¡Qué buen momento para ordenar el rumbo! Te propongo un plan claro en 3 frentes (Industria 4.0, Gantt, Almacén) y, muy importante, la estrategia de **datos** (Firestore vs SQL) para que lo que construyamos hoy no te limite mañana.

**0) Decisiones de datos (rápido y sin dolor)**

**Qué usamos para cada cosa**

* **Firestore (NoSQL, en tiempo real)** para: app móvil (altas, edición de órdenes, tablero, operadores), colas de trabajo, estados, notificaciones. Es excelente para sincronizar y funciona offline.
* **SQL/BigQuery (analítica & reportes grandes)** para: reportes financieros, históricos largos, KPIs pesados, trazabilidad a meses/años, integración ERP.

**Cómo lo unimos**

* **Dual-write por eventos**: mantenemos Firestore como *source of truth* operacional y **replicamos** eventos relevantes a SQL/BigQuery.
  + Triggers (Cloud Functions) en colecciones clave (production\_daily, movements, cnc\_events, etc.) → escriben en **Cloud SQL (PostgreSQL)** o **BigQuery**.
  + Con esto puedes crecer sin reescribir la app.

Resultado: hoy avanzas rápido; mañana haces BI robusto sin rehacer todo.

**1) Drawer “Industria 4.0” (CNC + OEE)**

**Flujo propuesto (muy estándar y fácil de escalar)**

1. **Capa de campo (shopfloor)**
   * Micro-servicio/raspberry en cada máquina (o una por celda) que se conecte por **OPC-UA** o directamente al controlador y publique a **MQTT**.
   * Publica tópicos como: cnc/{machineId}/state, .../cycleStart, .../cycleEnd, .../alarm, .../partCount.
2. **Ingesta en la nube**
   * MQTT → **Cloud Pub/Sub** → **Cloud Functions** → **Firestore**:
     + Colección cnc\_events: (machineId, ts, type, payload…)
     + Colección cnc\_states: documento por máquina con último estado (running/idle/alarm), contadores, última orden vinculada, etc.
3. **Vinculación con órdenes**
   * Cuando un supervisor asigna una orden a una máquina (en production\_daily.maquinaRef), guardamos también production\_daily.machineId.
   * Cloud Function escucha cnc\_events y, si la máquina tiene orden activa, **incrementa pass/fail** o **marca inicio/fin** automáticamente (según señal).
4. **OEE**
   * Mantén una colección oee\_daily/{machineId}\_{yyyyMMdd} con acumulados (runtime, downtime, good, scrap) calculados por funciones.

**UI (pantalla industry\_screen.dart)**

* Tarjetas por máquina (estado en vivo, orden activa, ciclo, tasas).
* Vista de **timeline** de estados del día (verde running, amarillo idle, rojo alarm).
* Botón “vincular a orden” si no hay orden activa.

**Colecciones nuevas mínimas**

machines/{machineId} { nombre, bodegaId, ... }

cnc\_states/{machineId} { status, lastHeartbeat, cycleTimeSec, partCount, activeOrderId? }

cnc\_events/{autoId} { machineId, ts, type, payload }

oee\_daily/{machineId\_yyyyMMdd} { runSec, stopSec, good, scrap }

**2) Gantt “entendible” por proyecto**

**Qué quieres ver**

* Un **Gantt por proyecto**, desglosado en **part numbers** y **operaciones** (ruta de proceso).
* Cada **barra** = una **operación** de una orden (cantidad), con plan vs real (fechas).

**Modelo de datos que lo hace posible**

* **Ruta por parte** (plano maestro):

projects/{projectId}

parts/{partId}

route/{stepId} {

secuencia: 10, // orden

operacionRef,

duracionStdMin: 15, // estándar x pieza

requiereMaquina: true,

maquinaTipo?: 'CNC', // opcional

obligatorio: true

}

* **Orden de producción** (ya usamos production\_daily) → una línea por operación:

production\_daily/{id} {

proyectoRef, parteRef, operacionRef, cantidad,

fechaPlanInicio, fechaPlanFin, // plan

inicio, fin, // real

status, maquinaRef, operadorRef, ...

}

Cuando creas una orden “macro” de X piezas de un P/N, generas **N sub-órdenes** (una por operación en route). Eso hace el Gantt trivial: cada sub-orden es una barra.

