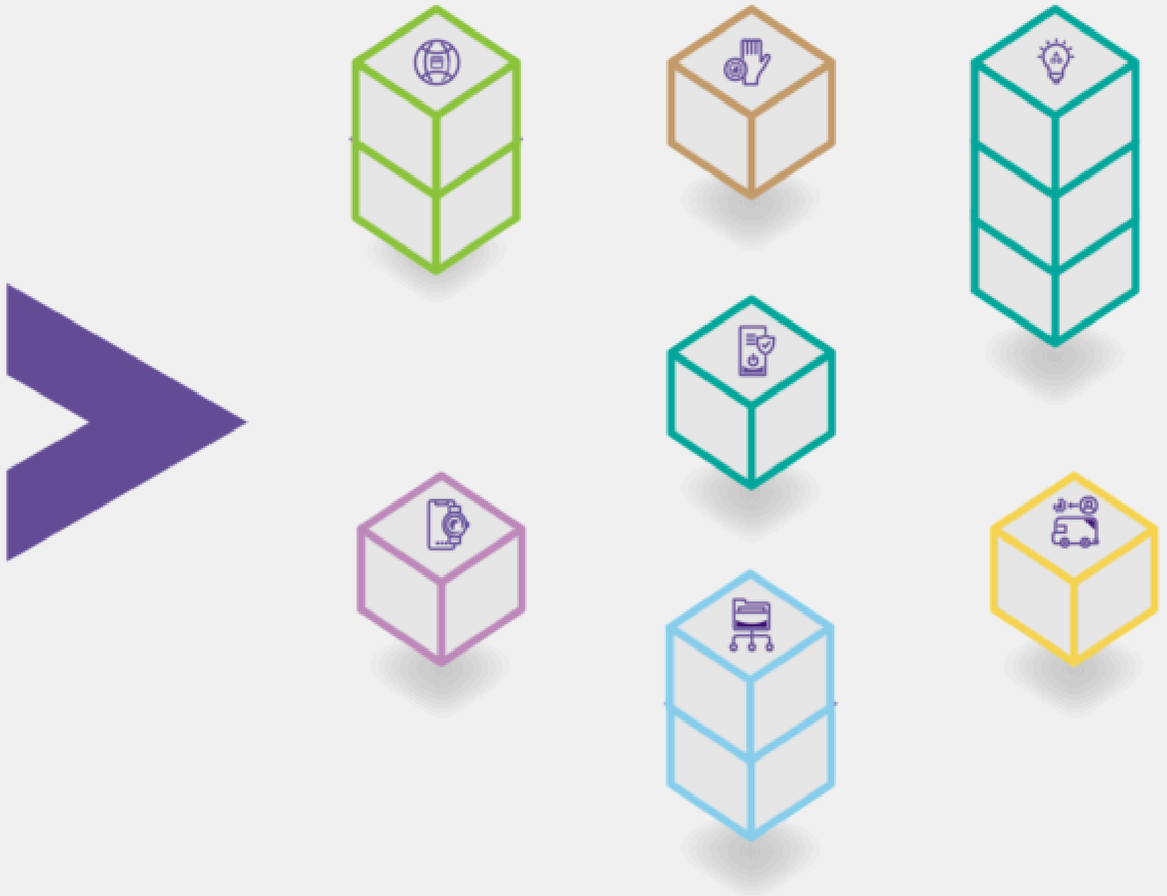
Monolitica vs Microservicios

Monolito



Microservicios

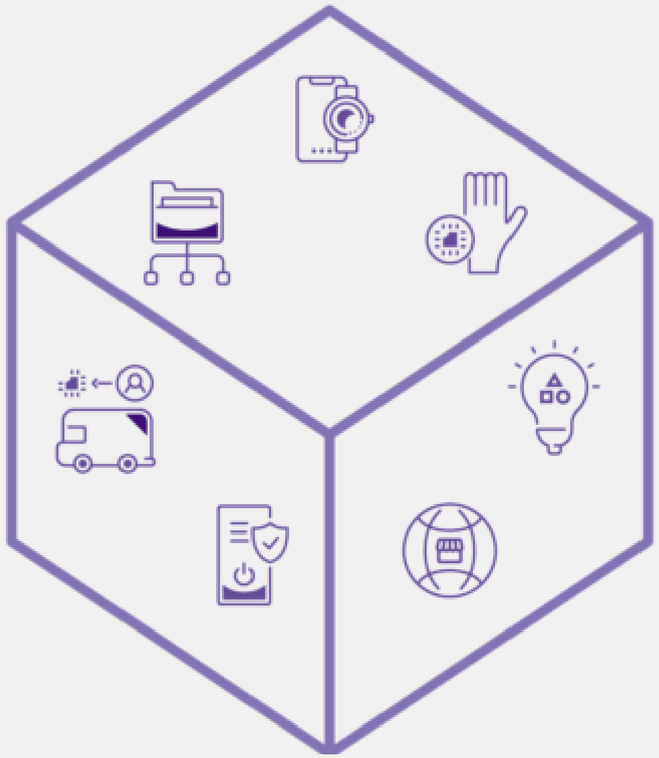
Arquitectura Monilitica

- Es un modelo de desarrollo de software que integra todos los componentes de una aplicación en un solo sistema.
- El código, la base de datos y la interfaz de usuario están acoplados y se ejecutan como una sola entidad.
- Todos los componentes de software de un sistema monolítico son interdependientes.
- Las aplicaciones monolíticas pueden ser útiles al principio de un proyecto para aliviar la sobrecarga cognitiva de la gestión de código

Arquitectura de Microservicios

- Es un estilo arquitectónico que permite dividir una aplicación en partes independientes, llamadas microservicios.
- Cada microservicio tiene su propia lógica de negocio y base de datos.
- Los microservicios se comunican entre sí a través de API.
- Cada microservicio se puede desarrollar, implementar y escalar de forma independiente.
- Los microservicios permiten actualizaciones y correcciones rápidas con un impacto mínimo en el sistema.
