**Hito 1 — Marco base de Mercadillos (sin romper nada)**

**1) Objetivo del hito**

* Separar responsabilidades en un **MercadillosManager** (reglas de negocio/estados/selecciones) y un **MercadilloRepository** (lectura/sync offline-first para mercadillos).
* Mantener el comportamiento actual de UI, pero **hacerlo fiable/recalculado siempre**.
* **No** tocar AuthRepository (ya queda: “descarga masiva solo si Room vacío y Premium”).
* **Sin implementación de arqueo** (solo placeholder), **sin** funcionalidades nuevas de edición (solo reglas y recálculo).

**2) Alcance exacto**

**2.1 MercadillOS (datos y sync)**

* **Lectura offline-first (confirmada):**
  + Si hay **pendientes locales** → *push* a Firebase y luego *pull selectivo* de mercadillos → **merge en Room** (nunca reemplazo total).
  + Si **no hay pendientes** → *pull selectivo* de mercadillos → **merge en Room**.
  + La pantalla **siempre** lee desde Room.
* **Merge / conflictos:**
  + Gana el registro con **lastModified mayor**. Tras aplicar, sincronizadoFirebase = true si local y remoto coinciden.
* **Usuarios FREE vs PREMIUM:**
  + Misma política offline-first para lectura/escritura.
  + **Única** limitación para FREE: **no** puede recuperar datos cuando **Room está vacío** (eso lo controla ya AuthRepository y un mensaje en PantallaMercadillos cuando detectemos ese caso).

**2.2 Estados y recálculo (constante)**

* **Estados confirmados** y transiciones automáticas:
  + **En curso (3):** desde **00:00** del día fecha hasta **05:00** del día siguiente (tz del dispositivo). **Ignoramos totalmente** horaInicio/horaFin para lógica de estado.
  + Al salir de “en curso”:
    - Si **0 líneas de venta** → **Cancelado (7)** automáticamente.
    - Si **tiene ventas** y no hay arqueo → **Pendiente de Arqueo (4)**.
    - Tras arqueo sin asignar saldo → **Pendiente de Asignar Saldo (5)**.
    - Tras asignar saldo → **Cerrado (6)**.
* **Recalc “constante”:** mientras PantallaMercadillos está visible:
  + Recalcular **cada minuto**.
  + Recalcular **onChange** de: fecha del sistema (incl. transición 00:00 y 05:00), conectividad, **cambios en tablas** (Flow de Room), **vuelta del background** y **cambios en el campo estado**.
  + En background: solo triggers de conectividad/fecha y al volver a la pantalla.

**2.3 Reglas Premium/Free**

* **Múltiples mercadillos en un mismo día:** **solo Premium** (FREE máx. 1/día).
* **Asignación automática de saldo (Premium):**
  + Cuando un mercadillo se cierra con saldoFinal definido:
    - Si existe el **siguiente mercadillo por fecha** de ese usuario en estado **1/2/3**, **preguntar** si asignar saldoFinal como saldoInicial.
    - Si ya tiene saldoInicial, **reemplazar con confirmación**.
    - **Sin deshacer** (se puede editar después desde alta/edición en estados 1/2/3).
* **Mensaje FREE con Room vacío y datos en nube:**
  + Debajo del calendario: *“Tienes datos guardados en la nube. Solo los usuarios Premium pueden recuperarlos en este dispositivo.”* + botón *“Hacerme Premium”*.  
    *(Claves reales las meteremos al final en StringResourceManager.)*

**2.4 Edición / Alta (solo reglas que aplican al Hito 1)**

* En **En curso (3)**:
  + **No** se puede cambiar **fecha**.
  + **Sí** se puede cambiar **saldoInicial**, **mostrando aviso** (“modificar el saldo en curso puede descuadrar caja. ¿Continuar? Sí/No”).
* **No** crear mercadillos en **fechas pasadas**. **Sí** crear “hoy” aunque horaFin orientativa haya pasado.
* horaInicio/horaFin formato **HH:mm** pero **no** validamos inicio < fin (puede acabar al día siguiente).
* En **4/5/6**: no se puede cambiar **nada**, y **no** se puede tocar el flag de cancelado.
* **Recordatorio**: al cerrar PantallaAltaMercadillos se **recalculan estados** y se **repinta** PantallaMercadillos.

