Microservicios: diseño por dominio, integración y resiliencia

W. J. Carrillo Sandoval
Carnet: 7690-21-3740
wcarrillos1@miumg.edu.gt

Resumen
Los microservicios dividen el sistema en unidades pequeñas con despliegue independiente y propiedad de datos. Se cubren límites de dominio (DDD, bounded contexts), contratos (REST/gRPC), mensajería/eventos, consistencia eventual (sagas, outbox), pruebas de contrato, observabilidad y patrones de resiliencia (timeouts, reintentos con *jitter*, circuit breakers, bulkheads, rate limits).
Palabras clave: microservicios, DDD, eventos, sagas, trazabilidad, resiliencia

1. Diseño por dominio

Cada servicio encapsula su modelo y su base de datos (*bounded context*). Evitar *joins* inter-servicio; preferir *eventos de dominio* y materializaciones locales. La Ley de Conway sugiere alinear equipos pequeños con cada dominio.

2. Contratos y versiones

APIs REST para interoperabilidad amplia y gRPC para baja latencia. Definir esquemas versionados (OpenAPI/Protobuf) y *consumer-driven contract testing* para prevenir rupturas.

3. Integración asíncrona y consistencia

Mensajería (Kafka/RabbitMQ) para desacople temporal. Patrón *outbox* publica eventos de manera atómica. *Sagas* coordinan pasos distribuidos; consumidores idempotentes y *dead-letter queues* manejan fallos.

4. Resiliencia operativa

Timeouts y reintentos exponenciales con *jitter*; *circuit breakers* para fallos remotos; *bulkheads* para aislar recursos; *rate limiting* para proteger endpoints.

5. Observabilidad

Trazas distribuidas (correlación por *trace/span id*), métricas por endpoint (latencia, error, saturación) y logs estructurados. SLOs con *error budgets* para gobernar canarios/rollbacks.

6. Service mesh (opcional)

mTLS, *traffic shifting* y políticas sin tocar el código; evaluar su complejidad antes de adoptarlo.

7. Observaciones y comentarios

Sin límites claros y observabilidad, aparece el "monolito distribuido". La calidad del contrato es tan importante como el código.

8. Conclusiones

1. Propiedad de datos por servicio y contratos firmes sostienen el desacoplamiento.
2. Sagas + outbox equilibran integridad y agilidad en transacciones distribuidas.
3. La observabilidad reduce MTTR y evita fallas silenciosas entre servicios.

Bibliografía

1. Newman, S. (2021). *Building Microservices* (2a ed.). O'Reilly.
2. Fowler, M., & Lewis, J. (2014). *Microservices*. `https://martinfowler.com/articles/microservices.html`
3. Richards, M., & Ford, N. (2020). *Fundamentals of Software Architecture*. O'Reilly.