MarketSales — Evolución del flujo de ventas (documentación de sesión)

Fecha: 2025-08-15 20:44

Este documento recoge, de forma exhaustiva y en español, \*\*todo el trabajo realizado\*\* durante la sesión para la app \*MarketSales\*.  
Está pensado para que puedas reabrir el proyecto dentro de un tiempo, continuar el desarrollo sin perder contexto y entender los porqués de cada decisión.  
  
La documentación está organizada cronológicamente pero también por áreas (navegación, pantallas, datos, etc.).  
Incluye rutas, módulos y nombres de clases/composables exactos, más decisiones arquitectónicas y “pitfalls” que resolvimos.  
  
\*\*Resumen muy corto del objetivo:\*\* construir el flujo de ventas híbrido \*offline‑first\* con TPV manual y por productos,  
incluyendo método de pago, pantalla de cambio, recibo centrado, carrito editable, y un resumen de ventas por mercadillo;  
además de ajustes de diseño y de persistencia en Room + sincronización con Firebase.

# 1. Punto de partida

- App existente con navegación básica (mercadillos, alta/edición, etc.).

- Persistencia en Room y algunos repositorios básicos.

- Requisitos nuevos:

1) Flujo de venta completo (creación de ticket, selección método de pago, cambio en efectivo, recibo, etc.).

2) Modelo \*offline-first\*: guardar \*\*siempre\*\* en Room y sincronizar a Firebase cuando sea posible.

3) Integrar la \*\*pestaña de productos\*\* en “Ventas” con un grid 3×N y añadir al ticket con pulsaciones.

4) Diseño: teclado TPV, botón “Realizar cargo” en una línea, moneda desde `ConfigurationManager`, badges estables, etc.

5) Pantalla \*\*Resumen de ventas\*\* desde PantallaMercadillos (BottomBar) con listado y total.

# 2. Navegación y rutas (MainActivity)

Archivo: `app/src/main/java/es/nuskysoftware/marketsales/MainActivity.kt`

Se consolidaron rutas y se creó un \*\*wrapper\*\* para ventas que carga el mercadillo por ID:

- `mercadillos`

- `ventas/{mercadilloId}` → `PantallaVentasWrapper()` obtiene el mercadillo y pinta `PantallaVentas` con ese contexto.

- `metodo\_pago/{mercadilloId}/{totalFmt}` → `PantallaMetodoPago()`.

- `cambio/{mercadilloId}/{totalFmt}` → `PantallaCambio()`.

- `enviar\_recibo/{mercadilloId}/{totalFmt}/{metodo}` → `PantallaEnviarRecibo()`.

- `carrito/{mercadilloId}` → `PantallaVentasCarrito` (edición y borrado de líneas antes de finalizar).

- `resumen/{mercadilloId}` → `PantallaResumenVentas` (desde BottomBar en Mercadillos).

\*\*Bug resuelto:\*\* crash al navegar a `metodo\_pago/...` por ruta sin `mercadilloId`. Se unificó a `metodo\_pago/{mercadilloId}/{totalFmt}`.

\*\*Bug de enums:\*\* coexistían dos `MetodoPago` (UI y data). Se \*\*eliminó el de UI\*\* y se usa el del paquete `data.repository` en toda la app.

# 3. ViewModel de Ventas y lógica TPV

Archivo: `app/src/main/java/es/nuskysoftware/marketsales/ui/viewmodel/VentasViewModel.kt`

- Estado `VentasUiState` con: pestaña activa, líneas del ticket, total, importe actual de teclado TPV (“0,00”), descripción, filtros y errores.

- \*\*Teclado estilo TPV\*\* (2 decimales fijos que “corren” de derecha a izquierda):

- `onDigitoPresionado(digito: String)`

- `onDobleDecimalPresionado()`

- `onBorrarDigito()`

- Añadir línea \*\*manual\*\* valida descripción e importe y la agrega a `lineasTicket` recalculando el total.

- `finalizarVenta(metodoPago: MetodoPago)`:

- Valida que haya líneas y total > 0.

- Guarda en \*\*Room\*\* usando `VentasRepository.guardarVenta(...)` y \*\*luego\*\* (best effort) envía a Firebase.

- Limpia el estado del carrito (resetea importe/desc y borra líneas) para que el \*\*badge\*\* del carrito y el botón queden a cero.