**UI sugerida (limpia & clara)**

* Usa **syncfusion\_flutter\_gantt** (muy completo) o construye con timelines/fl\_chart (más trabajo).
* Estructura:
  + GanttScreen → filtra por proyecto
  + Izquierda: árbol (Parte → Operación 10, 20, 30…)
  + Derecha: barras **Plan** (gris) y **Real** (color por status)
  + Click en barra → abre orden (para editar o ver detalle)
* Ordenación por secuencia y luego por fecha.

**Tareas**

* Crear route por parte (CRUD simple para diseñador).
* Al “planear” producción de una parte (X piezas), generar las sub-órdenes production\_daily con operacionRef de cada step y sus fechaPlan\*.
* En Gantt, agrupar por parte y pintar por operación.

**3) Almacén “de verdad” (stocks + movimientos + reservas)**

**Estructura robusta (funciona en Firestore y replica perfecto a SQL)**

* **Catálogo de ítems** (materiales, semiterminados, terminados):

items/{itemId} { sku, tipo: 'MP|WIP|PT', descripcion, unidad }

* **Ubicaciones (bodegas / racks)**:

locations/{locId} { nombre, tipo: 'BODEGA|LINEA|CALIDAD' }

* **Stock por ítem/ubicación** (snapshot actual, rápido para leer):

stocks/{itemId\_locId} {

itemRef, locationRef, onHand: 120, reserved: 30, updatedAt

}

* **Movimientos (libro mayor, la verdad absoluta)**:

movements/{id} {

ts, tipo: 'IN|OUT|TRANSFER|ADJUST',

itemRef, qty, fromLocRef?, toLocRef?,

refType: 'purchase|production|scrap|manual',

refId, // id de documento origen (orden, compra…)

userRef

}

**Reglas de negocio clave**

* Nunca edites stocks directamente.  
  **Siempre** crea un movement y una **Cloud Function** actualiza stocks de forma transaccional.
* Para producción:
  + Al **iniciar** operación, opcionalmente **reserva** material (incrementa reserved).
  + Al **terminar**:
    - OUT de materia prima → reduce onHand.
    - IN de producto/semiproducto (a WIP o PT) → aumenta onHand en destino.
* Trazabilidad garantizada por movements.

**UI mínima**

* WarehouseScreen:
  + Pestañas: **Stock** (tabla con filtros por bodega), **Movimientos** (últimos), **Transferir** (formulario de mover ítems).
  + **Escáner** (más adelante): lectura de QR para itemId y locId.

**4) Qué hacemos ya (plan de 2–4 semanas)**

**Semana 1**

* **Admin de usuarios** (pantalla lista para pegar).
* En new\_project\_part\_screen.dart: **múltiples P/N** por lote (filas dinámicas).
* gantt\_screen.dart: consumir production\_daily y mostrar por proyecto → parte → operación (plan vs real).

**Semana 2**

* **Modelo de almacén**: items, locations, movements, stocks, reglas.
* Cloud Function onCreate(movements/\*) que actualice stocks atómicamente.
* UI WarehouseScreen: ver stock y registrar **IN/OUT/TRANSFER** manual.

**Semana 3**

* **Industria 4.0 (fase 1)**:
  + cnc\_states, cnc\_events, oee\_daily colecciones y pantallas base.
  + Mock de eventos (simulación) para probar flujo
  + Vincular máquina ↔ orden activa (botón en tarjeta máquina).

**Semana 4**

* **Replicación a SQL/BigQuery** (mínimo viable):
  + Triggers para production\_daily y movements.
  + Esquema relacional en Postgres (opciones: orders, order\_ops, movements, stock\_snapshots).
  + Reporte ejemplo en SQL (costo de material consumido por proyecto).

**5) Pequeñas tareas que cierran flecos**

* **Semana S0**: ya dejamos fallback en UI; si quieres backfill, te paso el script cuando me digas y lo corremos 1 sola vez.
* **Indices**: asegúrate de tener:
  + production\_daily: operadorUid ASC, fecha DESC
  + production\_daily: proyecto ASC, fecha DESC
  + users: role ASC, displayName ASC
* **Estados**: usa minúsculas en todo (programado/en\_proceso/pausado/hecho), ya lo normalizamos.

Si te late, arranco con:

1. **Pantalla Industry 4.0** (UI + colecciones + simulador local),
2. **Gantt** con jerarquía Proyecto → Parte → Operación y barras plan/real,
3. **Warehouse** con **movements** + función de stock.

