# **DEPRECATED — superseded by “Serapods — Functional Build Pack v2 (APIs, Data, Workflows)” and “Serapod UI Blueprint v2 (Zero‑Data Prototype)”.**

Keep this file for historical reference only. Use the two v2 documents for any new development.

# Serapods — Functional Build Pack v2 (APIs, Data, Workflows)

**Authoritative build guide** for engineers and AI code assistants. Pairs with **Serapod UI Blueprint v2 (Zero‑Data Prototype)** and **Serapod QR Code for Cursor V2 (MD)**. Clean rewrite for new development; all behavior reflects our latest decisions (derived verify behavior, 2‑level approvals, payment gating, QR settings per Product Group, WhatsApp controls in HQ Settings only).

## 1) Platform & Architecture

* **Clients**
  + Next.js Web Admin + Desktop PWA scanner (camera/HID; IndexedDB offline queue)
  + React Native apps (Android/iOS) for scanning (SQLite offline queue)
* **Core**
  + Supabase Auth (email/password for ops; Google/Facebook for consumers) with **user\_group\_roles**
  + Postgres (system of record) + **RLS** for Product Group isolation
  + Storage (images, PDFs, CSV/XLSX, SVG)
  + Edge Functions (Deno/TS) for secure writes, validation, QR/PIN/labels, notifications
  + Realtime/pg\_cron for dashboards & jobs
* **Derived verify behavior**
  + workflow\_profile='vape' ⇒ **verify ends at Shop/Kedai** (consumer scans → health warning page)
  + workflow\_profile='nonvape' ⇒ **consumer path allowed** only when campaign‑eligible

## 2) Data Model (scope for MVP)

All tables carry: id uuid pk, product\_group\_id uuid fk, timestamps, status text, sid uuid (delete guard).

* **product\_groups**: code, name, workflow\_profile in ('vape','nonvape'), verify\_behavior (**derived, stored**)
* **user\_group\_roles**: (user\_id, product\_group\_id, role)
* **categories**: code, name, requires\_regulatory
* **brands**: code, name, category\_id
* **manufacturers**: code, name, contacts, address
* **products**: code, name, category\_id, brand\_id, manufacturer\_id? (exclusive when set), images, tangible attrs (flavour, price\_cents MYR, country\_of\_origin, weight\_g), **Option‑B** fields (enforced for Vape/regulated)
* **qr\_settings**: per Product Group (units\_per\_master, buffer\_per\_1000)
* **orders**: code, manufacturer\_id, status in ('pending\_approval','approved','po\_sent','payment\_notified','payment\_verified'), approver stamps, po\_code, po\_sent\_at, payment\_notified\_at
* **order\_approvals**: (order\_id, approval\_type in ('hq\_l1','hq\_l2','mfr\_payment\_verified'))
* **batches**: batch\_no (8d), order\_id, product\_id, total\_units, masters\_count, master\_id\_mode in ('qr\_only','rfid\_only','both'), label badge fields, production\_date, mfg\_facility\_code
* **masters**: seq\_no, code (ULID), epc\_uri, epc\_hex, quantity
* **units**: master\_id, code (ULID), seq\_in\_master, pin\_hash, pin\_salt
* **scan\_events**: ingest log with idempotency\_key, role, action, device info, result
* **purchase\_order\_documents**, **invoices**, **invoice\_lines** (commercial docs)
* **notification\_settings/templates/logs**
* **label\_badge\_catalog**, **qr\_exports**, **assignments** + **assignment\_items**, **points\_ledger**, **redemptions**

**RLS**: scope by set\_config('app.pg\_id', :pg\_id, true); policies require matching product\_group\_id and role in user\_group\_roles. **DELETE** denied on master data.