- Los microservicios permiten probar nuevas ideas y revertirlas si algo no funciona.

Ventajas de una arquitectura monolítica



Implementacion sencilla

un único archivo o directorio ejecutable facilita la implementación.

Desarrollo

desarrollar una aplicación compilada con una única base de código es más sencillo.

Rendimiento

en una base de código y un repositorio centralizados, una API suele realizar la misma función que muchas API en el caso de los microservicios.

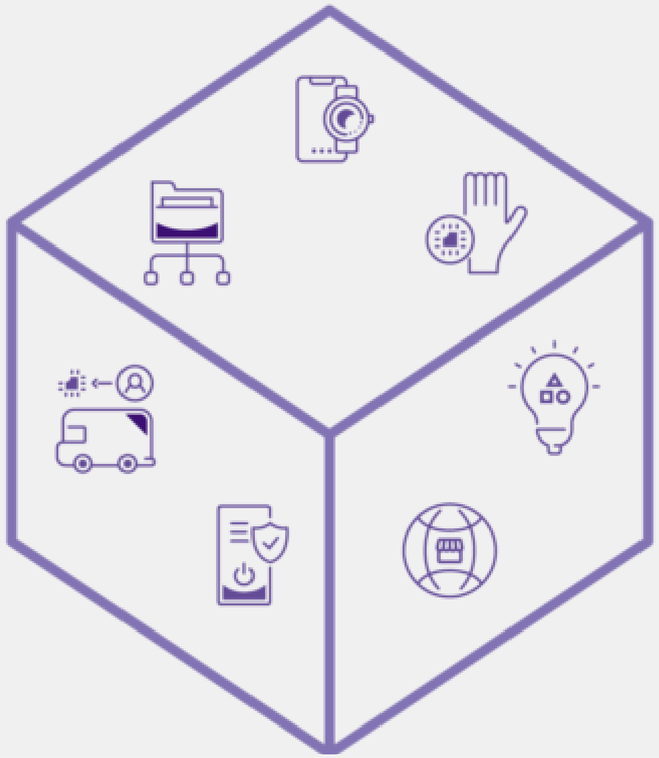
Pruebas simplificadas

una aplicación monolítica es una unidad única y centralizada, por lo que las pruebas integrales se pueden hacer más rápido que con una aplicación distribuida.

Depuración sencilla

con todo el código ubicado en un solo lugar, es más fácil rastrear las solicitudes y localizar incidencias.

Desventajas de una arquitectura monolítica



Velocidad de desarrollo más lenta

con una aplicación grande y monolítica, el desarrollo es más complejo y lento.

Escalabilidad

no se pueden escalar componentes individuales.