**3) Integración con PantallaMercadillos (sin romper nada)**

* **Refresco al entrar/volver**: siempre lanzar **recalc + recarga de listas**.  
  Esto resuelve tu caso: cambiar fecha a “hoy” y que **aparezca** en “en curso” sin retrasos.
* **Bottom bar** (Ventas/Gastos/Resumen):
  + Mostrar **solo** si hay **≥1 en curso**.
  + Si hay **varios**, primera vez **abrimos diálogo** de selección; **recordamos** la elección **durante la sesión** (no persistente).
  + Opción **“Cambiar Mercadillo Seleccionado”** en **TopBar** (visible solo si hay más de uno en curso).
* **Lista “Próximos mercadillos”**: lista **completa**, **scroll**, **orden ascendente por fecha**.
* **Modal de selección cuando hay varios el mismo día**: mostrar **Lugar** y **Organizador** (orden por hora/creación si hace falta).

**4) Eventos/Disparadores**

* **Pantalla visible** → iniciar *ticker* 1 min (cancelable al salir) + suscripción a flows de Room + detectar 00:00/05:00 + conectividad.
* **Al guardar/editar/cerrar** mercadillo → disparar recalcEstados() y reloadQueries().
* **Pull-to-refresh**:
  + **FREE**: sin restricción adicional (misma sync normal), **no** hay descarga masiva (eso ya lo controla AuthRepository).
  + **PREMIUM**: *push* pendientes y *pull selectivo + merge*.  
    *(Esto es la sync normal de Mercadillos, no el bootstrap masivo).*

**5) Entradas/Salidas por componente (sin código)**

**5.1 MercadillosManager (negocio)**

* **Entradas**:
  + Flujos de Room (mercadillos, recibos, líneas, etc.).
  + Eventos del ciclo de vida (visible/oculto), temporizador 1 min, cambios de conectividad, reloj del sistema, acciones del usuario (alta/edición/cerrar/seleccionar en curso).
* **Salidas**:
  + Estados derivados: mercadillosEnCurso, mercadillosProximos, mercadillosPorDia, tieneMercadillos, etc.
  + Acciones: recalcEstados(), onMercadilloCerrado(), onAsignacionSaldoConfirmada(), recordarSeleccionEnCurso(id) y olvidarSeleccionEnCurso().
  + **UI Hints**: mostrar placeholder “Arqueo en desarrollo” (título/mensaje confirmados).
  + Mensajes FREE cuando Room vacío + nube con datos.

**5.2 MercadilloRepository (datos/sync)**

* **Entradas**:
  + userId actual (o usuario\_default comportándose como Free).
  + Señal *pull-to-refresh* y triggers de conectividad.
* **Salidas**:
  + Flujos de listas (Room) + sync controlada (push de pendientes y pull selectivo) con **merge por lastModified**.
  + Métricas de sync para logs (inicio/fin/errores).

**6) Requisitos de logging (para poder depurar fácil)**

* **Tag** fijo: MercadillosManager / MercadilloRepository.
* **Nivel DEBUG/INFO**:
  + Entradas a recalcEstados() con contadores por estado antes/después.
  + Decisiones automáticas (e.g., “Día cambió → 3 → 4” o “0 líneas → 7 Cancelado”).
  + Selección de mercadillo activo (guardar/olvidar).
  + Inicio/fin de pull-to-refresh (FREE/PREMIUM), contadores de *push/pull/merge*.
* **Nivel WARN/ERROR**:
  + Cualquier fallo de Room/Firebase, o incoherencias detectadas (e.g., fecha inválida).