- `inicializar(mercadilloId, usuarioId)`: necesario para persistir la venta con el contexto correcto.

- Errores: se propagan en `uiState.error` y se muestran con `Snackbar`.

# 4. Modelo de datos y DAOs de ventas

\*\*Entidades\*\* añadidas/empleadas:

- `ReciboEntity` (`app/src/main/java/es/nuskysoftware/marketsales/data/local/entity/ReciboEntity.kt`)

- Campos principales: `idRecibo`, `idMercadillo`, `idUsuario`, `fechaHora`, `metodoPago`, `totalTicket`, `estado`.

- `idRecibo` \*\*formato\*\*: `RChhmmss-<IDIOMA>` (ej.: `RC153045-ES`). Importante:

- El \*\*idioma\*\* se toma de \*\*Configuración\*\* a través de `ConfiguracionDao.getConfiguracion()` (`idioma`), \*\*no\*\* del `Locale` del dispositivo.

- `LineaVentaEntity` (`app/src/main/java/es/nuskysoftware/marketsales/data/local/entity/LineaVentaEntity.kt`)

- `idLinea` (string), `idRecibo`, `idMercadillo`, `idUsuario`, `numeroLinea`, `tipoLinea`, `descripcion`, `idProducto?`, `cantidad`, `precioUnitario`, `subtotal`, `idLineaOriginalAbonada?`.

\*\*DAOs\*\*:

- `RecibosDao`:

- `obtenerRecibosPorMercadillo(idMercadillo): Flow<List<ReciboEntity>>`

- `insertarRecibo(...)`, `actualizarRecibo(...)`, etc.

- `LineasVentaDao` (versión final usada):

- Lecturas por recibo/mercadillo.

- \*\*Para numeración\*\*: `@Query("SELECT MAX(numeroLinea) FROM lineas\_venta WHERE idMercadillo = :idMercadillo") suspend fun obtenerMaxNumeroLinea(idMercadillo: String): Int?`

- La \*\*numeración se reinicia por mercadillo\*\*. Es decir, la primera línea de cada mercadillo es \*\*0001\*\* y se incrementa secuencialmente sin reiniciar dentro del mismo mercadillo.

- Inserciones/borrados en lote.

\*\*AppDatabase\*\* (`version = 7`)

- Se eliminó `fallbackToDestructiveMigration()`. Se definió la \*\*estrategia de migraciones reales\*\* para futuras versiones (añadir `Migration(V7, V8)` cuando haya cambios de esquema).

- Configuración por defecto en `onCreate` sin destruir datos.

# 5. VentasRepository (offline‑first)

Archivo: `app/src/main/java/es/nuskysoftware/marketsales/data/repository/VentasRepository.kt`

- `guardarVenta(idMercadillo, lineas, metodoPago, total) -> Result<String>`:

1) Obtiene `usuarioId` (FirebaseAuth) y `idioma` desde `ConfiguracionDao`.

2) Genera `idRecibo = "RC" + hhmmss + "-" + idiomaUpper`.

3) Pide a `LineasVentaDao` el `maxNumeroLinea` del mercadillo para seguir numeración (\*\*0001, 0002, …\*\*).

El campo `idLinea` puede ser UUID, pero el \*\*campo de negocio\*\* y orden es `numeroLinea`.

4) Inserta \*\*primero en Room\*\*: `ReciboEntity` + `LineaVentaEntity` (con `numeroLinea` consecutivo).

5) Intenta replicar en \*\*Firebase\*\* (best effort): si falla, \*\*no revierte\*\* (el dato ya está en Room).

- Métodos auxiliares: obtener ventas por mercadillo, obtener líneas por recibo, crear abonos (línea con cantidades negativas y recibo de abono).

# 6. Pantallas del flujo de venta

### 6.1. `PantallaVentas`

- Título con flecha back (`ic\_arrow\_left`) → `popBackStack()`.

- \*\*Tabs\*\*: “Venta Manual” y “Productos”.

- En “Manual” (`PestanaVentaManual`):

- Campo de \*\*importe\*\* grande, campo de \*\*descripción\*\* y \*\*teclado TPV\*\*.

- Botón \*\*“Añadir más ventas”\*\* agrega la línea al carrito.

- Botón negro \*\*“Realizar cargo XX,XX{moneda}”\*\* en \*\*una sola línea\*\* y con números tabulares (`fontFeatureSettings = "tnum"`).

