Curso
Herramientas de Computación en la Nube

Clase 1

Febrero 7 de 2026

Aplicaciones confiables, escalables y mantenibles

Este documento se fundamenta en los principios de arquitectura y sistemas de datos descritos en **Designing Data-Intensive Applications**, *aplicados a un caso académico de analítica operativa. Consultar la bibliografía al final.*

Profesor

Alvaro Mauricio Montenegro Díaz
Universidad de la Sabana

Febrero 2026

Fuente: Creada por Chat GPT

Esta lección establece el marco conceptual que guiará todas las decisiones técnicas del curso. Ninguna tecnología se estudiará como fin en sí mismo, sino como una respuesta a compromisos entre confiabilidad, escalabilidad y mantenibilidad.

Las lecciones posteriores —almacenamiento, procesamiento distribuido, pipelines y analítica— deben entenderse como aplicaciones concretas de estos principios, no como catálogos de herramientas.

Introducción

Las aplicaciones modernas orientadas a datos se caracterizan por estar limitadas principalmente por el **volumen, la complejidad y la velocidad de cambio de la información**, más que por la capacidad de cómputo. Este tipo de aplicaciones se construye a partir de componentes estándar —almacenamiento persistente, cachés, índices de búsqueda, sistemas de mensajería y procesamiento por lotes— que, al combinarse, forman **sistemas de datos compuestos** con propiedades emergentes. Un sistema de datos no es la suma de sus componentes individuales, sino el resultado emergente de sus interacciones. Por esta razón, decisiones aparentemente locales pueden tener efectos globales significativos en el comportamiento del sistema.

El diseño de estos sistemas requiere decisiones arquitectónicas explícitas, ya que la integración de múltiples herramientas introduce nuevos compromisos en términos de consistencia, rendimiento y operabilidad.

1. Los compromisos son inevitables

En arquitectura de sistemas no existen soluciones universalmente óptimas. Toda decisión técnica implica priorizar ciertos atributos de calidad en detrimento de otros. Algunos compromisos clásicos incluyen:

- consistencia vs. Disponibilidad,
- simplicidad vs. Flexibilidad,
- aislamiento vs. Rendimiento,
- automatización vs. control manual.

Comprender estos compromisos es más importante que memorizar tecnologías específicas. El objetivo del diseño no es eliminarlos, sino hacerlos explícitos y gestionables.

2. Confiabilidad

La confiabilidad se define como la capacidad de un sistema para **funcionar correctamente incluso en presencia de condiciones adversas**. Un sistema confiable no solo evita caídas, sino que cumple su función esperada, mantiene niveles aceptables de rendimiento y protege los datos frente a accesos indebidos o usos incorrectos.

Es fundamental distinguir entre:

- **Falla (fault)**: desviación de un componente respecto a su comportamiento esperado.
- **Fallo (failure)**: incapacidad del sistema completo para prestar el servicio requerido.

Dado que las fallas son inevitables, el objetivo del diseño no es eliminarlas, sino **impedir que se propaguen hasta convertirse en fallos visibles para el usuario**. Esto se logra mediante técnicas de tolerancia a fallas y diseño defensivo.

Las fuentes principales de fallas incluyen:

- **Fallas de hardware**, como errores de disco, memoria, red o suministro eléctrico. Estas son estadísticamente inevitables y se mitigan mediante redundancia, replicación y la capacidad de tolerar la pérdida de nodos completos, especialmente en entornos distribuidos y en la nube.
- **Errores de software**, que suelen ser sistemáticos y correlacionados. Un mismo defecto puede afectar simultáneamente a múltiples instancias y provocar fallas en cascada. La mitigación requiere pruebas exhaustivas, aislamiento de componentes, reinicios controlados y una fuerte instrumentación para detectar comportamientos anómalos en producción.
- **Errores humanos**, que constituyen una de las principales causas de interrupciones en sistemas reales. Para reducir su impacto se aplican principios como la simplificación de interfaces operativas, la separación entre entornos de prueba y producción, despliegues graduales, reversión rápida de cambios y una observabilidad clara del estado del sistema. Para reducir el impacto de errores humanos se aplican principios como la simplificación de interfaces operativas, la separación estricta de entornos, despliegues graduales, reversión rápida de cambios y una observabilidad clara del estado del sistema.

Una práctica avanzada asociada a la confiabilidad es la **inyección deliberada de fallas**, cuyo objetivo es validar que los mecanismos de tolerancia funcionan correctamente bajo condiciones reales.

3. Escalabilidad

La escalabilidad describe la capacidad de un sistema para **mantener un nivel aceptable de rendimiento a medida que aumenta la carga**. No es una propiedad absoluta, sino relativa a un conjunto concreto de condiciones: tipo de carga, métricas de rendimiento y objetivos de servicio.

