

MRB Builder - Tech stack, sikkerhet og implementeringsplan

Oppsummering av innspill og anbefalinger basert på partner-review-materialet fra Tore og vår gjennomgang.

Utarbeidet for: Fredrik Vangsal (Technical Advisor)

Dato: 20 February 2026

Kilder: (1) Digital MRB Builder - Build Status Report (19 Feb 2026). (2) Shipment Documentation Requirements (SDRL/MRB) vedlegg (PDF).

1. Executive summary

Målet er å etablere en robust, compliance-drevet MRB (Manufacturing Record Book) motor med ERP-grensesnitt, maskindata-innhenting og sikker distribusjon/arkivering. Det som allerede er levert (QMS-prosedyrer, pipeline-arkitektur og 20 ERP-interfaces) er et godt fundament. Neste kritiske steg er å spesifisere tech-stack og 'buildable' detaljnivå: datamodell, regel/valideringsmotor, dokument- og metadata-håndtering, sikkerhet (Zero Trust), tilgjengelighet og recovery-testregime.

Kjerneanbefaling: Bygg rundt en selvhostet Supabase-stack (Postgres-first) i egen cloud-konto (data suverenitet), med et tydelig skille mellom 'records' (strukturerte data) og 'artifacts' (PDF/filer). Prioriter kundeportal fra start (minimal, men ekte), og gjør hybrid cloud/on-prem til en planlagt senere fase når baseline er stabil.

2. Svar på Tore sine konkrete spørsmål

Tema	Anbefalt svar / beslutning
ERP-interfaces (20 stk)	God start i nåværende fase. Lås ikke endelig; re-evaluer etter første ende-til-ende 'mock lifecycle' og når ERP-kandidat er valgt. Fokus nå: idempotens, state machine og dokument-metadata-kontrakt.
Simuleringsdata (SDRL)	Ideelt ekte SDRL, men i Phase 0 holder syntetiske dersom Tore gjør manuell kvalitetssikring. Lag et spenn (gode + 'dårlige' SDRL) for å teste fleksibilitet, feilhåndtering og edge cases.
Kundeportal i MVP	Bygg inn kundeportal fra start (minimal) for tidlig feedback og for å kunne demonstrere konseptet for kunder/investorer/partnere. Utvikle ett konkret case (Tore+Fredrik) som intern faggruppe kan utfordre før ekstern bruk.

3. Foreslått tech-stack (kort sikt vs. lang sikt)

Dette er en stack som matcher kravene dere har diskutert: Postgres-first logikk, høy kontroll på data, robust tilgangsstyring, god tilgjengelighet med få brukere, og en vei til hybrid cloud/on-prem.

Område	Kort sikt (Phase 0-1)	Lang sikt (Phase 2+)
Kjerneplattform	Selvhostet Supabase (Postgres + Auth + Storage + Edge/Functions) i egen cloud-konto (EEA/Norge).	Evt. multi-region aktiv/standby, strengere WORM/immutability, og utvidet tenant-isolasjon.
Applikasjon	Next.js + TypeScript (app + portal + intern QA UI).	Utvidet portal, mer workflow-automatisering, evt. separate frontend-apps per rolle.
API	Start med PostgREST + RPC + evt. GraphQL gateway for "MRB Views".	GraphQL som stabilt kontraktlag + API gateway med rate limiting og policy enforcement.
Valideringsmotor	DB-naere regler: constraints, triggers, plpgsql-funksjoner + applikasjonsvaliderere for tunge regler.	Regelmotor/regel-DSL (versjonerte regler per kunde/ordre) + AI-assistert validering.
Dokument/fil-lager	Object storage (S3-kompatibelt) i egen konto + metadata i Postgres (JSONB).	WORM/immutability, dedup, livssyklus-policy (active->deep archive), og integritetsrapporter.
Observability	Grafana + Prometheus (metrics) + Loki (logs) + Tempo (traces) via OpenTelemetry.	SLO/SLA dashboard, syntetiske sjekker, mer automatisert incident response.
Testing	Jest (unit) + Playwright (E2E) + k6 (lett lasttest) + contract tests for interfaces.	Chaos/DR-øvelser, mer lasttest, sikkerhetstesting (SAST/DAST) i CI.
Dokumentasjon	GitBook (system- og prosessdokumentasjon) + ADRs i repo.	Strengere governance, versjonerte docs per release/phase.

