PROYECTO BLESS

Edgar Felipe Beltrán Cárdenas
edbeltran@unal.edu.co

1. Información General

- **Nombre del Proyecto:** ProyectoBless
- **Tecnologías Usadas:** Android Studio, Java, Firebase (Authentication y Realtime Database), XML, Glide (para imágenes), Material Design Components.
- **Versión de SDK mínima:** API 27 (Android 8.1)
- **Versión de compilación:** API 35
- **Build configuration language:** Groovy DSL (Gradle)
- **Repositorio GitHub:** https://github.com/Felipe-un/ProyecotBless3.git

2. Objetivo General

Desarrollar una aplicación móvil en Android que permita la gestión de inventario de productos, el control de acceso de usuarios (administrador y empleados), el registro y gestión de clientes, la creación y seguimiento de pedidos, y la visualización de comentarios de productos, facilitando la operación y mejora continua del negocio.

3. Alcance del Producto

Este aplicativo móvil impacta directamente la gestión interna de un negocio, ofreciendo herramientas para:

- **Gestión de Usuarios:** Autenticación segura y registro de nuevos usuarios con roles definidos (administrador, empleado).
- **Gestión de Inventario:** Funcionalidades completas de Crear, Leer, Actualizar y Eliminar (CRUD) productos, incluyendo detalles como nombre, stock, precio, categoría y URL de imagen.
- **Gestión de Clientes:** Registro y mantenimiento de una base de datos de clientes con información de contacto.
- **Gestión de Pedidos:** Creación de nuevos pedidos directamente desde la aplicación, asociándolos a clientes registrados y permitiendo la selección de productos del inventario. Visualización y actualización del estado de los pedidos.
- **Visualización de Comentarios:** Acceso a comentarios de productos (recopilados externamente) con información de calificación y filtrado por categoría, para facilitar el análisis y la toma de decisiones.

3.1. Análisis de Requisitos:

Se han identificado y abordado los requisitos clave para la gestión de inventario, clientes, pedidos y comentarios, así como el control de acceso basado en roles.

3.2. Diseño y Arquitectura:

La aplicación sigue una arquitectura modular, utilizando Activities como contenedores principales y Fragments para las secciones de contenido, gestionadas por un ViewPager2 y TabLayout. Se emplea View Binding para una interacción segura y eficiente con los layouts.

3.3. Programación:

Desarrollada en Java, utilizando las APIs de Android y las librerías de Firebase para backend (Authentication y Realtime Database). Se han implementado adaptadores (RecyclerView.Adapter) para la visualización eficiente de listas.

3.4. Pruebas Unitarias:

(Pendiente de implementación) Se planifican pruebas unitarias para asegurar la correcta funcionalidad de los componentes individuales.

3.5. Documentación:

Este documento sirve como parte de la documentación del proyecto, describiendo la estructura, funcionalidades y requisitos.

3.6. Mantenimiento:

La arquitectura modular y el uso de Firebase facilitan el mantenimiento y la escalabilidad futura de la aplicación.

4. Entorno Operativo

- **Sistema Operativo:** Android (API 27+).
- **Base de Datos:** Firebase Realtime Database para el almacenamiento de datos de usuarios, productos, clientes, pedidos y comentarios.
- **Autenticación:** Firebase Authentication para la gestión de usuarios y roles.
- **Entorno de Desarrollo:** Android Studio, Gradle.

5. Requerimientos Funcionales

A continuación, se detallan los requisitos funcionales implementados en la aplicación:

RF1: La app debe permitir iniciar sesión a administradores y empleados.

- **Implementación:** LoginActivity permite la autenticación con email y contraseña. Tras el éxito, verifica el rol del usuario en Firebase Realtime Database y redirige a MainActivity.

RF2: Solo el administrador puede registrar nuevos usuarios.

- **Implementación:** El LoginActivity no tiene una opción de registro público. El botón para "Registrar Nuevo Usuario" solo es visible en MainActivity si el usuario logueado tiene el rol de "admin". La RegisterActivity maneja la creación de usuarios en Firebase Authentication y guarda su rol en Realtime Database.

RF3: Los usuarios pueden visualizar, agregar, modificar o eliminar productos del inventario.

- **Implementación:**
  - **Visualizar:** InventarioFragment muestra la lista de productos en un RecyclerView.
  - **Agregar:** Un FloatingActionButton (visible solo para "admin") en InventarioFragment lanza ProductoFormActivity para crear nuevos productos.
  - **Modificar/Eliminar:** Botones "Editar" y "Eliminar" (visibles solo para "admin") en cada ítem del RecyclerView en InventarioFragment permiten modificar (lanzando ProductoFormActivity con datos existentes) o eliminar productos directamente de Firebase.

RF4: Los pedidos se listan en la app y se pueden crear desde ella, mostrando el cliente asociado.

- **Implementación:**
  - PedidosFragment muestra una lista de pedidos en un RecyclerView. Cada pedido muestra el nombre del cliente asociado, estado y fecha.
  - Un FloatingActionButton (visible solo para "admin") en PedidosFragment lanza PedidoFormActivity para crear nuevos pedidos.
  - PedidoFormActivity permite seleccionar un cliente de una lista, añadir productos del inventario (a través de ProductSelectionActivity) y definir el estado del pedido.
  - Los pedidos se guardan en dos ubicaciones en Firebase: una colección global /pedidos y una subcolección /clientes/{clienteId}/pedidos para facilitar la consulta por cliente.

RF5: La app podrá recolectar todos estos comentarios, definir la categoría y dividirlos para que al administrador o empleado les sea más fácil saber qué mejorar en cada producto.

- **Implementación:**
  - ComentariosFragment muestra una lista de comentarios en un RecyclerView.
  - Los comentarios incluyen texto, calificación (1-5) y la categoría del producto.
  - Un Spinner en ComentariosFragment permite filtrar los comentarios por categoría de producto, facilitando el análisis.
  - La recolección inicial de comentarios se asume que se realiza a través de un Google Form externo y se integra a Firebase.

RF Adicionales Implementados:

- **Gestión de Clientes:**
  - ClienteFormActivity permite registrar y editar clientes (nombre, email, teléfono).
  - Un FloatingActionButton (visible solo para "admin") en PedidosFragment lanza ClienteFormActivity.
- **Visualización de Pedidos por Cliente:**
  - ClientePedidosActivity muestra una lista de todos los pedidos asociados a un cliente específico, accesibles desde el botón "Ver Pedidos del Cliente" en cada ítem de pedido en PedidosFragment.

- **Edición de Estado de Pedido:**
  - Un botón "Editar Estado" (visible solo para "admin") en cada ítem de pedido en PedidosFragment permite actualizar el estado del pedido a través de un diálogo, actualizando ambas ubicaciones en Firebase.
- **Eliminación de Pedido:**
  - Un botón "Eliminar" (visible solo para "admin") en cada ítem de pedido en PedidosFragment y ClientePedidosActivity permite eliminar el pedido, eliminándolo de ambas ubicaciones en Firebase.

6. Requerimientos No Funcionales

6.1. Eficiencia

- La carga de datos desde Firebase (productos, pedidos, comentarios, clientes) se realiza de forma asíncrona y en tiempo real gracias a los ValueEventListener, asegurando que la UI se actualice rápidamente.
- El uso de RecyclerView y sus adaptadores optimiza el rendimiento y el consumo de memoria para la visualización de listas grandes.

6.2. Seguridad y Lógica de Datos

- **Autenticación:** Uso de Firebase Authentication para el control de acceso.
- **Roles:** Implementación de un sistema de roles (admin/empleado) para restringir funcionalidades críticas (registro de usuarios, CRUD de inventario, gestión de clientes y pedidos).
- **Consistencia de Datos:** Los pedidos se guardan y eliminan en dos ubicaciones de Firebase (/pedidos y /clientes/{clienteId}/pedidos) para mantener la integridad y facilitar consultas diversas.