Y en paralelo te dejo la versión **multifila** del alta de P/N en new\_project\_part\_screen.dart para que tu diseñador cargue rápido los BOM/rutas.

¿Listo? Te acompaño paso a paso para que esto sea la base del **sistema de manufactura** de FWS sin dolores futuros.

**0) Foto actual (muy breve)**

* **Colecciones**: projects/{id}/parts/{id}, operations, production\_daily, users, machines, warehouse\_\*, i40\_\*.
* **Campos clave** en production\_daily: proyecto, numeroParte, operacionNombre, opSecuencia (a veces falta), operadorUid, fecha, fechaCompromiso, inicio, fin, métricas y estatus.
* **Roles**: admin, supervisor, diseñador, operador.

**1) Priorización (qué hacemos y en qué orden)**

**P0 – Urgentes (desbloquean el flujo)**

1. **Gráfica (tab “Gráfica”)**
   * Agregar filtros por **operación** y **operador**.
   * Cambiar agregación para **no sumar todas las operaciones**; por defecto mostrar **piezas plan vs real por proyecto** (puedes filtrar *“solo MANUFACTURA”* o *“solo última operación del flujo”*).
2. **Dropdown de operadores en Alta (supervisor)**
   * Regresarlo (bug de normalización de role y/o query).
3. **opSecuencia**
   * Dejar de meterlo a mano: **levantarlo de operations** y **backfill** los docs que ya existen.

**P1 – Alto impacto, inmediata siguiente iteración**

1. **Alta masiva de flujo completo** en AddProductionEntryScreen: chips con las operaciones por defecto (ordenadas por secuencia) para crear **N actividades** en batch (poder activar/desactivar algunas).
2. **Gantt** (ya lo enderezamos): asegurarnos que toma opSecuencia siempre (si falta, resuelve contra operations en memoria), y que el orden es por parte→secuencia.

**P2 – Construcción de base futura**

1. **Carga de archivos del diseñador** (planos .jpg/.pdf y sólidos .x\_t/.step)
   * Suben a **Firebase Storage** y se guardan URLs y metadatos en projects/{}/parts/{}.
   * Mostrar esos archivos cuando el supervisor asigna y al operador al ejecutar.
2. **Módulo de requisiciones** simple: requisitions con flujo *Abierta→Aprobada→Entregada* (material/planos).

**P3 – Seguimiento y control**

1. **Scrap workflow**: al registrar scrap → scrapPendiente=true + crear doc en scrap\_requests. Vista para supervisor (aprove/aprueba) y badge rojo. (Opcional: Telegram/WhatsApp webhook o FCM).

**2) Cambios concretos (qué tocar y cómo)**

**A) Tab “Gráfica”: filtros + agregación correcta (P0)**

* **UI**: arriba de la gráfica añade dos DropdownButtonFormField:
  + **Operación**: opciones de operations (orden por sec), más “Todas”.
  + **Operador**: lista de users con role == 'operador', más “Todos”.
* **Query base**: sigue leyendo production\_daily, pero el **cálculo**:
  + Por **defecto** calcula **piezas plan/real por proyecto** contando **solo la operación “MANUFACTURA”** (o, si prefieres, la **última operación** por opSecuencia → así no duplicas).
  + Si usuario selecciona una operación específica: entonces **sí** suma por esa operación únicamente.
  + Si elige operador: filtra por operadorUid.
* **Campos** que usaremos:
  + opSecuencia o operacionNombre. Si falta opSecuencia, lo resuelves con un map en memoria Map<String,int> opsByName leído de operations.

**Índice sugerido** (composite) para esta pantalla:  
production\_daily(proyecto ASC, operacionNombre ASC, operadorUid ASC, fecha ASC)

**B) Alta (supervisor): dropdown de operadores (P0)**

* Revisa tu query: en tu código yo ya normalicé roles en minúsculas. Asegúrate de que **en Firestore** el role de los operadores esté exactamente 'operador' (sin acentos/espacios).
* Cambia query a:
* FirebaseFirestore.instance
* .collection('users')
* .where('role', isEqualTo: 'operador')
* .orderBy('displayName')
* Si quieres robustez: **normaliza** el rol al guardar (lowercase, sin tildes).

