



UNIVERSIDAD CENTRAL DEL ECUADOR
FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

CARRERA:

Computación

NOMBRES:

Diego Casagallo
Kevin Celi
Dylan Lema
Jhonny Ninabanda
Luis Perenguez

ASIGNATURA:

DISPOSITIVOS MÓVILES

TAREA:

TARE07-U2-G5 APP PRODUCTOS

DOCENTE:

MILTON MONCAYO

QUITO-ECUADOR

Modelo de Navegación de la App BookStore UCE

1. Introducción

El modelo de navegación de la aplicación BookStore UCE define cómo el usuario se desplaza entre las distintas pantallas de la app, garantizando una experiencia clara, fluida y coherente. Este modelo está implementado utilizando **Jetpack Compose** y se basa en un enfoque de **navegación declarativa**, donde el estado de la aplicación determina qué pantalla se muestra.

2. Tipo de Navegación

Navegación Declarativa

La app utiliza **navegación declarativa**, característica principal de Jetpack Compose. En este enfoque:

- La UI se actualiza automáticamente cuando cambia el estado.
- No se manipulan vistas manualmente.
- La navegación depende de eventos (login, logout, agregar, editar, guardar).

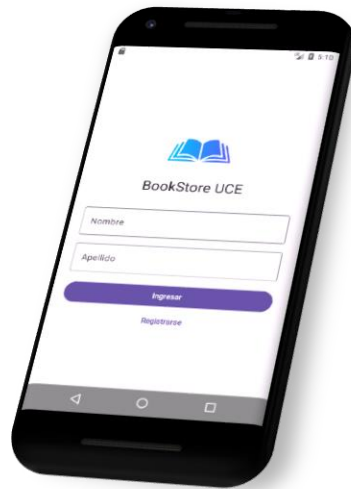
Navegación por Pantallas (Screen-based Navigation)

La app está compuesta por **pantallas independientes**, cada una representando una funcionalidad específica:

- Login
- Registro
- Home (Listado de libros)
- Formulario de libro (Agregar / Editar)

3. Pantallas Principales

3.1 LoginScreen



Función: Permitir el acceso del usuario a la aplicación.

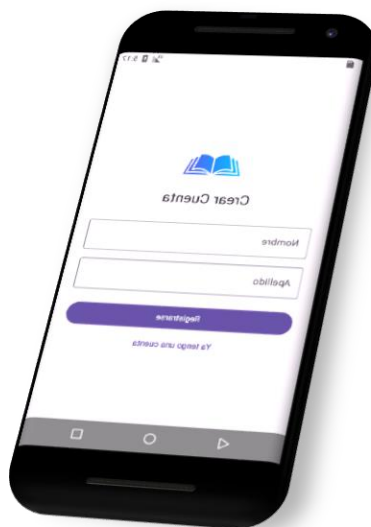
Acciones disponibles:

- Iniciar sesión → Navega a HomeScreen
- Ir a registro → Navega a RegisterScreen

Lógica:

- Si el login es correcto, se cambia el estado global y se muestra el Home.

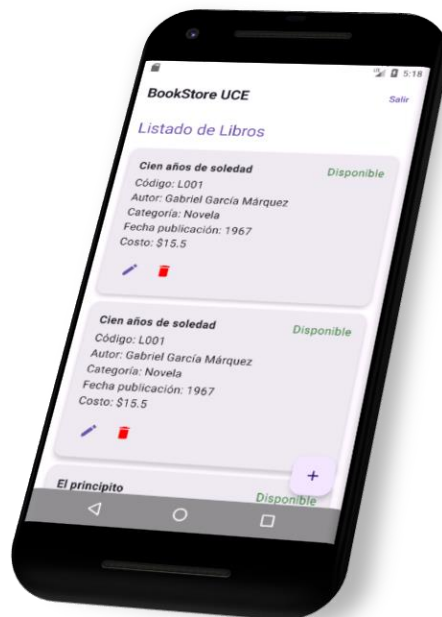
3.2 RegisterScreen



Función: Registrar un nuevo usuario.