El análisis de escalabilidad comienza con la **descripción precisa de la carga**, mediante parámetros como solicitudes por segundo, proporción entre lecturas y escrituras, número de usuarios concurrentes o distribuciones extremas que generan colas largas. El rendimiento debe analizarse como una **distribución de latencias**, no como un promedio. *Los sistemas fallan en los extremos, no en el promedio.*

En muchos sistemas, el comportamiento está dominado por estos valores extremos más que por los promedios.

El rendimiento debe medirse como una **distribución de tiempos de respuesta**, no como un valor medio. Los percentiles altos (p95, p99, p99.9) reflejan la experiencia real de los usuarios y son críticos para definir objetivos de nivel de servicio. Fenómenos como el retraso por colas, el bloqueo de solicitudes y la amplificación de latencias en arquitecturas compuestas hacen que unas pocas operaciones lentas degraden de forma desproporcionada el comportamiento global.

Las estrategias para afrontar el crecimiento incluyen:

- **Escalado vertical**, aumentando la capacidad de una sola máquina.
- **Escalado horizontal**, distribuyendo la carga entre múltiples nodos independientes.
- **Elasticidad**, ajustando dinámicamente los recursos según la demanda.

No existe una solución universal: cada enfoque implica compromisos entre costo, complejidad y previsibilidad.

4. Mantenibilidad

La mantenibilidad se refiere a la capacidad de un sistema para **ser operado, entendido y modificado de manera eficiente a lo largo del tiempo**. Dado que la mayor parte del costo del software se incurre después de su despliegue inicial, este atributo es tan crítico como la funcionalidad.

La mantenibilidad se apoya en tres pilares principales:

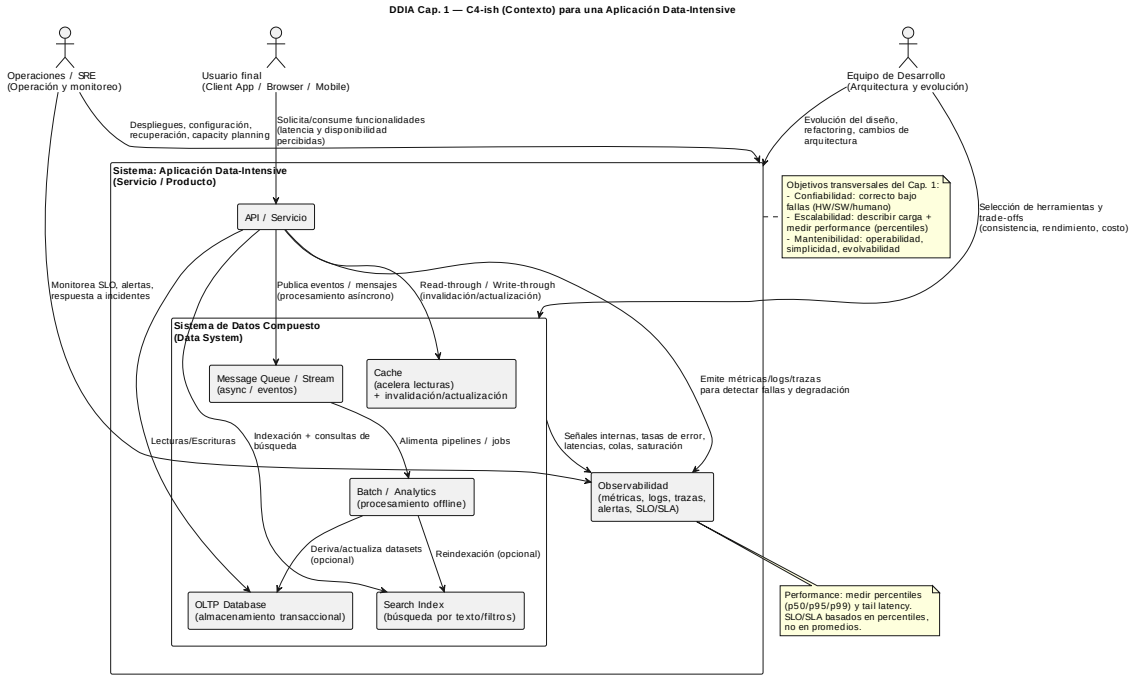
- **Operabilidad**, que busca facilitar el trabajo de los equipos responsables del funcionamiento diario del sistema. Incluye monitorización, automatización, despliegues seguros, recuperación rápida ante incidentes, planificación de capacidad y preservación del conocimiento operativo.

- **Simplicidad**, entendida como la reducción de la complejidad accidental. Los sistemas tienden a degradarse con el tiempo debido al acoplamiento excesivo, la proliferación de casos especiales y la acumulación de estados implícitos. Las abstracciones claras y bien definidas permiten ocultar detalles innecesarios y limitar el impacto de los cambios.
- **Evolvabilidad**, que reconoce que los requisitos cambian inevitablemente por factores técnicos, organizativos o regulatorios. Un sistema evolvable puede adaptarse sin reescrituras masivas, gracias a una arquitectura flexible, modular y basada en buenas abstracciones.