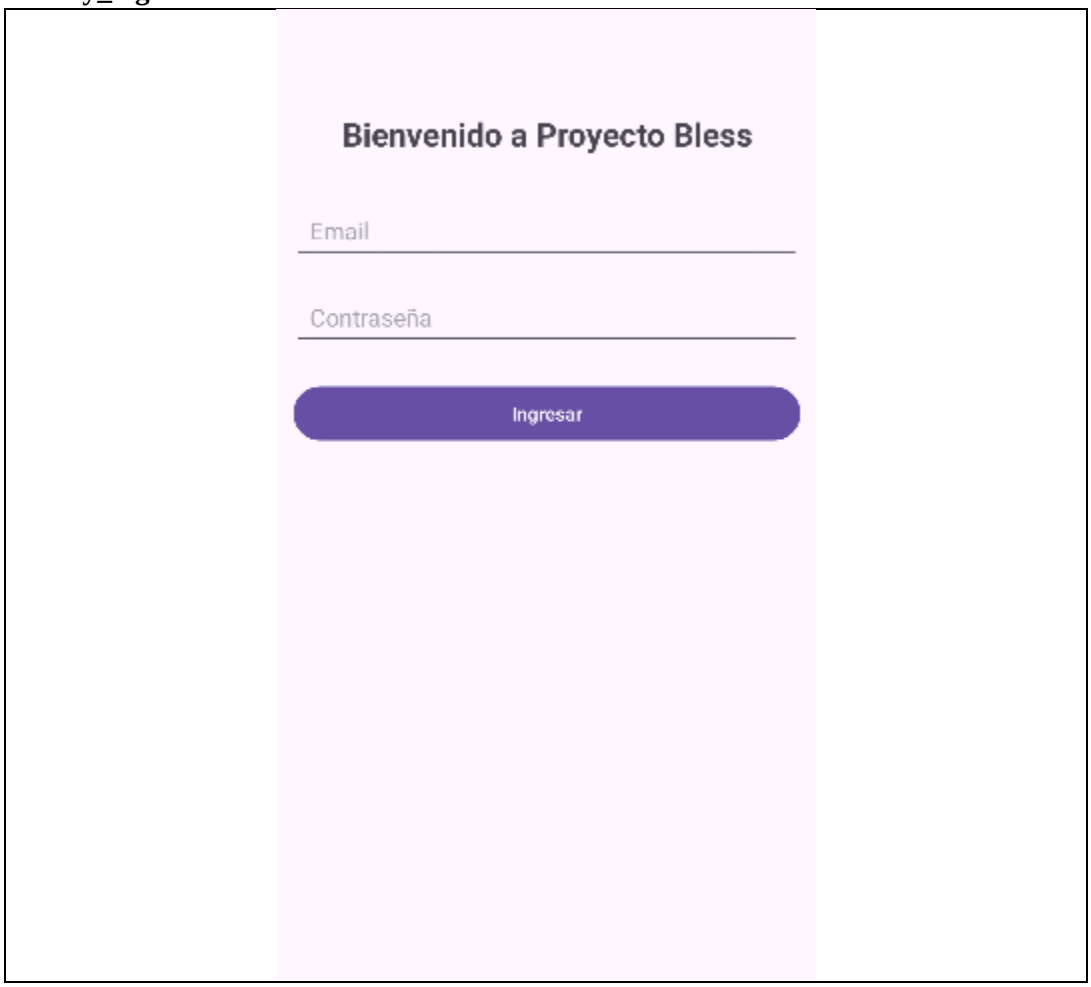
6.3. Hardware

- La aplicación está diseñada para funcionar en dispositivos Android con API 27 (Android 8.1) o superior.

7. Estructura del Proyecto

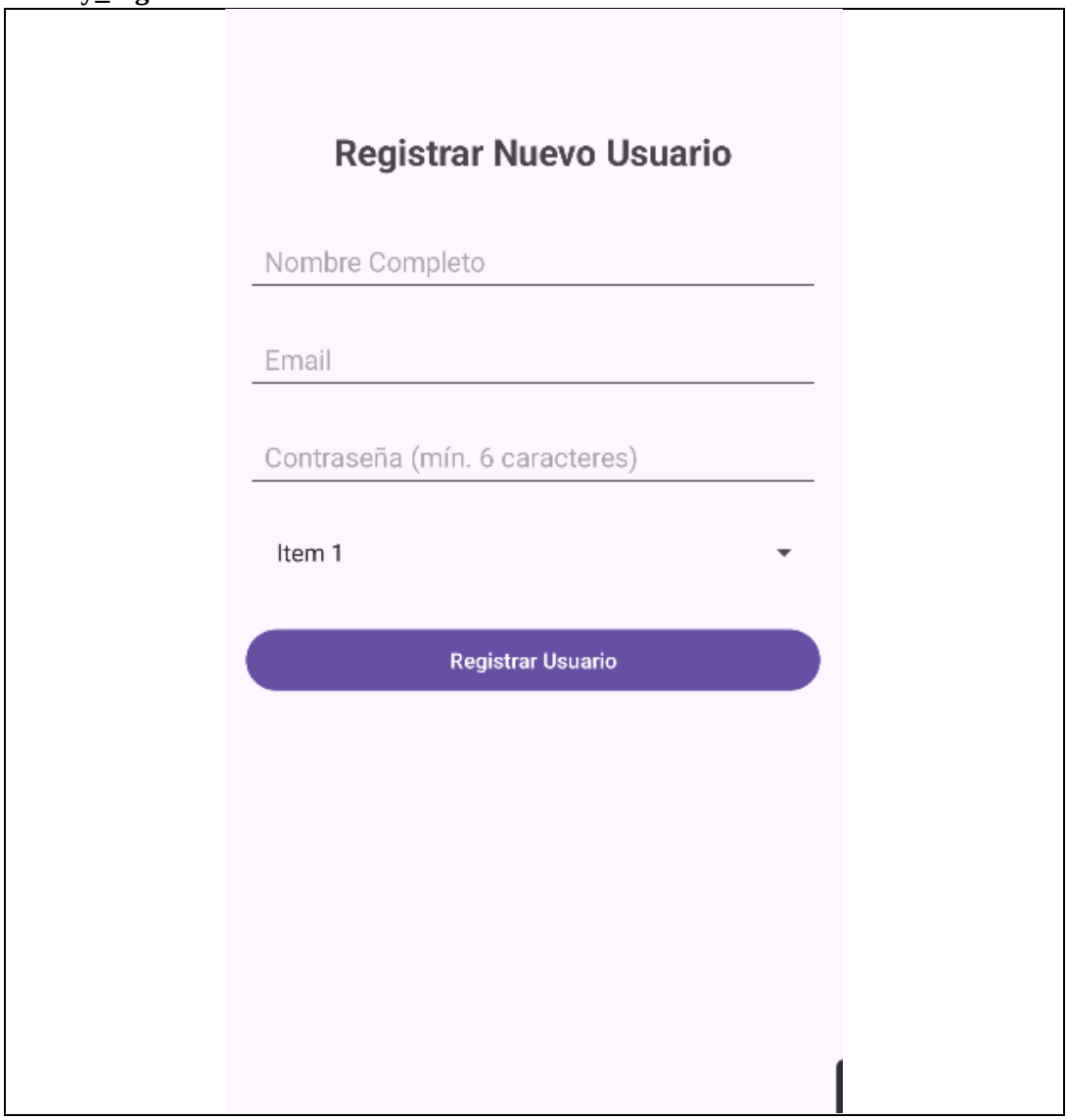
Desglosaremos paso a paso cada uno de los layouts con sus principales funcionalidades y métodos asociados:

A. activity_login.xml



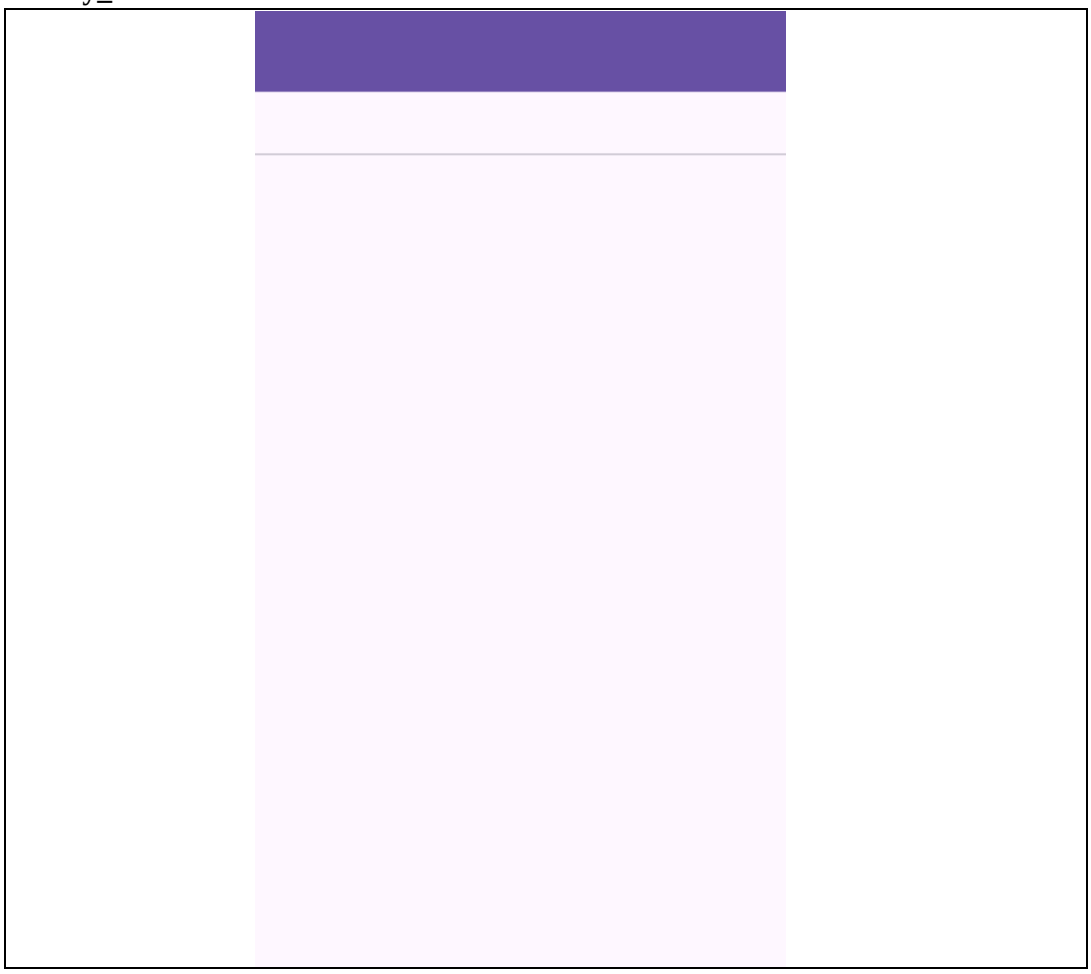
- **Clase Asociada:** LoginActivity
- **Métodos Principales:**
    - onCreate(): Inicializa la vista, configura los listeners para los botones de login y registro.
    - loginUser(): Maneja la lógica de autenticación de usuario con Firebase Authentication (signInWithEmailAndPassword).
    - checkUserRoleAndRedirect(): Consulta el rol del usuario en Firebase Realtime Database y redirige a MainActivity o muestra un mensaje de error si no tiene rol.

B. activity_register.xml



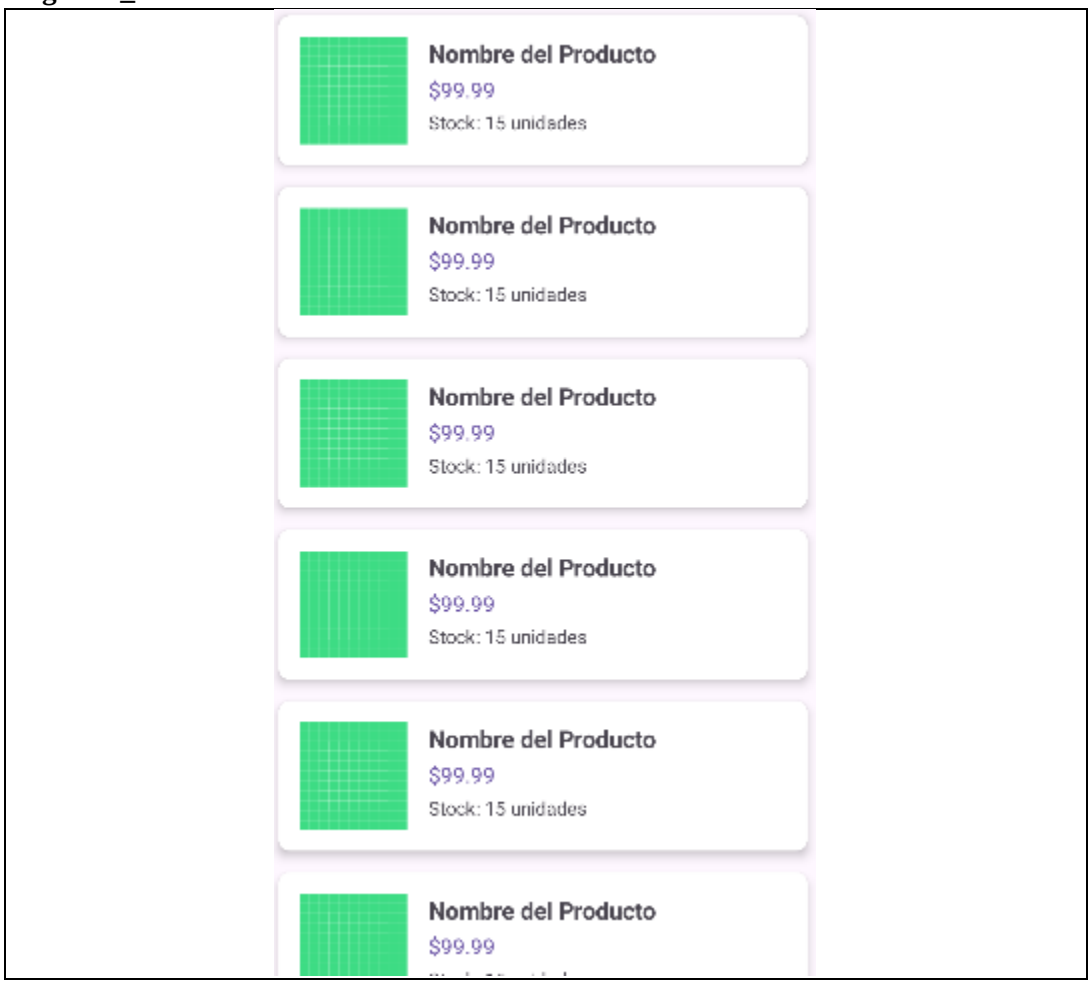
- **Clase Asociada:** RegisterActivity
- **Métodos Principales:**
    - onCreate(): Inicializa la vista, configura los listeners para el botón de registro.
    - registerUser(): Crea un nuevo usuario en Firebase Authentication (createUserWithEmailAndPassword) y guarda su información (nombre, email, rol) en Firebase Realtime Database.
    - validateInput(): Realiza validaciones de los campos de entrada (email, contraseña, confirmación de contraseña, rol).