**7) Criterios de aceptación (tests manuales)**

1. **Entra en PantallaMercadillos** con un mercadillo hoy → aparece **En curso (3)** y se muestra **bottombar**.
2. Cambia la fecha de un mercadillo a “hoy” y cierra la edición → al volver, la pantalla **muestra en curso** sin necesidad de hacer nada más.
3. Pasa la franja (simular tiempo o forzar recalc) con **0 líneas** → el mercadillo pasa a **Cancelado (7)** automáticamente.
4. Pasa la franja con **ventas** y sin arqueo → **Pendiente de Arqueo (4)**.
5. **Pull-to-refresh** con pendientes locales → se hace **push** y luego **pull** → **merge**; no hay reemplazos totales.
6. **FREE** con **Room vacío** y datos en nube → **mensaje** bajo calendario + botón “Hacerme Premium”; **no descarga**.
7. **Premium** cierra mercadillo con **saldoFinal** → si existe siguiente, **pregunta** asignar saldo; si el siguiente tiene saldoInicial, **pregunta** reemplazar.
8. **“Cambiar Mercadillo Seleccionado”** aparece en **TopBar** solo si hay >1 en curso; al pulsarlo, olvida la selección y vuelve a preguntar al navegar.
9. Lista “Próximos mercadillos” muestra **todos**, **orden ascendente**.
10. Modal con varios en un día muestra **Lugar y Organizador**.

**Hito 2 · Motor de estados + triggers**

**Objetivo**  
Calcular y mantener el estado correcto de cada mercadillo en todo momento, aplicando reglas y transiciones automáticas.

**Alcance clave**

* Estados y reglas confirmadas (1..7).
* Ventana “En curso (3)” = [00:00 día fecha, 05:00 día+1] en tz del dispositivo.
* Fin de ventana sin ventas → 7 (Cancelado). Con ventas → 4 (Pendiente de Arqueo).
* 5 (Pendiente de Asignar Saldo) tras arqueo; 6 (Cerrado) tras asignar saldo.
* Recalcular “constantemente” y persistir cuando cambie el valor.

**Plan (paso a paso)**

1. **Función pura** computeEstado(mercadillo, ventasCount, now) → Int
   * Ignorar horaInicio/horaFin.
   * Apoyarse en: fecha, ventana “en curso”, ventasCount, flags (pendienteArqueo, pendienteAsignarSaldo), saldoFinal.
   * Documentar tabla de verdad para cada transición.
2. **Servicio de recálculo** EstadoMercadillosManager
   * Itera mercadillos visibles (o todos del usuario actual).
   * Lee ventasCount (fast path: cache/consulta agregada).
   * Calcula nuevo estado; si cambia, updateEstado(id, nuevoEstado, lastModified=now).
   * Emitir log etiquetado por transición.
3. **Triggers**
   * **Cada minuto** mientras PantallaMercadillos está en foreground.
   * **OnChange** de tablas (Flow Room): mercadillos, recibos, líneas.
   * **Al entrar a la pantalla**, **al volver del background**, **cambio de usuario**.
   * **Timers de frontera**: disparo a las 00:00 y 05:00 locales.
4. **Persistencia mínima**
   * Solo persistir cuando nuevoEstado != estado.
   * Actualizar lastModified y marcar sincronizadoFirebase=false.
5. **Logs**
   * from→to, razón (regla), ids, fecha/hora.
6. **Performance**
   * Batch/limits (p.ej. recálculo en lotes de 200) si el dataset crece.
7. **QA / Hecho si**
   * Matriz de pruebas (días pasados/hoy/futuros, con/sin ventas, cambios a las 00:00/05:00).
   * Observas cambios en UI sin acciones del usuario.
   * Sin cambios espurios de estado.

**Hito 3 · Sync normal offline-first (Mercadillos)**

**Objetivo**  
Sincronización selectiva de mercadillos con Firebase: **push pendientes** y después **pull + merge** (sin reemplazos totales).

**Alcance clave**

* Independiente de AuthRepository (que ya controla bootstrap solo si Room vacío).
* Free y Premium idénticos aquí (según tu aclaración).
* Regla de conflicto: **gana mayor lastModified**.

