DOCUMENTACIÓN

ÍNDICE

Informe de la Vista Catálogo

1. Tecnologías, lenguajes y lógica general

La vista Catálogo está desarrollada en Android Studio utilizando el lenguaje Java. La interfaz se construye con componentes de Material Design y contenedores ConstraintLayout/LinearLayout. Se emplean recursos tipificados (strings, colors, dimens) para garantizar consistencia visual y mantenibilidad. Dependencias y SDK: compileSdk 36, minSdk 24; librerías Material Components y ConstraintLayout. La actividad principal (MainActivity) actúa como controlador de la pantalla de Catálogo.

Lógica principal de la vista:

- a) inicializa un listado de productos de ejemplo
- b) renderiza dinámicamente los ítems del catálogo inflando el layout de cada ítem
- c) maneja filtros por categoría (Todos/Impresiones/Encuadernado)
- d) al presionar “Agregar”, incorpora el producto al carrito compartido en memoria
- e) actualiza el badge/cantidad y total en el panel superior.

Lenguaje: Java (Android).
UI: ConstraintLayout + LinearLayout; MaterialToolbar; MaterialButton; TextView; ImageView.
Recursos: strings.xml, colors.xml, dimens.xml.
Plantillas XML reutilizables: include_app_bar.xml (toolbar) e include_footer.xml (footer).
Gestión de estado simple del carrito: clase singleton en memoria (CartStore) accesible desde Activities.
Navegación: el botón “Ver carrito” abre CartActivity redirigiendo a la vista carrito .

2. Mapa de archivos y responsabilidades

A continuación se detallan los archivos relevantes de la vista Catálogo, con su ruta, responsabilidades y relaciones con otros componentes del proyecto.

- app/src/main/java/com/example/parcial_1/**MainActivity.java**
  - Controlador de la vista Catálogo. Inicializa datos de ejemplo (seedMockData), configura filtros y renderiza la lista mediante inflado dinámico (item_catalog).
  - Maneja eventos: botones de filtro (Todos/Impresiones/Encuadernado), “Agregar”, “Vaciar carrito”, “Ver carrito”.
  - Actualiza resumen de carrito en el panel superior (cantidad total y monto total) leyendo del CartStore.
  - Relaciones: usa Product, Category; consulta/actualiza CartStore; se apoya en activity_catalog.xml e item_catalog.xml; navega a CartActivity.
  -
- app/src/main/java/com/example/parcial_1/data/**CartStore.java**
  - Singleton de carrito en memoria (sin backend). Permite agregar, incrementar/decrementar, setear cantidad y limpiar.
  - Provee totales (cantidad y monto) para actualizar la UI del Catálogo.

    - Relaciones: utilizado por MainActivity (para agregar/limpiar) y por CartActivity (para +/- y totales).

- app/src/main/java/com/example/parcial_1/model/**Product.java**
    - Entidad simple de producto: nombre, descripción, precio entero (ARS), categoría, recurso de imagen y flag copyBased (si puede detectar cantidad desde PDF).
    - Relaciones: instanciada por MainActivity (seedMockData); utilizada por CartStore y en la UI de item_catalog.

- app/src/main/java/com/example/parcial_1/model/**Category.java**
    - Enum de categorías: PRINT, BINDING. Se utiliza para filtros del Catálogo.
    - Relaciones: consultado por MainActivity al filtrar listas por categoría.

- app/src/main/java/com/example/parcial_1/model/**CartItem.java**
    - Composición producto + cantidad para el carrito.
    - Relaciones: producido y consumido por CartStore; leído por la UI del carrito (no por Catálogo). app/src/main/res/layout/activity_catalog.xml
    - Estructura principal del Catálogo: Toolbar reutilizable (include_app_bar), panel superior de carrito (resumen/cantidad/total), filtros y contenedor scrolleable del catálogo.
    - Define contenedores: llCatalogContainer para ítems renderizados dinámicamente; botones de filtro y de acciones; incluye footer común.
    - Relaciones: inflado por MainActivity; cada ítem se infla desde item_catalog.xml.