C. activity_main.xml



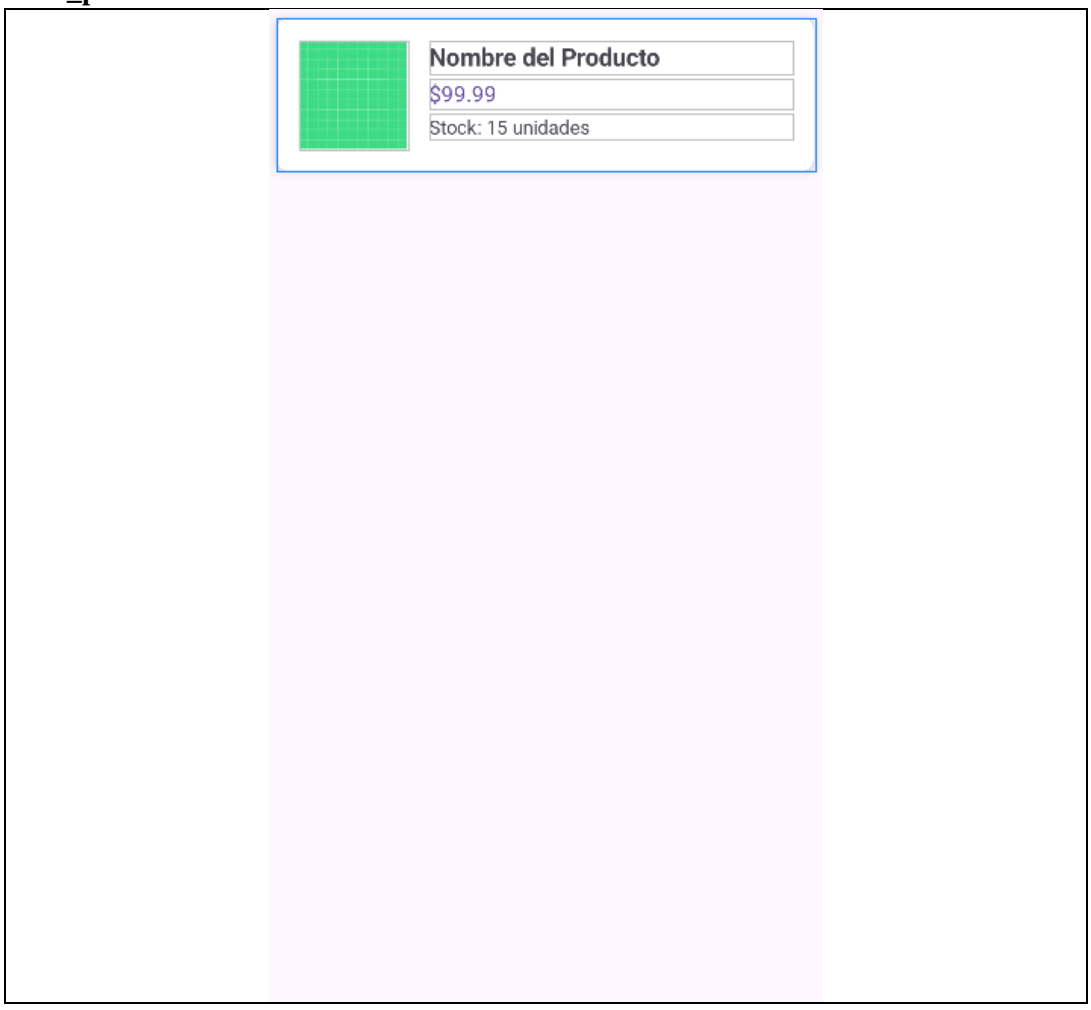
- **Clase Asociada:** MainActivity
- **Métodos Principales:**
    - onCreate(): Configura la Toolbar, el TabLayout y el ViewPager2. Inicializa el HomeViewPagerAdapter.
    - checkUserRole(): Obtiene el rol del usuario actual de Firebase Realtime Database y lo pasa a los fragments según sea necesario.
    - setupTabLayout(): Conecta el TabLayout con el ViewPager2 y sus fragments.
    - logout(): Cierra la sesión del usuario de Firebase Authentication y redirige a LoginActivity.

D. fragment_inventario.xml



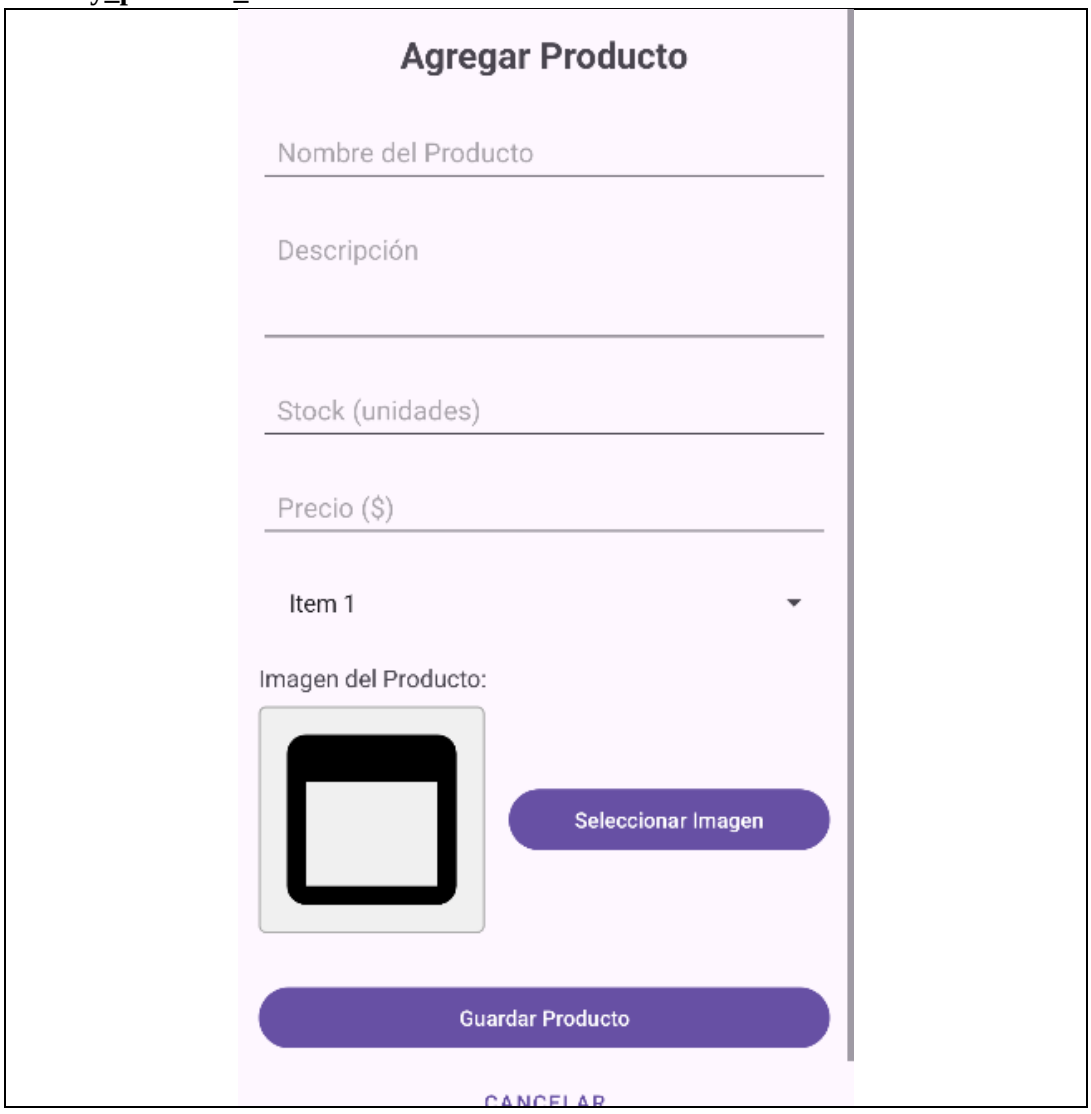
- **Clase Asociada:** InventarioFragment
- **Métodos Principales:**
    - onCreateView(): Infla el layout, inicializa el RecyclerView y su ProductoAdapter. Configura el FloatingActionButton (fab_add_product) para lanzar ProductoFormActivity.
    - loadProducts(): Escucha cambios en la colección "productos" de Firebase Realtime Database y actualiza la lista de productos en el RecyclerView.
    - onEditClick(Producto producto) (implementado de ProductoAdapter.OnItemActionListener): Lanza ProductoFormActivity para editar el producto seleccionado.
    - onDeleteClick(Producto producto) (implementado de ProductoAdapter.OnItemActionListener): Elimina el producto de Firebase Realtime Database.

E. item_producto.xml



- **Clase Asociada:** ProductoAdapter
- **Métodos Principales:**
    - onBindViewHolder(): Vincula los datos de un objeto Producto a las vistas del ítem.
    - Maneja la visibilidad de los botones "Editar" y "Eliminar" según el rol del usuario (userRole).
    - Configura los OnClickListener para los botones "Editar" y "Eliminar", llamando a los métodos correspondientes en la interfaz OnItemActionListener (implementada por InventarioFragment).