## 3) Workflows

### 3.1 HQ → Manufacturer (Order & Payment)

1. **Create Order** (status pending\_approval)
2. **Approve L1** (HQ A)
3. **Approve L2** (HQ B) ⇒ status approved
4. **Generate & Send PO** ⇒ status po\_sent
5. **Notify Payment** (HQ) ⇒ status payment\_notified
6. **Ack Payment** (Manufacturer) ⇒ records mfr\_payment\_verified ⇒ status payment\_verified
7. **Manufacturer generates Batch** (after payment verified)

### 3.2 Manufacturer → Warehouse → Distributor → Shop

* **Produced** (scan Unit/Master) → **In‑Warehouse** (dock‑door RFID or Master scan) → **Assign to Distributor** (multi‑scan Masters) → **Delivered to Shop**. Consumers participate **only in NonVape** when campaign‑eligible.

### 3.3 Distributor → HQ (B2B Orders)

* Distributor places order for available/coming batches → HQ approves/rejects → Distributor sends **D‑PO** → HQ sends **Invoice to Distributor** → Distributor **ack payment** → Complete. Food‑delivery style tracker in UI.

## 4) APIs (v1) — Exact Payloads

**Conventions**: Base /api/v1/…, header Authorization: Bearer <JWT>, header X-PG-ID: <uuid>. Success { ok:true, data, request\_id }; Error { ok:false, error:{ code,message,field\_errors?,hint? }, request\_id }.

### 4.1 Groups & MDM

* POST /groups → { code?, name, workflow\_profile } → server derives verify\_behavior
* POST /categories → { pg\_id, name, requires\_regulatory? }
* POST /brands → { pg\_id, category\_id, name }
* POST /products → tangible attrs + Option‑B; optional manufacturer\_id; **image required** (upload token)
* PATCH /products/:id/status → { pg\_id, status:'active'|'archived'|'blocked' }
* GET /catalog/products?pg\_id=&q=&brand\_id=&category\_id= → returns only status='active'

### 4.2 Orders (HQ→Manufacturer)

* POST /orders → { pg\_id, manufacturer\_id, lines:[{product\_id, qty\_units, unit\_price\_cents}] }
* POST /orders/approve-level-1 → { order\_id, user\_id }
* POST /orders/approve-level-2 → { order\_id, user\_id }
* POST /orders/:id/po/render
* POST /orders/:id/po/send
* POST /orders/:id/notify-payment
* POST /orders/:id/ack-payment *(optional)*

### 4.3 Batches (Manufacturer)

* POST /batches

{  
 "pg\_id":"uuid",  
 "order\_id":"uuid",  
 "product\_id":"uuid",  
 "total\_units":10000,  
 "production\_date":"2025-01-01",  
 "mfg\_facility\_code":"MY",  
 "master\_id\_mode":"both",  
 "label\_badge\_id":null,  
 "label\_badge\_text\_override":"FREE DEVICE",  
 "label\_badge\_apply\_to":"unit"  
}

**Logic**: load qr\_settings → BufferUnits=floor(total\_units/1000)\*buffer\_per\_1000; TotalUnique=total\_units+BufferUnits; Masters=ceil(TotalUnique/units\_per\_master); generate ULIDs; persist; render CSV/XLSX + PDF/SVG; return signed URLs.

### 4.4 Scans & Inventory

* POST /scan → { pg\_id, code\_type:'unit'|'master'|'rfid', code, actor\_role, action, device\_id, device\_seq, client\_ts } (idempotent)
* POST /assign → { pg\_id, distributor\_id, master\_ids:[...] }
* POST /move → { pg\_id, master\_id, bin\_id }

### 4.5 Distributor→HQ Orders

* POST /distributor-orders → { pg\_id, batch\_id?, product\_id?, qty\_units, notes? }
* POST /distributor-orders/:id/approve|reject *(HQ)*
* POST /distributor-orders/:id/send-dpo *(Distributor)*
* POST /distributor-orders/:id/send-invoice *(HQ)*
* POST /distributor-orders/:id/ack-payment *(Distributor)*
* GET /distributor-orders/:id/tracking

### 4.6 Notifications (HQ Settings only)

* PUT /settings/notifications → { pg\_id, whatsapp\_enabled, daily\_cap, events:{...} }
* PUT /settings/notification-templates → { pg\_id, templates:{ shop\_new\_batch\_coming:"New batch is coming. Order & contact your distributor now. {{free\_gift\_note}}" } }
* POST /notify/test → channel: 'whatsapp'|'email'

## 5) Validation & Errors

**Global**: require X-PG-ID; JWT must map to a role in user\_group\_roles. Enforce enums and ranges; Option‑B for Vape/regulated categories; image required on product create.