Acciones disponibles:

- Guardar usuario → Regresa a LoginScreen
- Cancelar → Regresa a LoginScreen

3.3 HomeScreen

Función: Pantalla principal de la aplicación.

Elementos:

- Lista de libros (LazyColumn)
- Botón flotante para agregar libro
- Opciones de editar y eliminar libro
- Opción de cerrar sesión

Acciones y navegación:

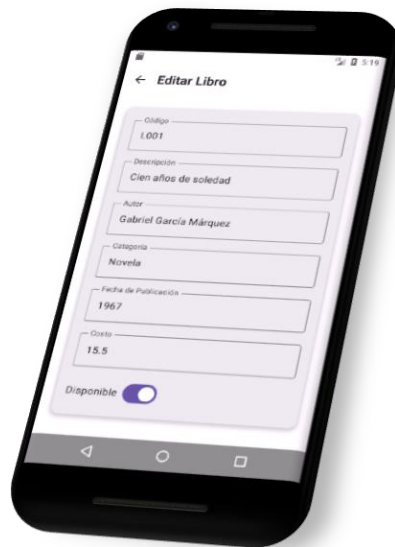
- Agregar libro → ProductFormScreen
- Editar libro → ProductFormScreen (modo edición)
- Eliminar libro → Se mantiene en Home y se actualiza la lista
- Cerrar sesión → LoginScreen

Lógica:

- La lista se actualiza automáticamente al agregar, editar o eliminar libros.

- Los datos se leen desde almacenamiento local (JSON).

3.4 ProductFormScreen



Función: Agregar o editar un libro.

Modo de uso:

- **Agregar:** cuando productToEdit == null
- **Editar:** cuando se recibe un producto y su índice

Campos:

- Código
- Descripción
- Autor
- Categoría
- Fecha de publicación
- Costo
- Disponibilidad

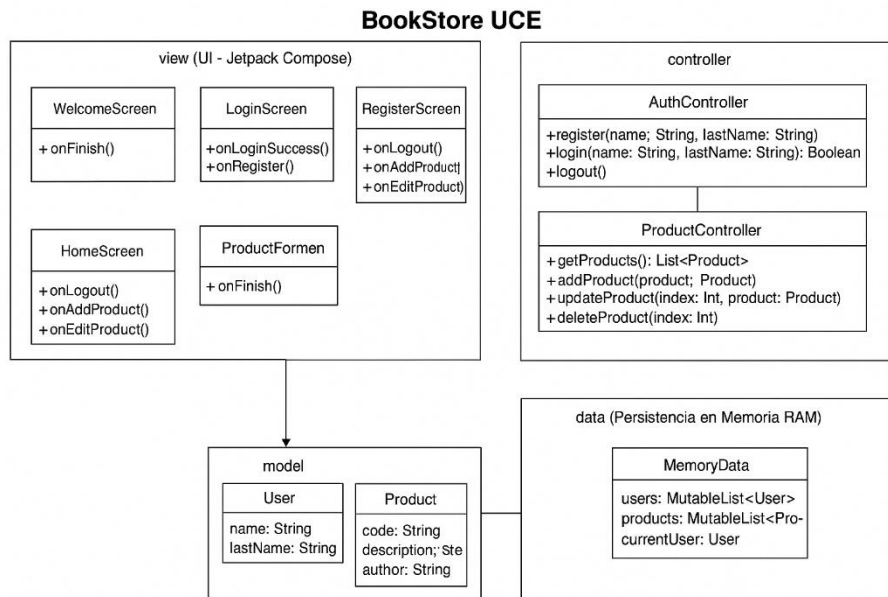
Acciones:

- Guardar cambios → Regresa a HomeScreen
- Volver sin cambios → Regresa a HomeScreen

Lógica especial:

- Se detectan cambios en los campos.
- Si no hay cambios, el botón actúa como "Volver".
- Si hay cambios, se guarda y luego se navega al Home.