- Botón \*\*carrito\*\* flotante grande (ícono \*\*grande\*\* + \*\*badge\*\* estable). El badge se arregló usando `Box(Modifier.align(Alignment.TopEnd))` y suficiente `padding` para evitar clipping.

- En “Productos” (`PestanaVentaProductos`):

- Grid \*\*3 columnas\*\* con \*\*cards cuadrados\*\*: nombre, precio y \*\*subrayado\*\* con color de la categoría.

- \*\*Tap\*\*:

- 1ª pulsación: añade línea con `cantidad = 1`.

- Pulsaciones adicionales sobre el mismo producto: incrementan la \*\*cantidad\*\* en la misma línea (recalculando `subtotal`).

- Botonera inferior idéntica (Realizar cargo + Carrito).

### 6.2. `PantallaMetodoPago`

- Recibe `totalFormateado` y callbacks.

- Muestra \*\*Total\*\* e importe \*\*más grande\*\* (28→32sp), y 3 botones grandes:

- 💶 \*\*Efectivo\*\*

- 📲 \*\*Bizum\*\*

- 💳 \*\*Tarjeta\*\*

- Back con `ic\_arrow\_left`.

### 6.3. `PantallaCambio` (solo efectivo)

- Tipografía \*\*mucho más grande\*\*: total, campo “entregado”, \*\*cambio\*\* (36sp ExtraBold).

- Opción de añadir un \*\*teclado numérico\*\* específico (mismo estilo que TPV) para introducir el entregado.

- Valida “importe insuficiente”. Al confirmar puede navegar a recibo o directamente finalizar.

### 6.4. `PantallaEnviarRecibo`

- \*\*Recibo centrado\*\* (card en el centro, ancho ~92%):

- \*\*Código grande\*\* `RChhmmss-ID`.

- \*\*Lugar\*\*.

- \*\*Fecha dd/MM/yy\*\* y \*\*hora HH:mm\*\*.

- \*\*Tabla de líneas\*\*: `CANTIDAD | DESCRIPCIÓN | PRECIO | TOTAL`.

- \*\*Total\*\* alineado a la \*\*izquierda\*\* (según requerimiento).

- \*\*Forma de pago\*\*.

- Botones: \*\*Enviar\*\* (pospuesto) + \*\*Finalizar venta\*\* (limpia el VM y vuelve a `PantallaVentas`).

### 6.5. `PantallaVentasCarrito` (edición del ticket vigente)

- Lista de \*\*cards horizontales\*\* por línea: cantidad (editable), descripción, \*\*precio unitario\*\*.

El \*\*total de línea\*\* es calculado y no editable.

- Acciones: \*\*Eliminar línea\*\*, \*\*Aceptar\*\* (aplica cambios al VM) y \*\*Cancelar\*\* (descarta cambios).

- \*Offline‑first\*: los cambios se consolidan al \*\*guardar la venta\*\*; hasta entonces es estado local del VM.

# 7. Moneda y formato

- Se eliminó el símbolo “€” hardcodeado en favor de `ConfigurationManager.moneda`.

- Utilidad `MonedaUtils.formatearImporte(double, moneda)` para producir `12,34€` (formato acordado: cantidad seguida del símbolo, sin espacio).

- Se aplicó a:

- Botón \*\*Realizar cargo\*\*,

- Pantalla \*\*Productos\*\* (precios),

- Pantalla \*\*Cambio\*\*, \*\*Recibo\*\* y \*\*Resumen\*\*.

# 8. Ajustes de diseño importantes

- \*\*TopAppBar\*\* con colores del tema y back `ic\_arrow\_left` donde se pidió.

- \*\*Teclado TPV\*\*:

- Botones blancos, texto negro, \*\*borde circular verde\*\* (`colorScheme.primary`).

- Botón \*\*borrar\*\* con borde \*\*rojo\*\* (`colorScheme.error`).

- \*\*Realizar cargo\*\*: una línea, tipografía 18sp, números tabulares (`tnum`).

- \*\*Badges\*\* (carrito y en cards de productos): se corrigió el \*\*offset\*\* y el \*\*zIndex\*\* para que \*\*no se recorten\*\*.