F. activity_producto_form.xml



- **Clase Asociada:** ProductoFormActivity
- **Métodos Principales:**
    - onCreate(): Inicializa la vista, configura los spinners, y los listeners para los botones de guardar, cancelar y seleccionar imagen.
    - checkPermissionAndOpenImageChooser(): Verifica y solicita permisos de almacenamiento para acceder a la galería.
    - openImageChooser(): Lanza el Intent para seleccionar una imagen de la galería.
    - loadProductData(): Carga los datos de un producto existente para edición, incluyendo la imagen con Glide.
    - saveProduct(): Valida los campos, inicia el proceso de subida de imagen si se seleccionó una nueva.

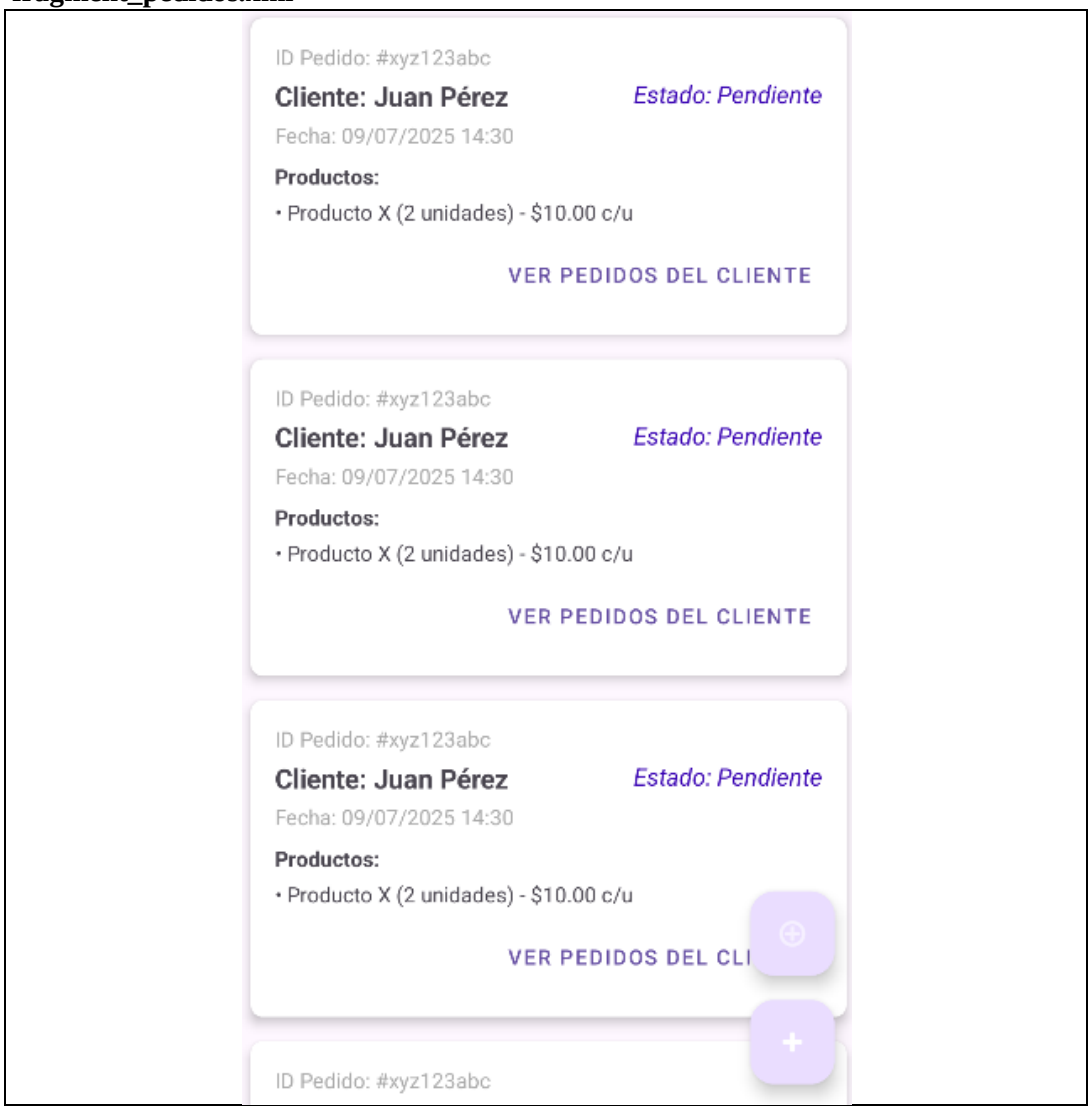
- uploadImageToFirebaseStorage(): Sube la imagen seleccionada a Firebase Storage, obtiene la URL de descarga.
- saveProductToDatabase(): Guarda los datos del producto (incluyendo la URL de la imagen) en Firebase Realtime Database (para creación o actualización).

G. item_pedido.xml

- **Clase Asociada:** PedidoAdapter
- **Métodos Principales:**
  - onBindViewHolder(): Vincula los datos de un objeto Pedido a las vistas del ítem.
  - Obtiene el nombre del cliente asociado utilizando el clienteMap y lo muestra.
  - Añade dinámicamente los productos dentro del pedido.
  - Maneja la visibilidad de los botones "Editar Estado" y "Eliminar" según el rol del usuario (userRole).

- Configura los OnClickListener para los botones "Ver Pedidos del Cliente", "Editar Estado" y "Eliminar", llamando a los métodos correspondientes en la interfaz OnItemClickListener.