4. Evaluering av fit mellom teknologiene

- **Supabase + Postgres** passer godt når du vil legge mye logikk i databasen (constraints, RLS, funksjoner) og samtidig få 'batteries included' for auth og storage uten å miste datasuverenitet (self-host).
- **Next.js + TypeScript** gir rask produktutvikling og type-sikkerhet for portal og intern UI. Passer godt med Postgres-first når dere bygger tydelige data-views og workflows.
- **GraphQL** er nyttig for 'MRB Views' (kunde-/ordre-spesifikke pakkestrukturer), men bør innføres kontrollert: start med klare server-side views/materialized views og en gateway slik at GraphQL ikke blir en ubegrenset spørringskanon mot produksjonsdata.
- **PDF lagres som artifact**, men all søkbar/nøkkelmetadata må normaliseres i DB (JSONB + indeks). Dette muliggjør validering og hurtig uthenting uten å åpne PDF.
- **Grafana-stacken** er et bra valg for lav brukerbase men høy tilgjengelighetskrav: den gir driftssynlighet og rask feilsøking uten tung enterprise-licensing.

5. Sikkerhet og tilgangsstyring (Zero Trust)

Målet er å unngå at kundedokumenter, tegninger og eksportkontrollert informasjon kommer på avveie. Zero Trust betyr at ingen del av systemet eller nettverket er 'implicit trusted' - alt må autentiseres, autoriseres og logges.

- **Identity-first:** all tilgang via sterke identiteter (MFA), korte tokens, og minst mulig privilegier.
- **Policy over roller:** RBAC for oversikt, men håndhev **ABAC** i praksis (kunde/ordre/rolle/clearance) via Postgres Row Level Security (RLS).
- **Tenant isolasjon:** data merkes med customer_id og order_id; alle queries må gå via RLS-beskyttede views.
- **Separate sikkerhetsdomener:** intern QA/produksjon vs kundeportal (separat app, separat auth policy, separat rate limiting).
- **Kryptering:** TLS overalt; kryptert lagring (disk + object storage). Vurder kundespesifikke KMS-nøkler for høy-sensitivt materiale.
- **Audit trail:** uforanderlig logg på alle 'read/download' av kundedokumenter + alle endringer i valideringsstatus og CoC-signering.
- **Secret management:** ingen hemmeligheter i kode/CI; bruk vault/KMS.

Autentisering - anbefaling

Start pragmatisk med Supabase Auth for Phase 0-1 (lav friksjon), men design et 'auth adapter'-lag slik at dere kan slå på enterprise SSO (Entra ID) senere. WorkOS kan være en god kandidat når (1) SSO blir krav, (2) dere trenger standardisert SCIM/provisioning og enterprise-policyer. Poenget er å gjøre dette til et utskiftbart lag, ikke en hard binding tidlig.

6. Tilgjengelighet, redundans og hybrid cloud/on-prem

Dere trenger høy tilgjengelighet selv med få brukere. Det betyr at designet må tåle nettutfall, deploy-feil og datacenter-problemer, samt ha testet recovery.

- **Definer RPO/RTO:** eksempel: RPO 15 min, RTO 2 timer (juster etter kundekrav).
- **Backup 3-2-1:** 3 kopier, 2 medietyper, 1 off-site. Test restore rutinemessig.
- **Aktiv/standby** i cloud: primær region + standby region med replikering og automatisert failover (manual approval i starten).
- **Offline-toleranse:** lokale 'edge buffers' for maskindata (MQTT broker + disk queue) som kan køe events/filer ved nettutfall.
- **Hybrid on-prem** (senere fase): etabler en on-prem replika/standby når baseline er stabil og dere har driftserfaring; start ikke med full aktiv-aktiv hybrid før dere har klare RPO/RTO og automatisert failover-prosedyre.

Hybrid-konfigurasjon (målilde)

Cloud kjører primær Supabase-stack og object storage. On-prem kjører (i en senere fase) en standby Postgres-replika og en minimal portal-cache for lokal tilgjengelighet. Failover er i første omgang kontrollert (runbook + manuell switch) og trenes i regelmessige øvelser.

7. Maskindata (MTConnect/MQTT) - prinsipper

Målet er å få produksjons- og inspeksjonsdata inn i MRB Builder med sporbarhet og robusthet.

