Marketplace app

Programación Dirigida por Eventos

Nombres: Marcos Alonso Amor; Daniel García Yuste

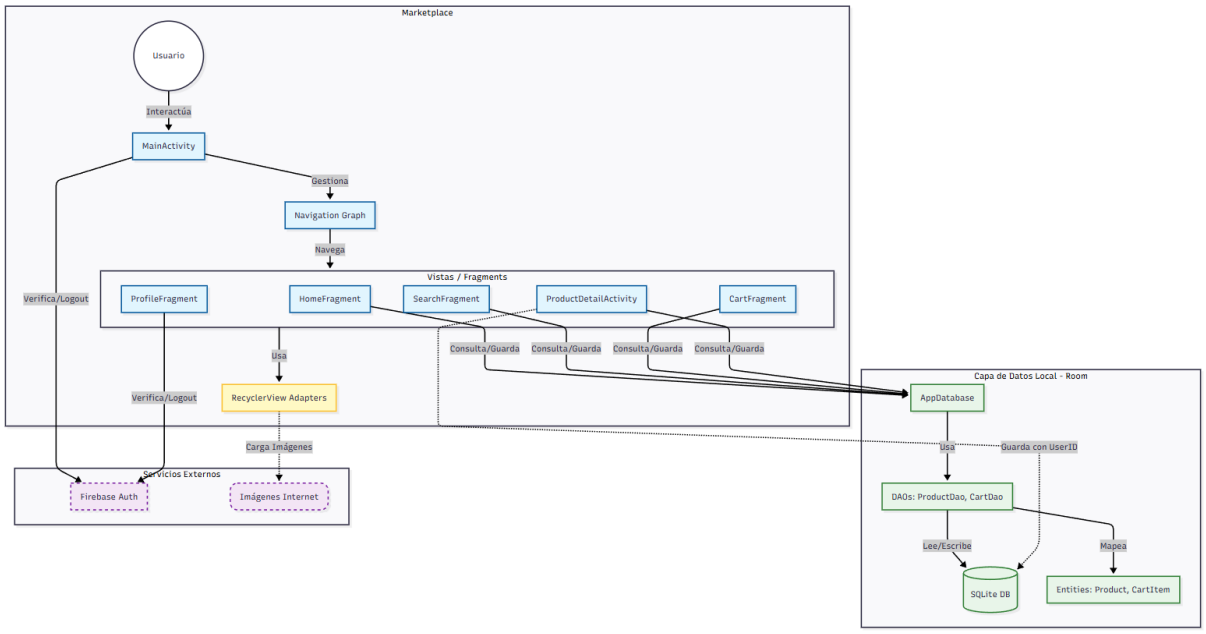
Fecha: 04/01/2026

Índice

Resumen*:*

Este proyecto consiste en el desarrollo de una prueba de conceptos (PCO) de una aplicación móvil de comercio electrónico, para el sistema operativo Android.
La aplicación está desarrollada en lenguaje Java, y esta permite a los usuarios registrarse, explorar un catálogo de productos, realizar búsquedas en tiempo real, gestionar un carrito de compras único para cada usuario y consultar su perfil.
Se ha utilizado una arquitectura basada en componentes de Navegación, persistencia de datos local mediante Room Database, y por último autentificación en la nube a través de Firebase Auth.

Diagrama de arquitectura:



El diagrama representa el flujo de funcionamiento de la aplicación, dividido en tres capas principales:

1. **Capa de Presentación (UI):**

La aplicación sigue el patrón Single Activity Architecture. Todo el ciclo de vida es gestionado por la MainActivity, que actúa como contenedor.

La navegación entre pantallas (Home, Search, Cart, Profile) se controla mediante el Navigation Component y un NavGraph, permitiendo un flujo fluido sin crear múltiples actividades pesadas.

Para visualizar las listas de productos, los Fragments utilizan Adapters y RecyclerViews, delegando la carga de imágenes a la librería Glide, que las obtiene de internet de forma asíncrona.

2. **Capa de Datos Local (Persistencia):**

El corazón de los datos es Room Database, que actúa como una capa de abstracción sobre SQLite.

La aplicación no escribe SQL directamente, sino que utiliza DAOs (Data Access Objects) para definir operaciones seguras (Insert, Query, Delete).

Las entidades Product y CartItem definen la estructura de las tablas en la base de datos local.

3. **Capa Externa y Lógica de Negocio (Integración):**

Firebase Auth gestiona la identidad del usuario.

El Flujo Clave (Multi-usuario): Cuando un usuario añade un producto al carrito en la UI, la aplicación solicita el UserID a Firebase. Este ID se inyecta en la entidad CartItem antes de guardarla en Room. De esta forma, aunque la base de datos es local, los datos están segmentados y protegidos lógicamente para cada usuario.