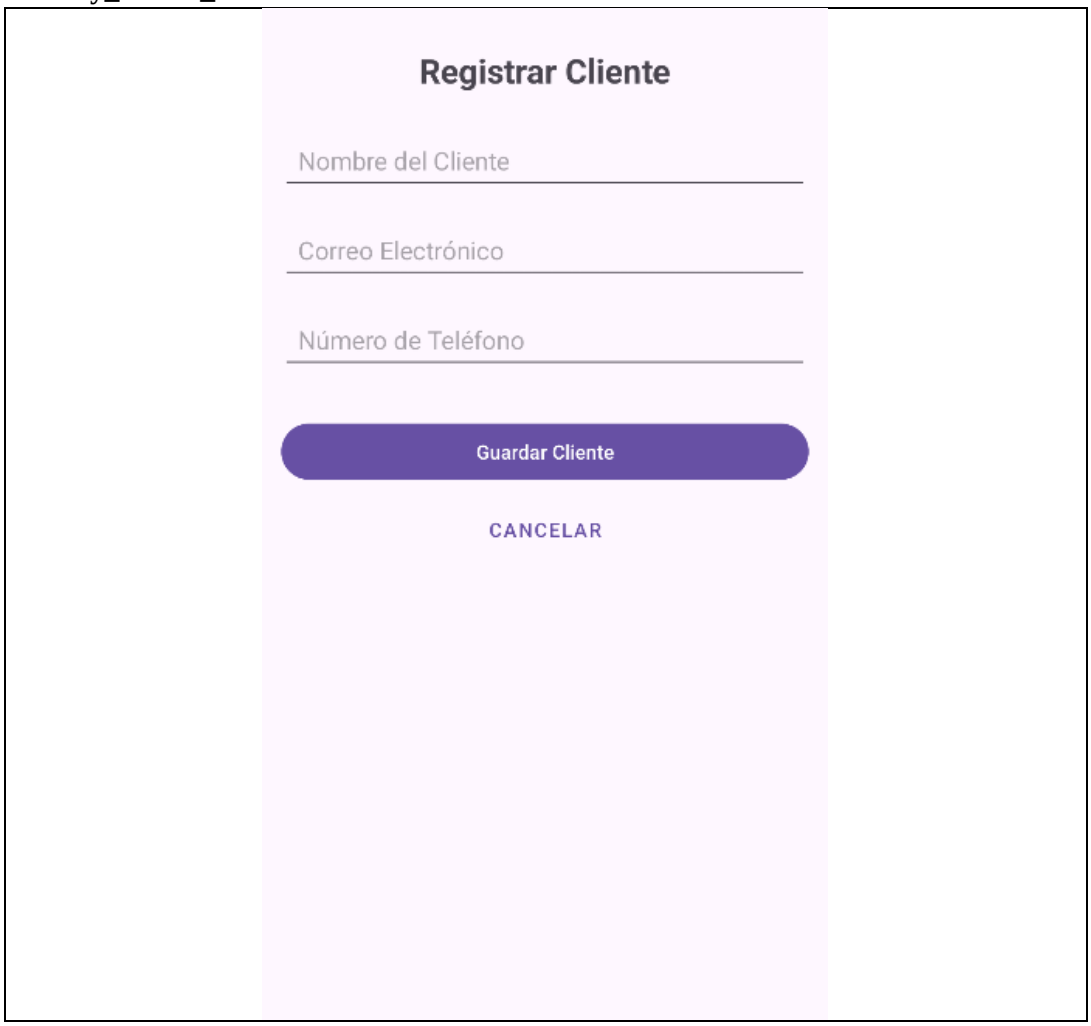
H. fragment_pedidos.xml



- **Clase Asociada:** PedidosFragment
- **Métodos Principales:**
  - onCreateView(): Infla el layout, inicializa el RecyclerView y su PedidoAdapter. Configura los FloatingActionButtons (fab_add_cliente y fab_add_pedido) para lanzar ClienteFormActivity y PedidoFormActivity respectivamente.
  - loadClientesAndPedidos(): Carga primero los clientes y luego los pedidos de Firebase Realtime Database, actualizando el PedidoAdapter.

- onViewClientOrdersClick(String clienteId) (implementado de PedidoAdapter.OnItemClickListener): Lanza ClientePedidosActivity para ver los pedidos de un cliente específico.
- onEditStatusClick(Pedido pedido) (implementado de PedidoAdapter.OnItemClickListener): Muestra un MaterialAlertDialogBuilder con un Spinner para seleccionar el nuevo estado del pedido.
- updatePedidoStatus(Pedido pedido, String newStatus): Actualiza el estado del pedido en Firebase Realtime Database (en la colección global y en la del cliente).
- onDeleteOrderClick(Pedido pedido) (implementado de PedidoAdapter.OnItemClickListener): Muestra un diálogo de confirmación y luego llama a deletePedido().
- deletePedido(Pedido pedido): Elimina el pedido de Firebase Realtime Database (en la colección global y en la del cliente).

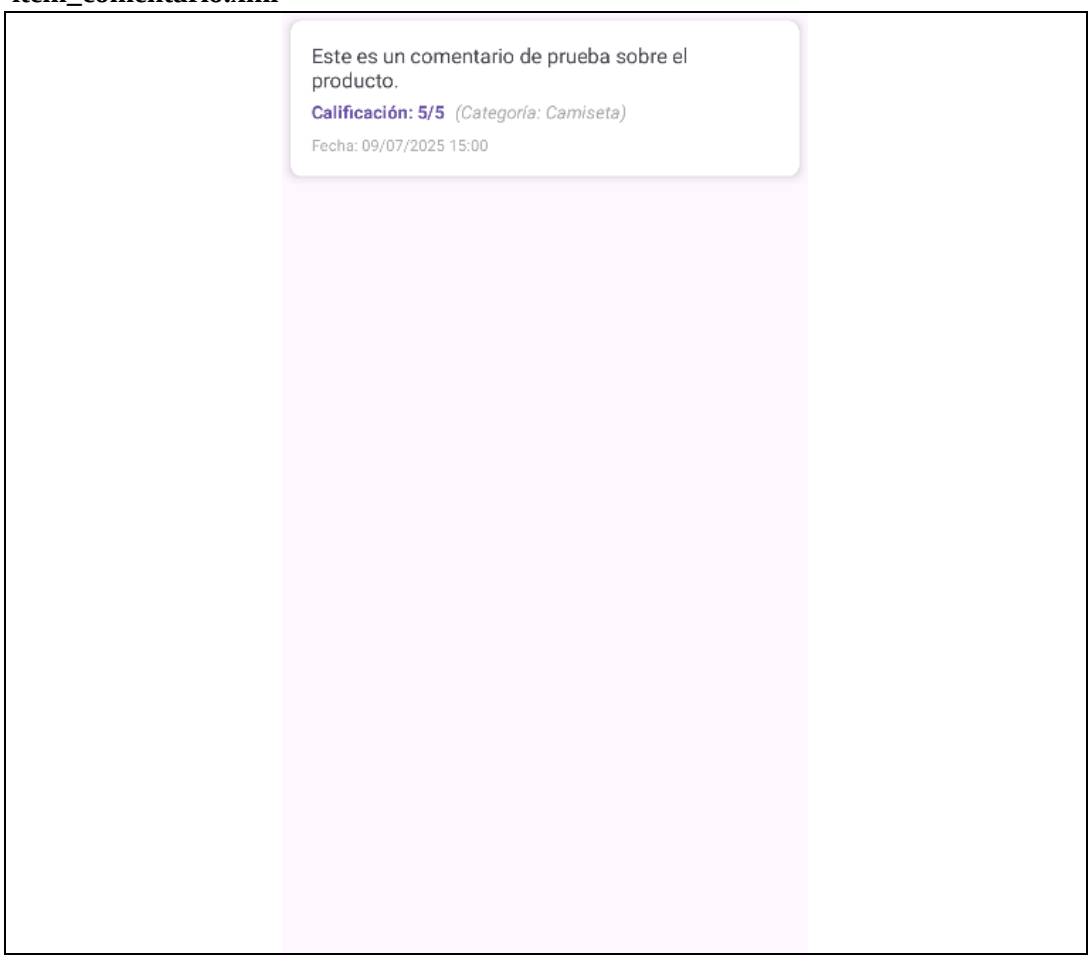
I. activity_cliente_form.xml



- **Clase Asociada:** ClienteFormActivity

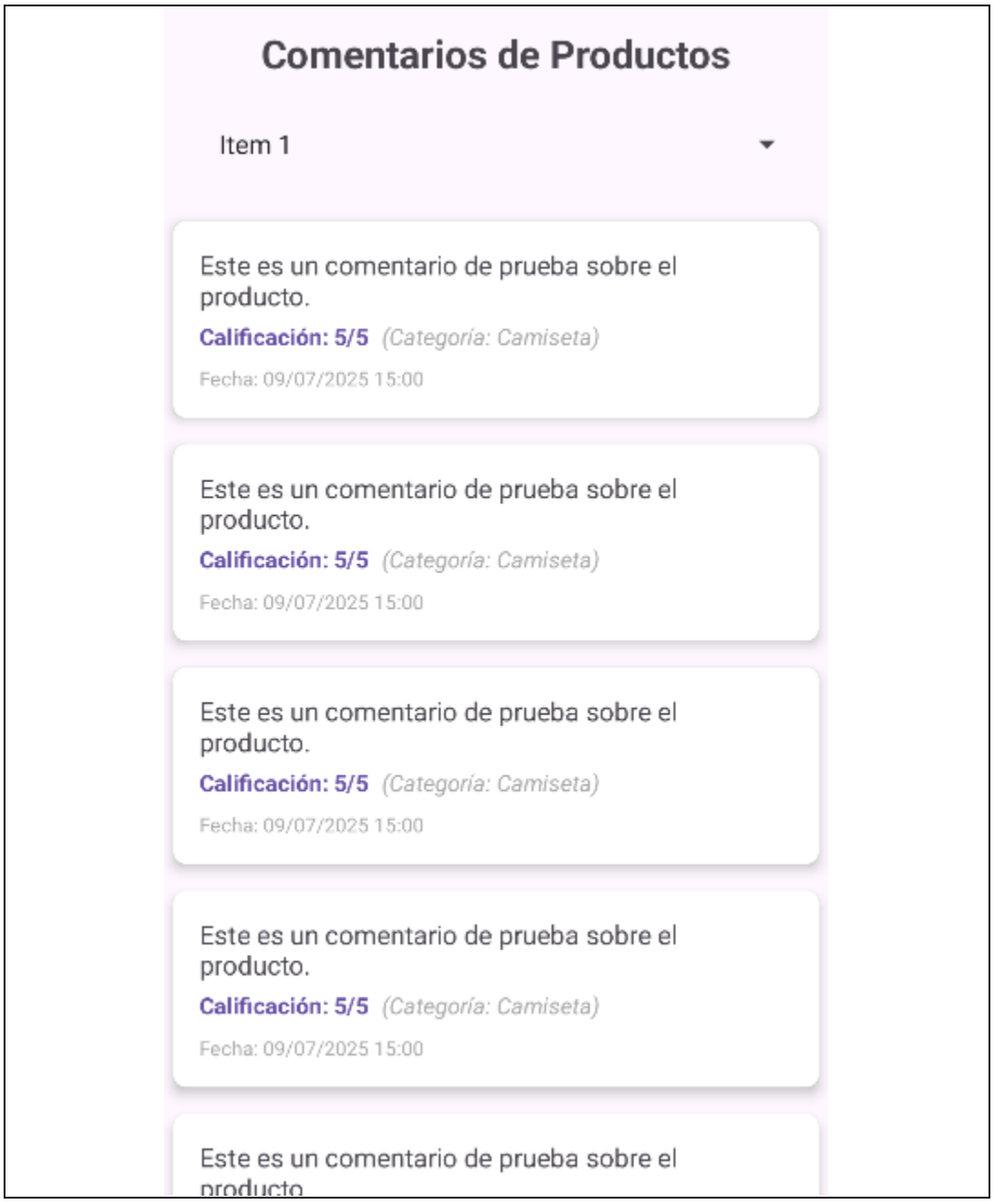
- **Métodos Principales:**
    - onCreate(): Inicializa la vista y los listeners para los botones de guardar y cancelar.
    - loadClienteData(): Carga los datos de un cliente existente para edición.
    - saveCliente(): Valida los campos y guarda la información del cliente en Firebase Realtime Database (creación o actualización).