- app/src/main/res/layout/**item_catalog.xml**
    - Template de cada tarjeta de producto en la grilla/columna del catálogo.
    - Contiene miniatura, nombre, descripción, precio y botón “Agregar”.
    - Relaciones: inflado por MainActivity dentro del CatalogContainer.

- app/src/main/res/layout/**include_app_bar.xml**
    - MaterialToolbar con logo y menú (menu_top).
    - Pensado para reutilizarse en distintas pantallas.
    - Relaciones: incluido en activity_catalog.xml y activity_cart.xml. app/src/main/res/layout/include_footer.xml
    - Pie común con datos de contacto y redes.
    - Relaciones: incluido en activity_catalog.xml y activity_cart.xml

- app/src/main/res/menu/**menu_top.xml**
    - Menú superior con acción de navegación.
    - Relaciones: referenciado por include_app_bar.xml.

- app/src/main/res/values/**strings.xml**
    - Textos visibles: títulos, etiquetas de botones, formatos, placeholders.
    - Relaciones: usados por layouts y por código (getString).

- app/src/main/res/values/**colors.xml**

    - Paleta de marca (marrones/naranjas/verdes), alias de apoyo (background, text, card, muted).
    - Relaciones: aplicada en layouts (fondos, textos, botones).

- app/src/main/res/values/**dimens.xml**
    - Tokens de espaciado (xs/sm/md/lg), tamaños de texto y radio de borde para consistencia.
    - Relaciones: aplicados en layouts para márgenes/padding y tipografía

- app/src/main/**AndroidManifest.xml**
    - Declara MainActivity como launcher y registra CartActivity.
    - Relaciones: punto de entrada de la app; navegación entre activities.

- app/**build.gradle.kts**
    - Configura compileSdk/minSdk, Java 11, y dependencias (Material, ConstraintLayout, appcompat).
    - Relaciones: determina APIs disponibles y componentes UI utilizados en los layouts/código.

3. Clases y métodos principales de la vista Catálogo

- **MainActivity.onCreate(Bundle)**: Inicializa la interfaz, toolbar, listeners de botones y carga inicial de productos.
- **MainActivity.seedMockData()**: Crea la lista inicial de productos con atributos simulados (nombre, descripción, precio, categoría e imagen).
- **MainActivity.renderCatalog(List<Product>)**: Infla dinámicamente item_catalog.xml para cada producto y configura listeners del botón 'Agregar'.
- **MainActivity.updateCartUi()**: Actualiza los valores de cantidad total y monto total del carrito en pantalla.
- **CartStore.add(Product)**: Agrega un producto al carrito o incrementa su cantidad si ya existe. Requiere validar que el producto no sea nulo.
- **CartStore.clear()**: Vacía el carrito, utilizado al presionar el botón 'Vaciar carrito'.
- **CartStore.getTotalAmount()**: Calcula el monto total multiplicando cantidad * precio de cada producto.
- **CartStore.getTotalQty()**: Devuelve la cantidad total de ítems en el carrito.
- **Product (constructor):** Define los campos name, desc, price, category, imageRes y copyBased. Todos deben respetar sus tipos; especialmente, 'price' debe ser un entero positivo.

4.Flujo de caso de uso típico

1. El usuario abre la app y visualiza la vista Catálogo con un panel superior que resume el carrito (cantidad y total).
2. El usuario utiliza los filtros (Todos / Impresiones / Encuadernado) para acotar la lista
3. El usuario revisa cada tarjeta (nombre, descripción, precio) y presiona “Agregar” en el producto deseado.
4. La aplicación incrementa el carrito en memoria (CartStore) y actualiza de inmediato el badge y el total en el panel superior.
5. El usuario puede repetir el proceso con múltiples productos, limpiar el carrito, o navegar a “Ver carrito” (opcional para la demo del Catálogo). Notas de demo: Los productos se cargan localmente (seedMockData) solo para la práctica del parcial, sin backend ni base de datos.