAuthRepository(firebaseAuth: FirebaseAuth): AuthRepository =
        AuthRepository(firebaseAuth)

    @Singleton
    @Provides
    fun provideFirestore(): FirebaseFirestore =
        Firebase.firestore

    @Singleton
    @Provides
    fun provideFirestoreRepository(firestore: FirebaseFirestore): FirestoreRepository =
        FirestoreRepository(firestore)
}

```

- Inyectar dependencias en el ViewModel

```

@HiltViewModel
class HomeViewModel @Inject constructor(
    savedStateHandle: SavedStateHandle,
    private val firestoreRepository: FirestoreRepository
) : ViewModel() {

```

- Inyectar el ViewModel en un Composable

```

@Composable
fun HomeScreen(
    viewmodel: HomeViewModel = hiltViewModel()
)

```

4. Conclusiones

Las conclusiones de este análisis del desarrollo de la aplicación Android para la gestión y compra de productos destacan las áreas de mejora que se han identificado. Utilizando Jetpack Compose, arquitectura MVVM y principios de Clean Code, la aplicación ha logrado cumplir sus objetivos iniciales, pero hay varias áreas donde se pueden implementar mejoras significativas.

- Personalización y Notificaciones: Implementar recomendaciones personalizadas y notificaciones push para mantener a los usuarios informados sobre nuevas colecciones, ofertas y actualizaciones de pedidos.
- Búsqueda: Implementar funcionalidad de búsqueda con filtros.
- Autenticación Multifactor: Implementar autenticación multifactor para una mayor seguridad en el inicio de sesión.
- Automatización de Inventario: Implementar alertas automáticas para reponer stock y mejorar la gestión de inventarios.
- Integraciones con Redes Sociales: Permitir a los usuarios registrarse e iniciar sesión a través de sus cuentas de redes sociales (Facebook, Google, etc).
- Automatización de Pruebas: Ampliar la cobertura de pruebas automatizadas para incluir pruebas de UI, pruebas de rendimiento y pruebas de seguridad.
- Kotlin Multiplatform: Desarrollar la aplicación para que sea compatible con iOS usando Kotlin Multiplatform, permitiendo que la aplicación esté tanto en Android como en iOS

Implementar estos puntos de mejora contribuirá significativamente a una mejor experiencia de usuario, seguridad, y escalabilidad, además de ampliar el alcance de la aplicación a usuarios iOS.