Perfecto: lo que quieres es un **híbrido** "tipo Vindax" pero on-chain:

- **BUY (compra)**: tu "swap local" (frontend) llama a **PancakeSwap v3** para comprar TCT con USDT.
- **SELL (venta)**: tu "swap local" NO llama a Pancake. Llama a **un contrato tuyo** que hace "recompra" (buyback) y paga USDT desde **tu tesorería** con **límites**.

Eso te da control operativo (como Vindax): el pool no se drena por ventas, y tú controlas cuánto USDT sale y a qué precio (con guardas para no romperte).

A continuación te dejo el **flujo completo** y un **smart contract listo** para Remix/Hardhat.

Flujo de trabajo completo

A) Preparación (una sola vez)

1. **Pool en Pancake v3**: ya lo tienes (TCT/USDT).
2. **Frontend "Swap Local"**:
   - Botón **Comprar** → llama a Pancake v3 Router (swap USDT→TCT).
   - Botón **Vender** → llama a tu contrato `LocalSellDesk.sellTCT(...)`.
3. **Contrato de ventas (recompra)**:
   - Tú depositas USDT al contrato (tesorería).
   - El contrato compra TCT a usuarios con límites (cap diario, cap por tx), y con un **precio de referencia**.

B) Operación diaria

- La gente compra TCT por tu UI (swap real en Pancake).
- La gente deposita/usa TCT en tu plataforma.
- Si alguien quiere vender:
  - tu UI lo manda al contrato de recompra
  - el contrato paga USDT **solo hasta el cupo** disponible
  - si se acaba el cupo o no hay USDT, la venta se rechaza (o pausas el sell).

✅ Resultado: Pancake queda como "precio público / compra", y tu venta queda controlada como "mesa de salida".

Smart contract: "Sell Desk" (recompra controlada)

Qué hace:

- Paga USDT a cambio de TCT (venta del usuario).
- Precio lo defines tú (manual), pero:
  - el contrato verifica que ese precio no esté demasiado lejos del precio spot del pool v3 (anti-abuso / anti-arbitraje).
- Tiene:
  - `dailyCapUSDT` (cupo diario)
  - `maxPerTxUSDT` (límite por transacción)

- `pauseSell`
- `fundUSDT`

Nota: Para simplificar y que sea deployable ya, uso **precio spot del pool (slot0)** como verificación. Si luego quieres TWAP real (promedio), se agrega con `observe()` del pool.