Descripción del trabajo realizado:

Funcionalidades incluidas:

1. **Autenticación de usuarios:**
    - Implementación de Login y Registro utilizando Firebase Auth
    - Control de seguridad: Validaciones de contraseña fuerte (mayúsculas, minúsculas y números) y formato email.
    - Restricción de acceso al MainActivity para aquellos usuarios que no estén registrados.

2. **Navegación Fluida:**
    - Uso de BottomNavigationView para un menú inferior.
    - Navegación mediante Jetpack Navigation Component para transiciones optimizadas entre pantallas.

3. **Catálogo y Búsqueda:**
    - Visualización de productos en cuadrícula (GridLayout) mediante RecyclerView.
    - Buscador en tiempo real: Barra de búsqueda que filtra productos en la base de datos SQL mediante consultas LIKE a medida que el usuario escribe.

4. **Gestión del carrito de compra:**

- Persistencia de datos local: El carrito del usuario se mantiene, aunque este cierre la aplicación.
- Soporte Multi-Usuario: Cada ítem guardado en el carrito guarda el userID de Firebase, permitiendo que varios usuarios usen el mismo dispositivo sin mezclar sus cestas de compra.
- Cálculo automático del precio total, y la posibilidad de eliminar ítems desde el carrito.

Partes no implementadas:

- **Pasarela de pagos reales**: El proceso para finalizar la compra es una simulación, ya que no hemos implementado ningún método de pago real dentro de esta práctica.
- **Backend remoto de productos**: Todos los productos se cargan localmente, en lugar de venir de una API REST externa que garantice la disponibilidad del producto.

Funcionalidades adicionales realizadas:

- **Sistema Multi-usuario Real:** Se mejoró el requisito inicial para que el carrito no solo fuese persistente, sino también, único para cada usuario.
- **Precarga inteligente de datos:** La app detecta si es la primera vez que se ejecuta y puebla la base de datos de forma automática.

Explicación de las decisiones tomadas:

A lo largo de este proyecto se tomaron las siguientes decisiones técnicas, para optimizar el funcionamiento de la aplicación.

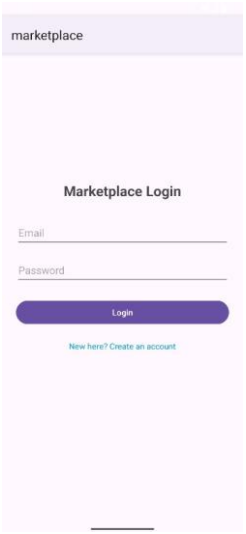
1. **Uso de Room vs SQLite puro:**
    - Se optó por la librería Room en lugar de gestionar SQLiteOpenHelper manualmente.
    - Motivo: Room ofrece verificación de consultas SQL en tiempo de compilación, reduce el código repetitivo y facilita la asignación de objetos java a tablas relacionales.

2. **Navigation Component vs Intent Transactions:**
    - Se decidió usar una sola actividad y múltiples Fragments.
    - Motivo: Es el estándar más moderno de Android. Con esto conseguimos reducir el consumo de memoria al no abrir múltiples actividades pesadas y facilita la gestión de la navegación.

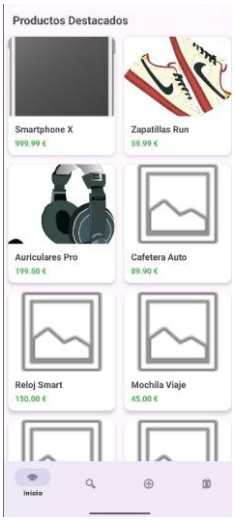
3. **Firebase Auth para el Login:**
    - Motivo: Decidimos delegar la seguridad y el almacenamiento a google, para así evitar vulnerabilidades de seguridad que podrían surgir al crear un sistema de login propio

4. **Gestión de la base de datos en el hilo principal:**
    - Se permite allowMainThreadQueries() en la configuración de Room
    - Justificación: Aunque a un nivel más elevado se usaron hilos secundarios, para esta actividad decidimos priorizar la velocidad de respuesta en la ui.

Visualización:

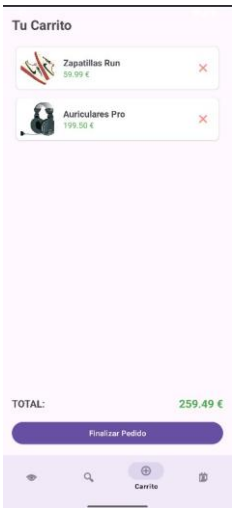


Login



Inicio

Perfil de Usuario



Carrito

Conclusiones

El proyecto ha resultado en una app funcional, estable y visualmente coherente con las guías de diseño. Se han integrado con éxito tecnologías como SQL local, NoSQL en nube, java nativo. Todo para poder crear una experiencia de usuario fluida.

El desarrollo de este código ha permitido profundizar en el ciclo de vida de los Fragments y la importancia de una arquitectura limpia.