Informe de la Vista: Carrito — La Montaña

Parcial 1 · Aplicaciones Móviles · Equipo: Herms — Tejero

Índice

1. Introducción y contexto general

La vista 'Carrito' forma parte del proyecto móvil 'La Montaña', una aplicación Android diseñada para gestionar pedidos de impresión y encuadernado. Esta pantalla permite al usuario visualizar los productos agregados desde el catálogo, modificar cantidades, eliminar ítems, simular la carga de archivos PDF para detección automática de páginas y realizar pedidos individuales o globales. Todo esto se realiza de forma local en memoria mediante una arquitectura sencilla, sin backend.

2. Tecnologías y lenguajes aplicados

• Lenguaje principal: Java.
• IDE: Android Studio.
• SDK: compileSdk 36, minSdk 24.
• Librerías principales: Material Components y ConstraintLayout.
• Arquitectura: Activity + modelo de datos en memoria (CartStore) + layouts XML.
• Patrón de datos: uso de singleton para el carrito compartido.
• Recursos organizados: strings.xml, colors.xml, dimens.xml.
• No se utiliza base de datos local ni backend; se apoya exclusivamente en el almacenamiento en memoria.

3. Descripción funcional de la vista Carrito

La vista Carrito (CartActivity) permite listar los productos que el usuario ha agregado desde el catálogo. Cada ítem muestra la miniatura del producto, su descripción, el precio unitario, el total por cantidad y los controles para modificar dicha cantidad (+/-). Cuando el producto admite detección por PDF (atributo 'copyBased'), se habilita un botón adicional 'Subir PDF' que simula la carga de un archivo y ajusta automáticamente la cantidad según el número de páginas detectadas.

Además, la vista ofrece un resumen del total general y dos acciones principales: 'Volver al catálogo' y 'Realizar todos los pedidos'. Esta última muestra una confirmación con el total acumulado antes de vaciar el carrito.

4. Mapa de archivos y responsabilidades

10. app/src/main/java/com/example/parcial_1/CartActivity.java

- Controlador principal de la vista Carrito.
- Infla dinámicamente el layout 'item_cart_detail' por cada CartItem en memoria.
- Gestiona los eventos de los botones +, -, Subir PDF, Realizar pedido, Realizar todos.
- Calcula y muestra el total general actualizado.

11. app/src/main/java/com/example/parcial_1/data/CartStore.java

- Clase Singleton que mantiene el estado global del carrito.
- Ofrece métodos para agregar, incrementar, decrementar, limpiar y obtener totales.
- Es compartida por MainActivity (Catálogo) y CartActivity.

12. app/src/main/java/com/example/parcial_1/model/CartItem.java

- Modelo que encapsula un producto y su cantidad actual en el carrito.
- Proporciona estructura simple para operaciones en CartStore.

13. app/src/main/res/layout/activity_cart.xml

- Layout raíz de la vista Carrito.
- Incluye Toolbar reutilizable, lista scrolleable de ítems, total general, botones principales y footer común.

14. app/src/main/res/layout/item_cart_detail.xml

- Plantilla para cada ítem del carrito.
- Define miniatura, textos descriptivos, precio unitario, total por ítem y botones de acción (+, -, subir PDF, realizar pedido).

5. Clases y métodos principales

A continuación, se describen los métodos más relevantes y su función:

• CartActivity.onCreate(Bundle): Inicializa la interfaz, configura la toolbar, listeners de botones y llama a renderCart().

• CartActivity.renderCart(): Infla dinámicamente cada ítem del carrito a partir de item_cart_detail.xml, enlazando datos y eventos.

• CartActivity.rebindRowAfterChange(View, Product): Actualiza la vista de un ítem tras modificar su cantidad.

• CartActivity.updateGrandTotal(): Recalcula el total general y actualiza la interfaz.

• CartActivity.onPlaceAllOrders(): Confirma y procesa todos los pedidos; vacía el carrito tras la simulación.

• CartStore.add(Product): Agrega un producto nuevo o incrementa su cantidad si ya existe.

• CartStore.inc(Product) / dec(Product): Aumenta o reduce la cantidad de un producto en memoria.

• CartStore.getTotalAmount() / getTotalQty(): Devuelve totales acumulados para reflejar en UI.

6. Flujo de uso típico (paso a paso)

- 1. El usuario llega al Carrito desde el botón 'Ver carrito' en el Catálogo.
- 2. CartActivity recupera los ítems desde CartStore y los muestra.
- 3. El usuario puede ajustar cantidades usando los botones '+' o '-'.
- 4. Si el producto es copyBased, puede presionar 'Subir PDF' para simular la carga de un archivo y actualizar la cantidad.
- 5. El total se recalcula instantáneamente tras cada cambio.
- 6. El usuario puede vaciar el carrito o presionar 'Realizar todos' para confirmar el pedido global.
- 7. Tras confirmar, el sistema limpia el carrito y actualiza la interfaz.

7. Relaciones y dependencias entre componentes

• CartActivity depende de CartStore para mantener los datos del carrito.
• CartStore contiene objetos CartItem, que a su vez referencian objetos Product.
• activity_cart.xml define los contenedores principales que CartActivity manipula.
• item_cart_detail.xml es inflado por CartActivity para representar cada producto.
• MainActivity (Catálogo) y CartActivity comparten CartStore, lo que garantiza coherencia del estado.

8. Buenas prácticas y validaciones implementadas

• Uso de Singleton (CartStore) para evitar inconsistencias entre Activities.
• Validación de cantidad mínima en CartStore (no permite valores negativos).
• Recalculo inmediato del total tras cada operación para mantener coherencia visual.
• Manejo de listeners de botones desacoplados y uso de LayoutInflater para optimizar memoria.
• Separación clara entre presentación (XML) y lógica (Java).