¡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.