J. item_comentario.xml



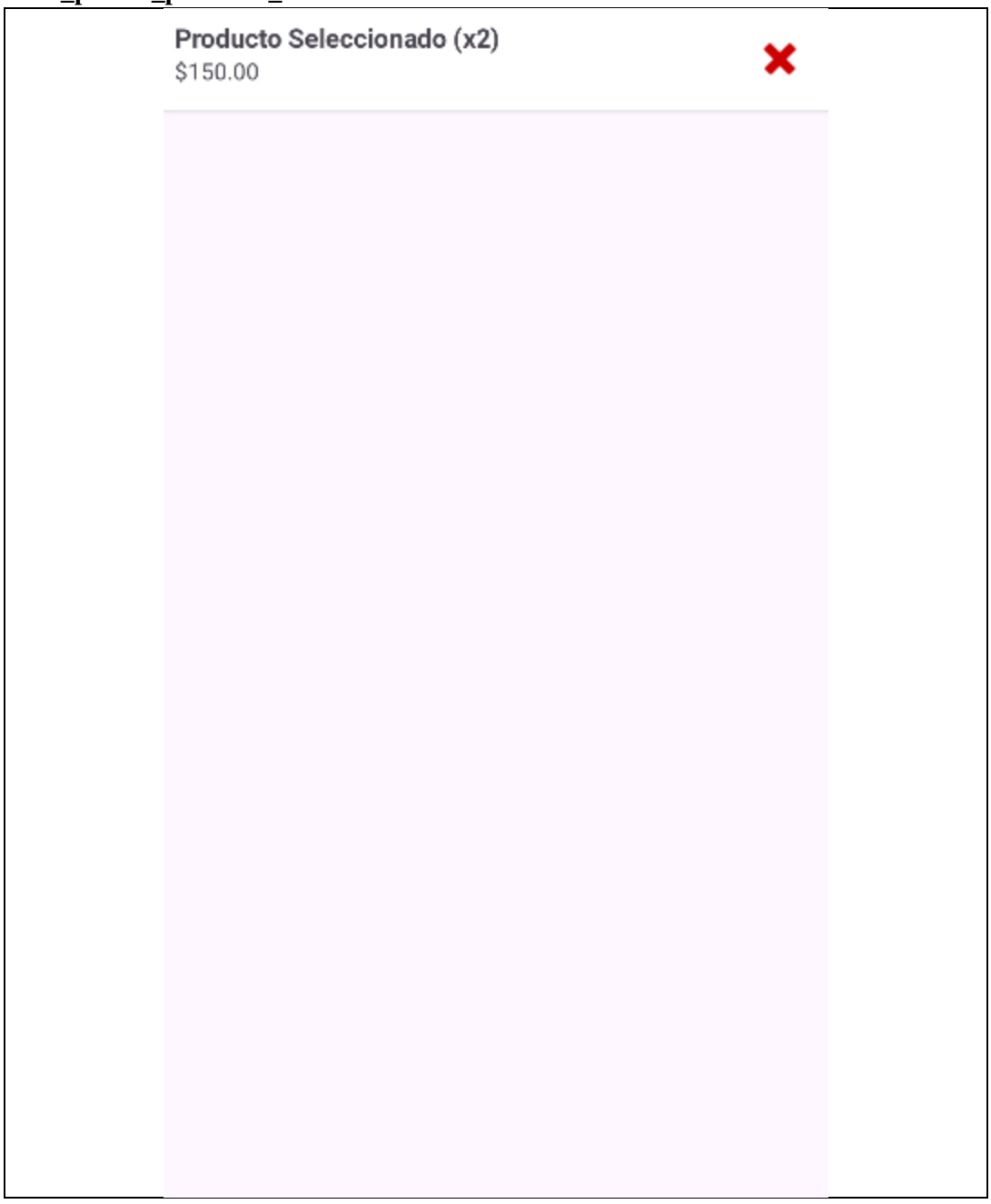
- **Clase Asociada:** ComentarioAdapter
- **Métodos Principales:**
    - onBindViewHolder(): Vincula los datos de un objeto Comentario a las vistas del ítem, mostrando el texto, calificación, categoría y fecha.

K. fragment_comentarios.xml



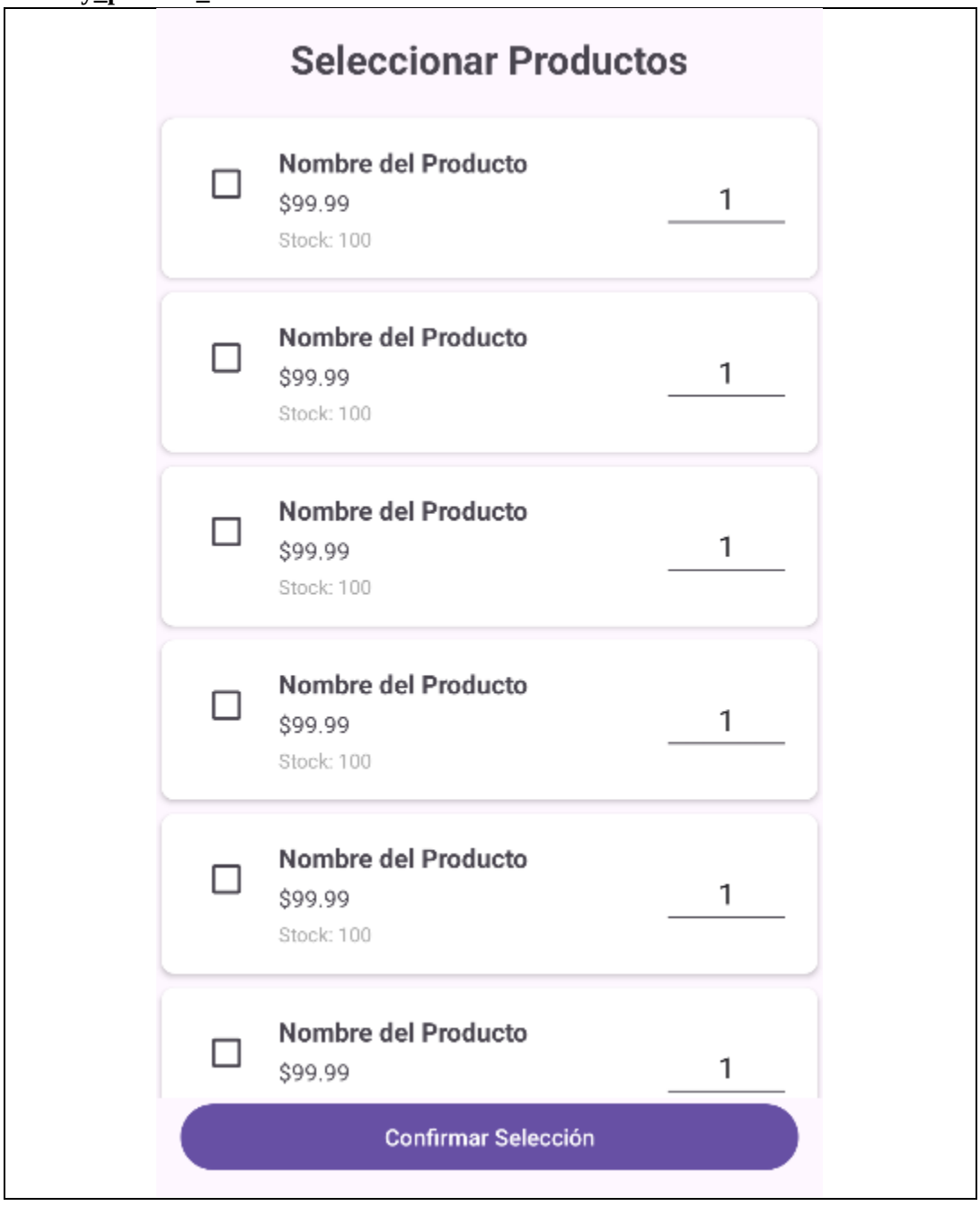
- **Clase Asociada:** ComentariosFragment
- **Métodos Principales:**
    - onCreateView(): Infla el layout, inicializa el RecyclerView y su ComentarioAdapter. Configura el Spinner de filtro de categorías.
    - loadComments(): Carga los comentarios de Firebase Realtime Database, aplicando un filtro por categoría si se selecciona uno en el Spinner.

