Decisión: Usaremos un enfoque híbrido C4 + UML.

C4 (nivel Contenedor/Componente): para comunicar la estructura macro y los componentes implementables.

UML (Secuencia/Despliegue): para detallar interacciones temporales y la infraestructura física/lógica

Tecnologías fijadas por restricción: C# (.NET 8), SQL Server (on-premise), herramientas de laboratorio de última generación (integración mediante adaptadores/ETL u OPC/CSV), Postman/JMeter para pruebas.

1) Backlog inicial (épicas → historias → criterios)

Épica E1: Registro y verificación de datos (RF01, RF02)

H1.1 Ingesta de datos de laboratorio

Como Operador, quiero cargar datos desde el equipo (archivo/endpoint), para registrarlos con trazabilidad.

Criterios: admite lote CSV/JSON; valida esquema; persiste con sello de tiempo, ID de lote, operador.

H1.2 Motor de validación de estándares

Como Analista, quiero configurar reglas/umbrales por tipo de grano, para comparar automáticamente.

Criterios: reglas CRUD; validación sincrónica y por lote; resultado: Conforme/No Conforme + métricas desviación.

H1.3 Auditoría y trazabilidad

Criterios: cada cambio guarda quién/cuándo/qué

Épica E2: Informes y visualización (RF01 – Salidas)

H2.1 Informe de calidad por lote (PDF/HTML)

Criterios: cabecera, parámetros, resultados, no conformidades, firma digital interna.

H2.2 Consulta histórica y filtros

Criterios: por rango de fechas, tipo de grano, equipo, operador; tiempos de respuesta ≤ 2 s en 95º percentil (RNF01).

Épica E3: Plataforma y seguridad (RNF)

H3.1 Autenticación y perfiles (Analista / Operador)

Criterios: control de acceso por rol; bitácora de accesos.

H3.2 Escalabilidad y rendimiento (RNF01, RNF02)

Criterios: pruebas con 1M registros; uso de índices/particiones; pruebas JMeter con ≥ 200 req/min estable.

2) Plan de trabajo – 3 sprints (1 semana cada uno)

Cada sprint culmina con revisión y retroalimentación; se ajustan artefactos según comentarios.

Sprint 1 – Arquitectura y diseño (Enfoque de entregable: Diagramas y decisiones)

Objetivo: Definir arquitectura, componentes y cómo se despliega.

Día 1-2: Investigar y fijar enfoque UML vs C4 (decisión: híbrido); definir límites de contexto y contenedores.

Día 3: Modelar módulos y dependencias internas (componentes, interfaces, contratos API superficiales).

Sprint 2 – Implementación núcleo (Enfoque de entregable: Algoritmos y datos)

Objetivo: Construir el esqueleto funcional: ingesta, validación, persistencia, reglas.

Tareas: Proyecto .NET (API + Worker) y solución C#.

Data Ingestion Service (CSV/JSON + validación de esquema).

Validation Engine con reglas parametrizables por tipo de grano.

Modelo de datos SQL Server (tablas: Lote, Medicion, Regla, ResultadoValidacion, Usuario, Bitacora).

Endpoints iniciales: carga de lote, ejecutar validación, consultar resultados.

Pruebas unitarias básicas y colección Postman.

Entregables: código base, scripts SQL, colección Postman, guía de despliegue local.

Sprint 3 – Informes y calidad (Enfoque de entregable: Pruebas e informes)

Objetivo: Generar informes, optimizar rendimiento y asegurar RNF.

Tareas:

Servicio de Reportes (HTML→PDF) con plantilla por tipo de grano.

Filtros y consultas históricas.

Autenticación y roles (Operador/Analista).

Pruebas de rendimiento con JMeter; índice/particionado; ajustes de queries.

Documentación final y script de despliegue.

Entregables: informes PDF, resultados de pruebas (gráficas/CSV), manual de usuario/admin.

4) Cronograma propuesto (por día)

Semana 1 (Arquitectura): D1–D2 componentes/enfoque; D3 módulos e interacciones; D4 secuencia; D5 despliegue y revisión.

Semana 2 (Algoritmos y datos): D1 solución .NET y esquema DB; D2 ingesta; D3 validación; D4 endpoints; D5 pruebas unitarias + Postman.