- **Event + artifact:** MQTT for hendelser (job complete, measurement ready). Råfiler/rapporter lagres som artifacts i object storage; MQTT-payload peker til artifact_uri + checksum + metadata.
- **Canonical schema:** normaliser MTConnect/OPC-UA-data til et internt 'MachineEvent' og 'MeasurementResult' schema (uavhengig av maskinleverandør).
- **Edge broker:** kjør MQTT broker nær maskinene; replicate upstream når linja er oppe. Legg inn persistent queue for nettfall.
- **Signing/checksum:** generer SHA-256 på artifacts ved ingest og lagre i DB; bruk dette i validering/arkiv.
- **Tid og identiteter:** standardiser tid (UTC), maskin-ID, program-ID, operator (hvis relevant) og ordre/operasjon-linking.

8. Kunde-konfigurerbare MRB 'views' og maler

Du ønsker at kunder kan definere egne maler for hvordan data struktureres i dokumentpakkene, samtidig som dere beholder kontroll på sikkerhet og ytelse.

- **Skille mellom 'data model' og 'presentation model':** dere kontrollerer datamodell og regler; kunder kan kun konfigurere presentasjon (felter, rekkefølge, gruppering) innenfor et trygt rammeverk.
- **Template DSL:** definer en enkel, versjonert mal-representasjon (JSON) som peker til predefinerte 'data views' (ikke vilkårlige SQL/GraphQL-spørringer).
- **GraphQL som read-layer:** GraphQL server bør eksponere kun godkjente views og ha query complexity limits.
- **PDF som output:** pakke-bygging tar (1) template + (2) validated data + (3) artifact-lenker og genererer PDF/A + manifest.

9. Test- og recovery-regime (rutiner)

Tilgjengelighet og sikkerhet må bevises i praksis. Dette krever faste rutiner, ikke bare design.

- **Restore-test:** månedlig restore av Postgres + object storage manifest i staging. Mål faktisk RTO/RPO.
- **DR-øvelse ('fire drill'):** kvartalsvis failover-øvelse (primær -> standby) med sjekklister og etteranalyse.
- **Integritet:** verifiser SHA-256 for et utvalg artifacts etter restore. Kjør 'annual archive integrity report' på hele arkivet.
- **Sikkerhetstester:** SAST i CI hver PR, dependency scanning, og periodisk pentest for portal.
- **Kontraktstester:** ERP-interfaces (20 stk) og portal API testes med contract tests slik at endringer ikke bryter integrasjon.

10. Implementeringsplan - kort og lang sikt

Planen under er et forslag til hvordan dere kan ta dette fra partner-review til byggbar spesifikasjon og første fungerende system, med tydelig prioritet på markedsnære leveranser.

Horisont	Fokus	Konkrete leveranser
0-6 uker	Gjøre systemet 'buildable'	WP-1 IT/System Spec (Supabase self-host + sikkerhet + deploy), v0.1 datamodell (identiteter/sporbarhet), Document Type Registry, Portal MVP scope, test-/DR-policy (RPO/RTO).
Phase 0 (Q2-Q4 2026)	Simulering & bygg	Syntetiske SDRL-sett (variasjon), end-to-end mock lifecycles, første portal med login og MRB-download, ingest pipeline (MQTT broker + artifact storage), grunnvalidering (L1-L3) for 2-3 doc-typer.
Phase 0.5 (Q1-Q2 2027)	Integrasjon & test	ERP-integrasjon mot valgt ERP, maskindata fra ekte CNC/CMM, utvidet validering (L4-L5) for kjerne-dokumenttyper, recovery-øvelser, signeringsflyt for CoC.
Phase 1 (Q3 2027)	Produksjon MVP	Produksjon MRB, driftsovervåkning, support-prosess, portal brukt av første kunder, kontrollert governance for maler.
Phase 2+	Skalering & automatisering	Enterprise SSO (Entra) hvis behov, hybrid on-prem standby, AI-assistert validering, mer kundeselvtjenning og APIer.

11. Åpne punkter som bør avklares tidlig

- Valg av hosting (cloud-leverandør, regioner, KMS) og hvordan data isoleres per kunde (tenant).
- Eksakt modell for kundemaler (hva kan kunden konfigurere uten å kunne lekket data eller trigge dyre queries).
- Hvilke dokumenttyper som er 'MVP-critical' for markedsnær leveranse (EN10204, CoC, ITP/FAIR, inspection reports).
- E-signature strategi for CoC (PKI-basert PDF signing vs ekstern leverandør) og hvem som er 'authorized signatory'.