¿Cómo elegir la arquitectura de la aplicación según su tipo?

William Carrillo
Universidad Mariano Gálvez
Facultad de Ingeniería en Sistemas

9 de agosto de 2025

1. Introducción

Seleccionar la arquitectura adecuada para una aplicación es una de las decisiones más importantes en el proceso de desarrollo de software. Una elección acertada puede significar un sistema más fácil de mantener, con mejor desempeño y preparado para evolucionar; por el contrario, una decisión inadecuada puede derivar en altos costos, problemas de escalabilidad y dificultades en la operación. La arquitectura no solo define la estructura técnica del sistema, sino que también condiciona la manera en que los equipos trabajan, cómo se distribuye la carga de desarrollo y qué tan flexible será la solución ante cambios futuros. Este documento analiza las principales arquitecturas, sus características, y presenta un conjunto de criterios prácticos para elegir la más apropiada según el tipo de aplicación y el contexto de negocio.

2. Principales Tipos de Arquitectura

- **Monolito**: Estructura en la que todo el código se despliega como una única unidad. Es recomendable en proyectos pequeños, prototipos o MVPs donde la prioridad es la velocidad de entrega y la simplicidad en la gestión.
- **Monolito Modular**: Mantiene un único despliegue, pero con separación clara de módulos internos, lo que mejora el mantenimiento y permite, en el futuro, migrar a arquitecturas distribuidas sin rehacer todo el sistema.

- **Microservicios**: Conjunto de servicios independientes, cada uno responsable de un conjunto específico de funciones. Favorece la escalabilidad y el trabajo paralelo de múltiples equipos, pero exige una infraestructura madura y capacidades sólidas de monitoreo y despliegue continuo.
- **Serverless**: Basado en funciones administradas por un proveedor cloud, con ejecución bajo demanda y pago por uso. Es ideal para cargas de trabajo variables o impredecibles, aunque con limitaciones en control y dependencia del proveedor.
- **Event-Driven**: Arquitectura orientada a eventos, donde los componentes se comunican de manera asíncrona. Facilita el desacoplamiento y es adecuada para sistemas con procesos que pueden ejecutarse en paralelo o de forma independiente.
- **Arquitecturas Limpias o Hexagonales**: Diseños que priorizan el dominio de negocio separando las reglas de los detalles técnicos, aplicables tanto en monolitos como en arquitecturas distribuidas.

3. Criterios de Selección

Al elegir una arquitectura se deben evaluar factores técnicos, organizacionales y de negocio. Entre los más relevantes:

1. **Tamaño y complejidad del dominio**: Un sistema con reglas simples puede funcionar eficientemente con un monolito; en cambio, un dominio amplio y complejo requiere modularidad o distribución.
2. **Escalabilidad esperada**: Determinar si el sistema debe soportar picos de tráfico, carga constante o crecimiento gradual.
3. **Capacidades del equipo**: Un equipo con poca experiencia en despliegues distribuidos tendrá mejores resultados iniciando con una arquitectura más simple.
4. **Tiempo de entrega**: Cuando el *time-to-market* es crítico, conviene optar por arquitecturas que permitan entregar valor rápidamente y evolucionar después.
5. **Requisitos no funcionales**: Considerar necesidades de disponibilidad, tiempos de respuesta, auditoría, seguridad y cumplimiento normativo.
6. **Presupuesto y recursos técnicos**: Arquitecturas complejas como microservicios o event-driven requieren inversión en herramientas y personal especializado.

4. Recomendaciones por Escenario

4.1. Aplicaciones Pequeñas o MVP

La mejor opción es iniciar con un **monolito** o **monolito modular**. Esto permite avanzar rápido en desarrollo, reducir costos iniciales y evitar sobrecargar al equipo con complejidades innecesarias.

4.2. Aplicaciones Empresariales con Múltiples Equipos

En casos donde el sistema cubra múltiples áreas de negocio y requiera que diferentes equipos trabajen en paralelo, los **microservicios** son recomendables. Es importante que cada servicio tenga límites claros y se gestione con buenas prácticas de CI/CD.

4.3. Sistemas con Carga Variable

Para sistemas que reciben picos de tráfico o con uso intermitente, el enfoque **serverless** ofrece escalado automático y costos proporcionales al consumo real.

4.4. Integraciones y Procesos Asíncronos

En entornos con alto intercambio de información entre sistemas y procesos que no requieren respuesta inmediata, la arquitectura **event-driven** proporciona flexibilidad y desacoplamiento.

5. Errores Comunes a Evitar

- Adoptar microservicios antes de tiempo, lo que eleva la complejidad sin aportar beneficios inmediatos.
- Implementar serverless sin considerar limitaciones de latencia o dependencia de un proveedor.
- Diseñar un monolito sin modularidad, lo que dificulta su mantenimiento y escalabilidad futura.
- No definir reglas claras para la comunicación entre componentes, lo que genera acoplamiento excesivo.

6. Conclusiones

La elección de la arquitectura debe estar alineada con los objetivos del negocio y las capacidades técnicas disponibles. No existe una única respuesta correcta; lo ideal es comenzar con la opción más simple que cumpla los requisitos actuales, y evolucionar la arquitectura conforme el proyecto crezca en complejidad y demanda. Una buena práctica es planificar la arquitectura con una visión de largo plazo, incorporando patrones que permitan cambios graduales sin interrupciones significativas.

7. Referencias Bibliográficas

- Richards, M., & Ford, N. (2020). *Fundamentals of Software Architecture*. O'Reilly Media.
- Newman, S. (2021). *Building Microservices*. O'Reilly Media.
- Fowler, M. (2015). *MonolithFirst*. Recuperado de: `https://martinfowler.com/bliki/MonolithFirst.html`
- AWS. (2024). *Serverless Architectures*. Recuperado de: `https://aws.amazon.com/serverless/`
- Reactive Manifesto. (2014). *The Reactive Manifesto*. Recuperado de: `https://www.reactivemanifesto.org/`