**Plan**

1. **Detección de pendientes**
   * Query Room por sincronizadoFirebase=false del usuario.
2. **Push**
   * Subir pendientes (upsert) a Firebase.
   * Al confirmar, marcar sincronizadoFirebase=true, version++ y lastModified servidor si aplica.
3. **Pull selectivo**
   * Estrategia simple para empezar: **pull completo de mercadillos del usuario** (filtrado por userId).
   * (Optimizable luego por lastModified > maxLocal si lo quieres).
4. **Merge**
   * Para cada remoto R: buscar local L por id.
   * Si no existe L → insertar R (sincronizado=true).
   * Si existe: comparar lastModified. Aplica el más reciente; marca sincronizadoFirebase acorde.
5. **Ejecución**
   * Función syncMercadillos() que haga pushPendientes() → pullAndMerge().
   * Exponer SyncState y mensajes breves (opcional overlay H9).
6. **Logs**
   * Conteo: subidos, insertados, actualizados, ignorados.
7. **QA / Hecho si**
   * Pendientes se vacían con conexión.
   * Sin pérdidas de datos locales (merge correcto).
   * Repetible sin efectos adversos.

**Hito 4 · Selección de mercadillo “en curso” (Bottom bar + TopBar acción)**

**Objetivo**  
Cuando haya ≥1 mercadillo en **estado 3**, mostrar BottomBar con acciones; si hay varios, pedir selección **una vez por sesión** y permitir **cambiar** desde el TopBar.

**Plan**

1. **Fuente de verdad**
   * Selector de “en curso”: query Room por estado=3 y usuario actual.
2. **Memoria de sesión**
   * Guardar idMercadilloSeleccionado en ViewModel/remember (no persistente).
3. **Primer uso**
   * Si hay 1 → seleccionar automáticamente.
   * Si >1 → abrir diálogo de selección; guardar elección.
4. **TopBar acción**
   * Mostrar “**Cambiar Mercadillo Seleccionado**” solo si hay >1 en curso.
   * Limpia selección y reabre diálogo.
5. **Invalidación**
   * Si el seleccionado sale de estado 3, olvidar selección.
6. **QA / Hecho si**
   * Flujo natural sin repetir diálogos.
   * Acción TopBar visible/oculta correctamente.

**Hito 5 · Próximos mercadillos + modal por día**

**Objetivo**  
Lista de **todos** los próximos mercadillos (scroll, **ascendente por fecha**). Al pulsar un día con varios, modal con **lugar y organizador** para elegir.

**Plan**

1. **Query**
   * Próximos: fecha >= hoy ordenados por fecha asc.
2. **UI**
   * Card/List completa (sin “+N más”).
3. **Calendario**
   * Click en día: si 0 → nada; si 1 → navegar edición; si >1 → modal.
4. **Modal contenido**
   * Por item: lugar + organizador (y opcional icono mínimo).
   * Orden ascendente por hora o por hora textual si quieres.
5. **Navegación**
   * Crear: alta\_mercadillo?fecha=yyyy-MM-dd.
   * Editar: editar\_mercadillo/{id}.
6. **QA / Hecho si**
   * Lista actualizada tras cambios; modal muestra y navega correctamente.

**Hito 6 · Validaciones alta/edición + límites Free/Premium**

**Objetivo**  
Reglas de creación/edición, restricciones por estado y límites por plan.

**Plan**

1. **Creación**
   * No permitir **fechas pasadas**.
   * **Free**: máximo **1 mercadillo por día** (misma fecha).
   * **Premium**: sin límite.
2. **Edición según estado**
   * **1/2**: editable completo.
   * **3**: no cambiar **fecha**; **saldoInicial** editable con **aviso** (“puede causar descuadre”).
   * **4/5/6**: **bloquear todo** (incluido flag cancelado). Botones: Cancelar / Arqueo (placeholder).
3. **Campos de importe**
   * esGratis=false exige importeSuscripcion informado (>=0 permitido; no vacío).
   * Moneda desde **ConfigurationManager**; 2 decimales de visualización.
4. **Hora**
   * Aceptar horaFin < horaInicio (puede pasar a madrugada).
5. **Mensajes**
   * Usar claves provisionales (se rellenarán al final).
6. **QA / Hecho si**
   * No se puede crear en pasado.
   * Free bloquea 2º mercadillo en el mismo día.
   * Ediciones respetan reglas por estado.