# 9. Pantalla Mercadillos (BottomBar + Próximos)

Archivos:

- `ui/components/BottomBarMercadillo.kt`

- `ui/pantallas/PantallaMercadillos.kt`

- Se añadió una \*\*BottomBar\*\* con 3 botones: \*\*Ventas\*\*, \*\*Gastos\*\*, \*\*Resumen\*\* (íconos y colores).

Si hay \*\*varios mercadillos en curso\*\*, aparece un \*\*diálogo\*\* de selección antes de navegar.

- El \*\*calendario\*\* queda \*\*fijo\*\* y \*\*debajo\*\* va la lista de \*\*Próximos mercadillos\*\* con \*\*scroll\*\* independiente (requisito).

Se reemplazó el “próximo mercadillo” único por una \*\*lista\*\* con contador (Badge) y, si hay más de tres, un indicador “+N”.

- \*\*Pasados\*\*: se anotó como requisito (cuando están cerrados/pendientes, deben abrirse en modo \*\*solo lectura\*\* con botones “Arqueo”/“Asignar saldo”).

\*\*Quedan anotadas\*\* estas tareas para un ciclo posterior.

# 10. Resumen de ventas por mercadillo

Archivo: `app/src/main/java/es/nuskysoftware/marketsales/ui/pantallas/PantallaResumenVentas.kt`

- Cabecera con \*\*lugar del mercadillo\*\*.

- Lista \*\*scrollable\*\* de líneas (card estrecha): `cantidad`, `descripción`, \*\*total de línea\*\*; debajo “PU: precio”.

- \*\*Icono de forma de pago\*\* en cada línea (a la \*\*izquierda\*\*, dentro del Row principal).

Se resolvió un problema de composición donde el icono quedaba fuera; ahora forma parte del `Row` inicial junto a un `CantidadChip`.

- Barra inferior con \*\*Total de ventas\*\* (suma de `subtotal`).

- Los íconos dependen de `metodoPago` del \*\*recibo\*\* al que pertenece la línea (se mapearon por `idRecibo`).

# 11. Errores típicos que resolvimos

- \*\*Rutas duplicadas / sin parámetros\*\* → crash al navegar. Solución: unificar parámetros y \*\*URI.encode\*\* en importes con comas/€.

- \*\*Enums duplicados\*\* (`MetodoPago`) en paquetes distintos → comparaciones “siempre falsas”. Solución: \*\*una sola enum\*\* (la del repositorio).

- \*\*Badges\*\* invisibles o cortados → asegurar `Box` contenedor, `align(Alignment.TopEnd)`, `padding` extra y `zIndex` si hace falta.

- \*\*Formato de recibo\*\*: el \*\*idioma\*\* debe proceder de \*\*Configuración\*\* (Room), \*\*no\*\* de `Locale` del dispositivo.

- \*\*fallbackToDestructiveMigration()\*\*: se \*\*quitó\*\* para no arriesgar datos. Se documentó cómo añadir \*\*migraciones reales\*\*.

# 12. Qué falta / siguientes pasos

1) \*\*Enviar recibo\*\* (compartir/WhatsApp/Email): generar PDF/imagen o texto y abrir `Intent.ACTION\_SEND`.

2) \*\*Pestaña Productos\*\*: búsqueda y filtros avanzados; contador sobre cards con cantidad en carrito.

3) \*\*Carrito\*\*: atajos de +/– en cada línea; confirmaciones y undo.

4) \*\*Sincronización\*\*: cola de pendientes y reintentos con conectividad; conflictos de versión.

5) \*\*Mercadillos pasados\*\*: abrir en modo lectura + botones “Arqueo total” / “Asignar saldo” según estado.

6) \*\*Migraciones\*\*: cada cambio de esquema → añadir `Migration(Vx, Vy)` y pruebas de actualización.

7) \*\*Tests\*\*: unit tests para `VentasViewModel` y `VentasRepository` (especialmente numeración y formato de `idRecibo`).

# Anexo A — Rutas clave

• mercadillos

• ventas/{mercadilloId}

• metodo\_pago/{mercadilloId}/{totalFmt}

• cambio/{mercadilloId}/{totalFmt}

• enviar\_recibo/{mercadilloId}/{totalFmt}/{metodo}

• carrito/{mercadilloId}

• resumen/{mercadilloId}

# Anexo B — Iconos método de pago

efectivo → ic\_pago\_efectivo

bizum → ic\_pago\_bizum

tarjeta → ic\_pago\_tarjeta

default → ic\_pago\_default