**C) opSecuencia automático + backfill (P0)**

* **Al guardar** en AddProductionEntryScreen:
  1. Cuando el usuario elige operacionId/operacionNombre, consulta operations/{id} y toma sec.
  2. Guarda opSecuencia: sec.
* **Backfill** (única vez): pequeño script dentro de la app (solo admin):
  1. Lee operations → arma Map<String,int> secByName.
  2. Paginado de production\_daily, si opSecuencia es null, setea con el map por operacionNombre.
  3. Sólo visible para admin en un botón “Backfill opSecuencia”.

**Índice**: production\_daily(operacionNombre ASC, opSecuencia ASC) (opcional, ayuda para Gantt/Gráfica).

**D) Alta masiva de flujo (P1)**

* En AddProductionEntryScreen agrega toggle **“Crear flujo completo”**.
* Carga operations donde default==true, ordenadas por sec. Muéstralas como **chips seleccionables** (pre-seleccionadas).
* Al guardar: **batch** con un doc por operación seleccionada, todos con la misma parte, fechas plan y el opSecuencia correcto.
* Validación: si selecciona **0** operaciones → error amigable.

**UX tip**: conserva el operador, máquina y fechas como “valores por defecto” del bloque y permite override por operación si el usuario abre un “detalle”.

**E) Gantt (P1)**

* Ya quedó el scroll y la línea de “Hoy”. Refuerza el orden de operaciones:
  + Si opSecuencia == null, resuélvelo con secByName[operacionNombre] ?? 9999.
* Para **orden de proyectos** en “escalera”: usar la **primera fecha** (plan o real) encontrada. Ya lo pusimos así.
* (Opcional) Botón “**Solo MANUFACTURA**” para resumir.

**F) Carga de planos y sólidos (P2)**

* **Paquetes**: file\_picker + firebase\_storage.
* **Dónde subir** (Storage):
* /projects/{projectId}/parts/{partId}/drawings/{archivo}.pdf|jpg
* /projects/{projectId}/parts/{partId}/solids/{archivo}.x\_t|step
* **Dónde guardar URLs** (Firestore en parts):
* {
* "numeroParte": "RA-00-56-001",
* "descripcionParte": "...",
* "files": {
* "drawing": {"name": "...", "url": "...", "contentType": "application/pdf"},
* "solid": {"name": "...", "url": "...", "contentType": "model/step"}
* }
* }
* **Reglas de Storage** (borrador):
* rules\_version = '2';
* service firebase.storage {
* match /b/{bucket}/o {
* match /projects/{proj}/parts/{part}/{folder}/{file} {
* allow read: if request.auth != null; // o filtra por rol
* allow write: if request.auth != null && ( // diseñador o admin
* get(/databases/(default)/documents/users/$(request.auth.uid)).data.role in ['diseñador','admin']
* );
* }
* }
* }
* **En la UI**: en NewProjectPartScreen agrega 2 botones “Subir plano (PDF/JPG)” y “Subir sólido (STEP/X\_T)”.  
  Al terminar, guarda/actualiza el doc de la parte con el bloque files.  
  Luego, en la **alta del supervisor** y en la pantalla del **operador**, muestra enlaces/preview si existen.

**G) Requisiciones (P2)**

* **Colección**: requisitions.
* {
* "proyecto": "CHM-1022-C1842",
* "parteId": "...",
* "tipo": "material|plano|otro",
* "descripcion": "Falta placa 3/8\"",
* "cantidad": 2,
* "unidad": "pz",
* "status": "abierta|aprobada|entregada|rechazada",
* "solicitanteUid": "...",
* "aprobadorUid": null,
* "createdAt": ts,
* "resolvedAt": ts
* }
* **Drawer**: “Requisiciones” con lista (filtrable por status) + formulario simple.

**H) Scrap workflow (P3)**

* Cuando un operador **carga scrap**:
  + En production\_daily: fail (o scrap) > 0, **scrapPendiente=true**.
  + Crea doc en scrap\_requests:
  + { "productionId": "...", "cantidad": 1, "causa": "golpe", "operadorUid": "...", "createdAt": ts, "status":"pendiente" }
* **Supervisores**: vista “Scrap” en el Drawer con badge si scrapPendiente==true.
* **Aprobar/Rechazar**: al aprobar, marca scrapAprobado=true en production\_daily y status='aprobado' en scrap\_requests.  
  (Opcional) **Notificación**: Cloud Function a Telegram/WhatsApp/Email.