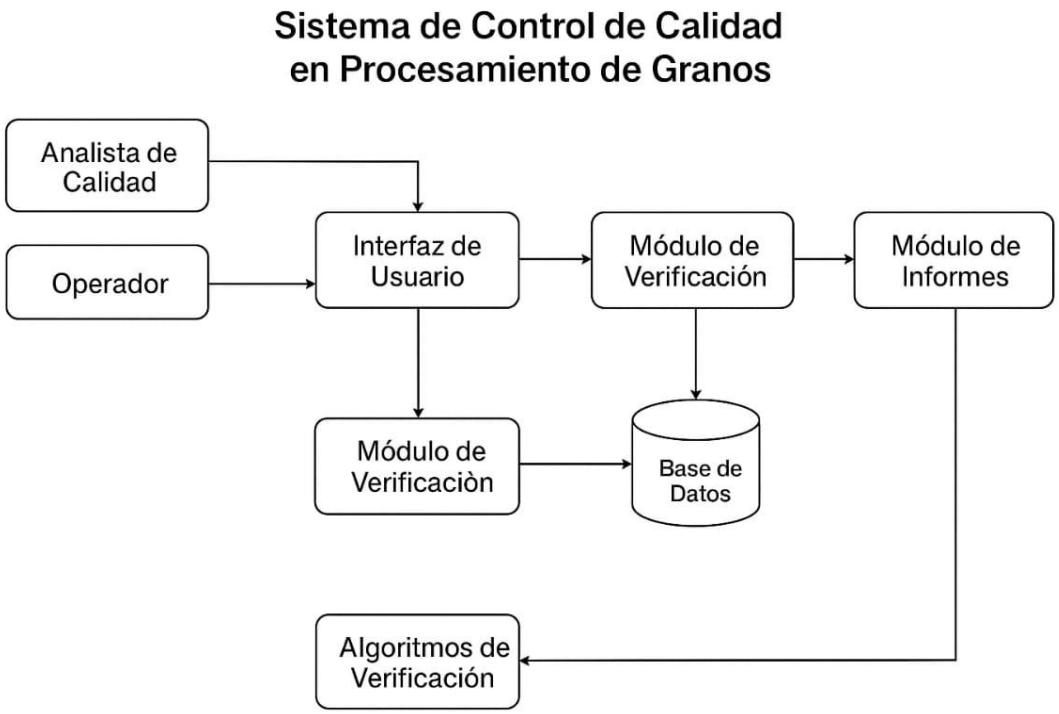
Semana 3 (Informes y calidad): D1 reportes; D2 consultas/filtros; D3 auth/roles; D4 JMeter y optimizaciones; D5 documentación y demo.

5) Riesgos y mitigaciones

Integración con equipos propietarios: plan B con exportación CSV; sandbox de datos sintéticos.

Rendimiento con grandes lotes: pruebas tempranas en Sprint 3; índices compuestos y particionado por fecha/lote.

Cambios en reglas: versionado de reglas y resultados; migraciones controladas.



Optamos por un enfoque híbrido C4 + UML para equilibrar claridad y precisión técnica. Con C4, modelamos los contenedores y componentes que materializan los requerimientos: una Web UI para operadores y analistas, un API Backend que expone contratos estables, y servicios de dominio especializados (Ingesta, Validación y Reportes) que encapsulan reglas de negocio. Esto nos permite evolucionar cada pieza de forma independiente y facilita pruebas y despliegues parciales. En la base de datos SQL Server separamos dominios (Lote/Medición, Reglas, Resultados, Usuarios, Bitácora) para privilegiar consultas rápidas y auditoría.

El diagrama de Secuencia modela el caso de uso crítico: carga de datos → validación automática → generación de informes. Separamos responsabilidades: la ingesta normaliza y persiste, el motor de validación aplica reglas configurables por el Analista y devuelve conformidad/no conformidades, y el servicio de reportes construye un PDF con trazabilidad completa. Este acoplamiento débil favorece pruebas unitarias y la sustitución de adaptadores de laboratorio (CSV/OPC) sin impactar el dominio.

En Despliegue, priorizamos un escenario on-premise acorde a la restricción de servidores locales y compatibilidad con equipos de laboratorio. Un App Server .NET expone la UI y el API; dos servicios Windows (o Workers) ejecutan ingesta/validación y reportes; todo respaldado por SQL Server en la misma red para minimizar latencia y facilitar control de acceso. Esta topología simplifica la operación inicial; a futuro, puede escalarse con separación de nodos (DB dedicada, balanceador web) y colas para desacoplar procesos pesados.

En los no funcionales, fijamos metas medibles: p95 ≤ 2 s en consultas, estabilidad ≥ 200 req/min y soporte a millones de registros, habilitado por índices, particiones y consultas optimizadas. Para la seguridad, implementamos autenticación/roles y bitácora completa, requisito clave de trazabilidad en calidad.