**Standard error codes**: VALIDATION\_ERROR, UNAUTHORIZED, FORBIDDEN, NOT\_FOUND, CONFLICT, PRECONDITION\_FAILED, UNPROCESSABLE\_ENTITY, RATE\_LIMITED, IDEMPOTENT\_REPLAY, INTERNAL\_ERROR.

## 6) Labels, QR & RFID

* QR payload: opaque ULID with short link https://q.serapod2u.com/u/<id>; human line SP<BRAND3><CAT3><YYYYMMDD><BATCH6><FAC2><PIN4>
* RFID: RAIN UHF, GS1 **SSCC‑96** EPC on masters (URI + HEX stored); QR fallback always available
* Label badges: catalog or custom text; apply to Unit/Master/Both; printed on PDFs and exported in CSV/XLSX

## 7) Verify (Public)

* Vape: authenticity + Shop flows; consumer scans → health warning page (Storage HTML)
* NonVape: consumer panel appears only when campaign‑eligible; Google/Facebook login; redeem/points

## 8) API Versioning

* Base path /api/v1; responses include X-API-Version: v1
* Breaking changes → /api/v2 with ≥6‑month deprecation window for v1
* Non‑breaking additive fields remain in v1

## 9) Acceptance Criteria (MVP)

* Groups: single choice (profile) → verify derived; persisted
* 4‑tier MDM with images and tangible attrs; delete protection via RLS + sid
* Orders: L1/L2 approvals; PO render/send; payment notify; manufacturer ack
* Batches: config‑driven buffer/split; exports created; product lock after first batch
* Scans: offline replay; idempotent; legal transitions enforced
* Catalog: Distributor/Shop see **only active** products
* Notifications: WhatsApp controlled only in HQ Settings; caps; logs

## 10) Diagrams (PlantUML)

**Legend**: Actors — HQ (HQ Admin), Man (Manufacturer), Wh (Warehouse); Participants — Web (Admin/PWA), Mobile (RN), OB (Order & Batch), QR (QR & Label), SI (Scan Ingest), RI (RFID Ingest), Inv (Inventory), PG (Postgres)

**Order & Payment Approval**

@startuml  
actor HQ as "HQ Admin"  
actor Man as "Manufacturer"  
participant Web as "Web Admin"  
participant OB as "Order & Batch"  
database PG  
HQ -> Web: Create Order  
Web -> OB: POST /orders  
OB -> PG: insert (pending\_approval)  
HQ -> Web: Approve L1  
Web -> OB: POST /orders/approve-level-1  
OB -> PG: record (hq\_l1)  
HQ -> Web: Approve L2  
Web -> OB: POST /orders/approve-level-2  
OB -> PG: record (hq\_l2); status=approved  
HQ -> Web: Send PO  
Web -> OB: POST /orders/:id/send-po  
OB -> PG: set po\_code, po\_sent\_at; status=po\_sent  
HQ -> Web: Notify Payment  
Web -> OB: POST /orders/:id/notify-payment  
OB -> PG: payment\_notified\_at; status=payment\_notified  
Man -> Web: Ack Payment (optional)  
Web -> OB: POST /orders/:id/ack-payment  
OB -> PG: record (mfr\_payment\_verified); status=payment\_verified  
@enduml

**QR Batch Generation**

@startuml  
actor Man as "Manufacturer"  
participant Web  
participant OB as "Order & Batch"  
participant QR as "QR & Label"  
database PG  
Man -> Web: Generate Batch (after payment verified)  
Web -> OB: POST /batches  
OB -> PG: load qr\_settings (UPM, B1000)  
OB -> PG: insert batch (masters\_count)  
OB -> QR: generate masters/units (ULIDs)  
QR -> PG: persist  
QR -> Web: CSV/PDF/SVG links  
@enduml