**3) Cambios mini de código (copiables)**

**3.1. Resolver opSecuencia antes de guardar (Alta)**

Future<int> \_getSecForOperation(String operationId) async {

final doc = await FirebaseFirestore.instance.collection('operations').doc(operationId).get();

return (doc.data()?['sec'] ?? 9999) as int;

}

// al construir el payload:

final sec = await \_getSecForOperation(\_operationId!);

data['opSecuencia'] = sec;

**3.2. Backfill rápido (solo admin; botón escondido)**

Future<void> backfillOpSecuencia() async {

final ops = await FirebaseFirestore.instance.collection('operations').get();

final map = {for (var d in ops.docs) (d['nombre'] as String): (d['sec'] ?? 9999) as int};

var q = FirebaseFirestore.instance.collection('production\_daily').limit(300);

QueryDocumentSnapshot<Map<String,dynamic>>? last;

int updated = 0;

while (true) {

final snap = await (last == null ? q.get() : q.startAfterDocument(last!).get());

if (snap.docs.isEmpty) break;

final batch = FirebaseFirestore.instance.batch();

for (final d in snap.docs) {

final m = d.data();

if (m['opSecuencia'] == null) {

final name = (m['operacionNombre'] ?? '') as String;

final sec = map[name] ?? 9999;

batch.update(d.reference, {'opSecuencia': sec});

updated++;

}

}

await batch.commit();

last = snap.docs.last;

}

debugPrint('Backfill opSecuencia done: $updated docs.');

}

**3.3. Carga a Storage (diseñador → parte)**

final res = await FilePicker.platform.pickFiles(allowMultiple: false,

type: FileType.custom, allowedExtensions: ['pdf','jpg','jpeg','png','step','stp','x\_t']);

if (res != null && res.files.single.path != null) {

final file = File(res.files.single.path!);

final name = res.files.single.name;

final contentType = lookupMimeType(name) ?? 'application/octet-stream';

final ref = FirebaseStorage.instance.ref('projects/$projectId/parts/$partId/drawings/$name');

await ref.putFile(file, SettableMetadata(contentType: contentType));

final url = await ref.getDownloadURL();

await FirebaseFirestore.instance

.collection('projects').doc(projectId)

.collection('parts').doc(partId)

.set({

'files': {

'drawing': {'name': name, 'url': url, 'contentType': contentType}

}

}, SetOptions(merge: true));

}

*(Cambia la subcarpeta a solids si es STEP/X\_T; puedes decidir por extension.)*

**4) Índices que te conviene crear ya**

1. production\_daily
   * (proyecto ASC, numeroParte ASC) (Gantt/tabla)
   * (proyecto ASC, operacionNombre ASC) (gráfica filtrada por op)
   * (operadorUid ASC, fecha DESC) (vista del operador)
   * (proyecto ASC, fecha ASC) (rangos por proyecto)
2. parts *(grupo de colección)*: (activo ASC, numeroParte ASC) (ya lo tienes)
3. projects: (activo ASC, proyecto ASC) (ya lo tienes)

**5) Visibilidad por rol (P0–P1)**

* **Operador**: en dashboard/tabla, query con where('operadorUid', isEqualTo: uid) y oculta acciones que no aplican.
* **Supervisor/Admin**: ven todo; filtros por status/proyecto/operación.

**6) Qué archivos vamos a tocar**

* dashboard\_screen.dart (filtros, contadores, visibilidad por rol).
* add\_production\_entry\_screen.dart (operadores, opSecuencia, alta masiva).
* new\_project\_part\_screen.dart (subida de plano/sólido a Storage).
* gantt\_screen.dart (listo; solo asegurar orden por opSecuencia).
* profile\_screen.dart / reglas (si normalizas rol desde aquí).
* **Nuevo**: requisitions\_screen.dart, scrap\_screen.dart (listados simples).

**7) Qué te recomiendo hacer hoy (checklist rápido)**

* Crear índices arriba (2–3 minutos cada uno).
* Arreglar dropdown de **operadores** (asegurar role == 'operador').
* Agregar **filtros** (operación/operador) en **Gráfica** y cambiar agregación a **solo MANUFACTURA** por defecto.
* **Guardar opSecuencia** al crear y preparar botón **Backfill** para admin.
* Habilitar **Firebase Storage** y probar **subida de plano PDF** en NewProjectPartScreen.
* (Si te da tiempo) toggle **“Crear flujo completo”** con chips en Alta.

Cuando completes estos, seguimos con requisiciones y el flujo de scrap con badge/notificación.