5. Síntesis

Un sistema bien diseñado equilibra confiabilidad, escalabilidad y mantenibilidad mediante decisiones conscientes y fundamentadas. La clave no reside en la adopción de tecnologías específicas, sino en **comprender los compromisos** y diseñar estructuras que toleren fallas, crezcan de forma controlada y puedan evolucionar con el tiempo sin incurrir en complejidad innecesaria. La complejidad no gestionada es una forma de deuda que siempre se cobra con intereses, usualmente en los momentos más críticos del ciclo de vida del sistema.

Pregunta guía del curso: *¿Qué atributo de calidad se está priorizando cuando se toma una decisión técnica concreta, y cuál se está sacrificando explícitamente?*



Contexto de una aplicación intensiva en datos. Fuente: Creada por Chat GPT

6. Diccionario de términos usados

Sistema de datos (Data-Intensive System)

Sistema cuyo comportamiento está dominado por la gestión, transformación y movimiento de datos, más que por el cómputo puro.

Confiabilidad (Reliability)

Capacidad de un sistema para funcionar correctamente bajo condiciones normales y adversas, manteniendo integridad de datos y servicio esperado.

Falla (Fault)

Desviación de un componente respecto a su comportamiento esperado.

Fallo (Failure)

Situación en la que el sistema completo deja de prestar el servicio requerido al usuario.

Tolerancia a fallas

Conjunto de técnicas de diseño que impiden que fallas locales se propaguen y se conviertan en fallos visibles.

Redundancia

Duplicación intencional de componentes o datos para mitigar fallas inevitables.

Replicación

Mecanismo mediante el cual los datos se mantienen en múltiples ubicaciones para mejorar confiabilidad y disponibilidad.

Errores humanos

Fallas introducidas por operadores o desarrolladores, mitigables mediante diseño, automatización, aislamiento y observabilidad.

Inyección de fallas

Práctica deliberada de introducir fallas controladas para validar mecanismos de tolerancia en condiciones reales.

Escalabilidad (Scalability)

Capacidad de un sistema para mantener niveles aceptables de rendimiento ante el crecimiento de la carga.

Carga (Load)

Descripción cuantitativa del trabajo que enfrenta el sistema (solicitudes, usuarios, volumen de datos, patrones extremos).

Latencia

Tiempo que tarda una solicitud en completarse, medido como distribución y no como promedio.

Percentiles (p95, p99, p99.9)

Medidas estadísticas que capturan el comportamiento en los extremos de la latencia y reflejan la experiencia real del usuario.

Retraso por colas (Queueing Delay)

Incremento de latencia causado por acumulación de solicitudes en sistemas saturados.

Escalado vertical

Aumento de capacidad mediante el fortalecimiento de una sola máquina.

Escalado horizontal

Aumento de capacidad mediante la adición de múltiples nodos independientes.

Elasticidad

Capacidad de ajustar dinámicamente los recursos según la demanda.

Mantenibilidad (Maintainability)

Facilidad con la que un sistema puede ser operado, entendido y modificado a lo largo del tiempo.

Operabilidad

Conjunto de prácticas que facilitan la operación diaria del sistema: monitoreo, automatización, recuperación y despliegue seguro.

Simplicidad

Reducción de complejidad accidental mediante abstracciones claras y acoplamiento controlado.

Evolvabilidad

Capacidad de un sistema para adaptarse a cambios de requisitos sin reescrituras masivas.

Observabilidad

Capacidad de inferir el estado interno de un sistema a partir de métricas, logs y trazas.

Propiedades emergentes

Comportamientos del sistema que surgen de la interacción de componentes, no de componentes individuales.

7. Bibliografía de referencia

7.1.1 Arquitectura de datos y lakehouse

- Armbrust, M., et al. (2021). *Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics*. Proceedings of CIDR.
- Kleppmann, M. (2017). *Designing Data-Intensive Applications*. O'Reilly Media.

Ingeniería de datos y pipelines

- Reis, J., & Housley, M. (2022). *Fundamentals of Data Engineering*. O'Reilly Media.
- Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit* (3rd ed.). Wiley.

Analítica operativa y KPIs

- Eckerson, W. (2010). *Performance Dashboards: Measuring, Monitoring, and Managing Your Business*. Wiley.
- Marr, B. (2016). *Key Performance Indicators: Developing, Implementing, and Using Winning KPIs*. Pearson.

Gobierno, calidad y trazabilidad de datos

- DAMA International. (2017). *DAMA-DMBOK: Data Management Body of Knowledge* (2nd ed.). Technics Publications.
- ISO/IEC 25012:2008. *Data Quality Model*.