**Hito 7 · Asignación automática de saldo (Premium) con confirmación**

**Objetivo**  
Al **cerrar** mercadillo con saldoFinal, proponer asignarlo como saldoInicial del **siguiente** mercadillo (por fecha) del usuario.

**Plan**

1. **Disparo**
   * Transición a 6 (Cerrado) con saldoFinal != null en Premium.
2. **Búsqueda**
   * Siguiente mercadillo por fecha > fechaActual, en estado 1/2/3 (programado o en curso).
3. **Confirmaciones**
   * Pregunta “¿Asignar saldoFinal como saldoInicial del siguiente mercadillo?”
   * Si el siguiente ya tiene saldoInicial → confirmación de **reemplazo**.
4. **Aplicación**
   * Actualizar saldoInicial del siguiente; marcar sincronizadoFirebase=false, lastModified=now.
5. **No hay “deshacer”**
   * Editable manualmente en estados 1/2/3.
6. **QA / Hecho si**
   * Solo Premium.
   * Doble confirmación (si corresponde).
   * Datos quedan coherentes en UI y Room.

**Hito 8 · Banner Free cuando Room vacío y hay datos en nube**

**Objetivo**  
Mostrar un **banner** bajo el calendario si: **Room está vacío**, **usuario Free** y **existen datos en Firebase**.

**Plan**

1. **Señal**
   * Reutilizar flag de AuthRepository o calcular al entrar: RoomEmpty && remoteHasData.
2. **UI**
   * Texto: “Tienes datos guardados en la nube. Solo los usuarios Premium pueden recuperarlos en este dispositivo.”
   * Botón: “Hacerme Premium”.
3. **Navegación**
   * Botón lleva a pantalla de suscripción/Configuración.
4. **QA / Hecho si**
   * El banner aparece/desaparece en el momento correcto.
   * No bloquea interacción normal.

**Hito 9 · Progreso de Sync en PantallaMercadillos**

**Objetivo**  
Indicar visualmente el proceso de sincronización (Checking/Uploading/Downloading/Merging) sin interferir con la navegación.

**Plan**

1. **Estados UI**
   * Consumir syncState + (opcional) downloadProgress/message cuando aplique.
2. **Overlay o Snackbar**
   * **Overlay ligero** cuando el proceso sea corto y crítico (bloqueo parcial de toques, card con spinner y mensaje).
   * **Snackbar** para avisos rápidos.
   * Tu criterio: activar overlay para fases > 500–800 ms.
3. **Ubicación**
   * En **PantallaMercadillos**, no en login.
4. **QA / Hecho si**
   * Mensajes claros; no queda “pegado” tras finalizar.
   * No rompe gestos de scroll/calendario.

**Hito 10 · Cierre y hardening**

**Objetivo**  
Dejar todo robusto y listo para crecer sin romper lo existente.

**Plan**

1. **Logs y telemetría**
   * Etiquetas por módulo (Estados, Sync, Selección, Alta/Edición, Asignación).
   * Logs de error con contexto (userId, idMercadillo, transición).
2. **Protecciones**
   * Anchors: no permitir ediciones en estados bloqueados (doble verificación).
   * Validaciones de parámetros de rutas.
   * “Circuit breakers” si Firebase falla (función local sigue).
3. **QA / Matriz**
   * Casos por cada estado y transición.
   * Free vs Premium (límite por día, banner, asignación).
   * Offline → online (sync push/pull).
   * Cambio de tz / 00:00 / 05:00.
4. **Strings**
   * Recopilar claves usadas en todo el flujo. Entregarte listado final para StringResourceManager.
5. **Entregables**
   * Documento de “puntos de extensión” (para futuras tablas/features).
   * Checklist de regresión.
   * Resumen de decisiones y reglas (este mismo plan consolidado).