4. Lógica de Navegación



Flujo General

LoginScreen

↓

HomeScreen

↓ ↘

ProductForm Logout

↓

HomeScreen

Desde **HomeScreen**, el usuario puede realizar distintas acciones principales.

Si decide **agregar un nuevo libro** o **editar uno existente**, la aplicación navega a la pantalla **ProductFormScreen**, donde se presenta un formulario para el ingreso o modificación de los datos del libro.

Una vez que el usuario guarda los cambios o decide volver sin modificar información, la navegación regresa a **HomeScreen**, donde la lista de productos se actualiza de forma automática.

Adicionalmente, desde **HomeScreen** el usuario tiene la opción de **cerrar sesión (Logout)**. Al ejecutar esta acción, el estado global de la aplicación se reinicia y el usuario es redirigido nuevamente a **LoginScreen**, cerrando el ciclo de navegación.

Este flujo garantiza una navegación sencilla, controlada y coherente, permitiendo al usuario desplazarse de manera intuitiva entre las principales funcionalidades de la aplicación.

5. Manejo del Estado

- Se utiliza `remember` y `mutableStateOf` para manejar el estado local.
- La navegación se controla mediante **lambdas** (`onFinish`, `onAddProduct`, `onEditProduct`, `onLogout`).
- No se utiliza `Navigation Component` por simplicidad académica.
- La aplicación sigue el patrón de diseño MVC (Modelo–Vista–Controlador), donde el Modelo está representado por los datos almacenados en memoria (`MemoryData`), la Vista corresponde a las pantallas desarrolladas con `Jetpack Compose` y el Controlador se implementa mediante eventos y lambdas que gestionan la lógica y navegación.

6. Persistencia de Datos en Memoria

Los datos de la aplicación se gestionan en memoria durante la ejecución mediante estructuras como `mutableListOf`, centralizadas en el objeto `MemoryData`.

Esta implementación permite que los usuarios y productos permanezcan disponibles mientras la aplicación está activa, cumpliendo con la consigna académica de no utilizar archivos ni bases de datos.

7. Ventajas del Modelo Implementado

Simple y fácil de entender

Ideal para proyectos académicos

Navegación clara y controlada

Compatible con `Jetpack Compose`

Fácil de escalar a `Navigation Component`

8. Conclusión

El modelo de navegación de `BookStore UCE` está diseñado para ser intuitivo, modular y coherente con los principios de `Jetpack Compose`.

Permite una experiencia fluida al usuario y facilita el mantenimiento y crecimiento de la aplicación.

9. Bibliografía:

[1] H. F. Siregar y A. Anggara, "Implementasi Android Jetpack Compose pada Aplikasi Machine Learning," Jurnal Informatika Teknologi dan Sains (JINTEKS). Disponible en: <https://doi.org/10.51401/jinteks.v7i2.5853>

Accedido: 14-dic-2025.

[2] J. Szczukin, "Performance analysis of user interface implementation methods in mobile applications," Journal of Computer Sciences Institute. Disponible en: <https://doi.org/10.35784/jcsi.3070>

Accedido: 14-dic-2025.

[3] I. Fjodorovs y S. Kodors, "Jetpack Compose and XML layout rendering performance comparison," HET – International Scientific Conference. Disponible en: <https://doi.org/10.17770/het2021.25.6779>

Accedido: 14-dic-2025.

[4] R. M. Salman, M. A. Akbar y T. Afirianto, "Analisis Perbedaan Performa LazyGrid dan RecyclerView (Jetpack Compose)," Jurnal Teknologi Informasi dan Ilmu Komputer (JTIK). Disponible en: <https://doi.org/10.25126/jtiik.2025128498>

Accedido: 14-dic-2025.

[5] S. Sahadevan Mary, "Accelerating UI Loading with Jetpack Compose," Zenodo. Disponible en: <https://doi.org/10.5281/zenodo.14358150>

Accedido: 14-dic-2025.