L. item_pedido_producto_seleccionado.xml



- **Clase Asociada:** PedidoProductoSeleccionadoAdapter
- **Métodos Principales:**
    - onBindViewHolder(): Vincula los datos de un Pedido.PedidoItem a las vistas del ítem.
    - Configura el OnClickListener para el botón de eliminar, llamando a listener.onRemoveItem().

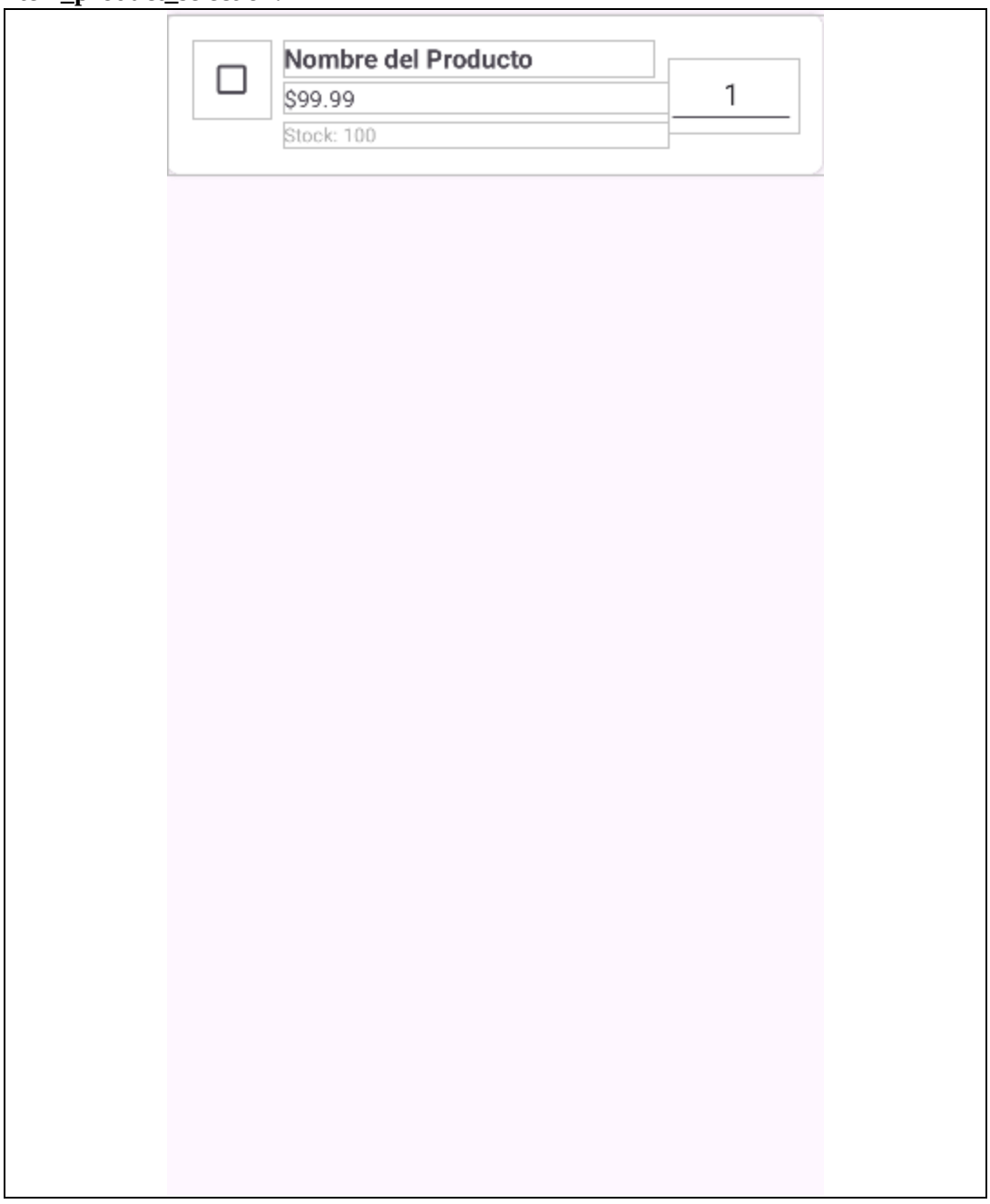
M. activity_product_selection.xml



- **Clase Asociada:** ProductSelectionActivity
- **Métodos Principales:**
    - onCreate(): Inicializa la vista, el RecyclerView y su ProductSelectionAdapter. Configura el listener para el botón "Confirmar Selección".
    - loadAvailableProducts(): Carga todos los productos del inventario desde Firebase Realtime Database.

- confirmSelection(): Recopila los productos seleccionados y sus cantidades del adaptador, los empaqueta en un Intent y devuelve el resultado a PedidoFormActivity.

N. item_product_selection.xml



- **Clase Asociada:** ProductSelectionAdapter
- **Métodos Principales:**
  - onBindViewHolder(): Vincula los datos de un Producto a las vistas del ítem.
  - Maneja el estado del CheckBox (seleccionado/deseleccionado).

- Configura un TextWatcher para el EditText de cantidad, validando que no exceda el stock disponible y actualizando la cantidad seleccionada internamente.
- getSelectedProducts(): Retorna una lista de Pedido.PedidoItems que representan los productos que el usuario ha seleccionado con sus cantidades.
-

O. activity_cliente_pedidos.xml

- **Clase Asociada:** ClientePedidosActivity

- **Métodos Principales:**
    - onCreate(): Obtiene el clienteId y userRole del Intent. Inicializa el RecyclerView y PedidoAdapter.
    - loadClienteDetailsAndOrders(): Carga los detalles del cliente para mostrar su nombre en el encabezado y luego llama a loadClientOrders().
    - loadClientOrders(): Carga los pedidos específicos de ese cliente desde la subcolección clientes/{clienteId}/pedidos en Firebase Realtime Database.
    - onDeleteOrderClick(Pedido pedido) (implementado de PedidoAdapter.OnItemClickListener): Muestra un diálogo de confirmación y luego llama a deletePedido().
    - deletePedido(Pedido pedido): Elimina el pedido de Firebase Realtime Database (en la colección global y en la del cliente).