Fiabilidad

si hay un error en algún módulo, puede afectar a la disponibilidad de toda la aplicación.

Barrera para la adopción de tecnología

cualquier cambio en el marco o el lenguaje afecta a toda la aplicación, lo que hace que los cambios suelan ser costosos y lentos.

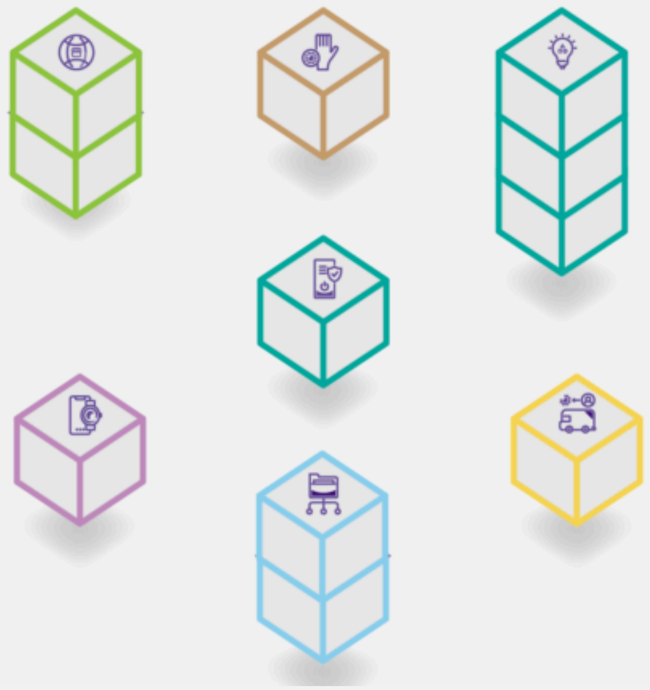
Falta de flexibilidad

un monolito está limitado por las tecnologías que se utilizan en él.

Implementación

un pequeño cambio en una aplicación monolítica requiere una nueva implementación de todo el monolito.

Ventajas de los microservicios



Agilidad

promueve formas ágiles de trabajar con equipos pequeños que implementen con frecuencia.

Escalado flexible

si un microservicio está alcanzando su capacidad de carga, se pueden implementar rápidamente nuevas instancias de ese servicio al clúster que lo acompaña para aliviar la presión. Ahora tenemos varios inquilinos y no tenemos control de estado, con clientes repartidos en varias instancias. Así, podemos respaldar tamaños de instancias mucho mayores

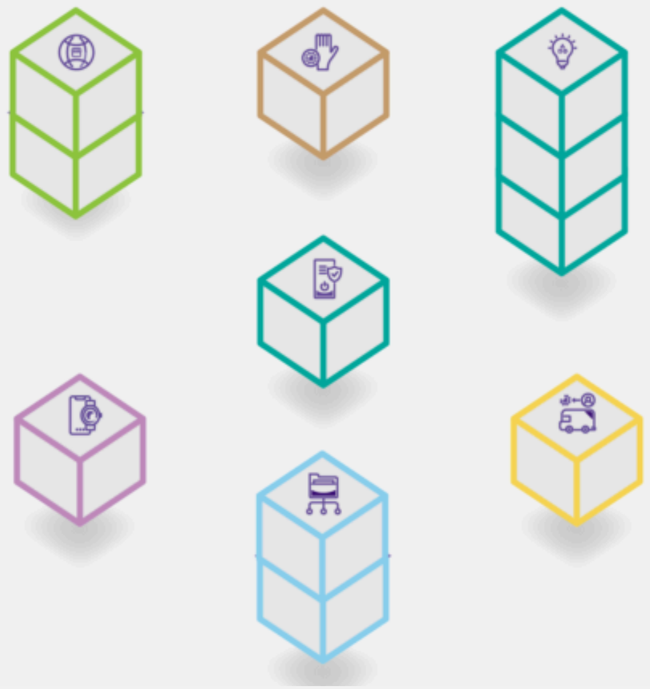
Implementación continua:

tenemos ciclos de lanzamiento frecuentes y más rápidos. Antes enviábamos actualizaciones una vez a la semana y ahora podemos hacerlo dos o tres veces al día.

Muy fácil de mantener y probar

los equipos pueden hacer pruebas con el código y dar marcha atrás si algo no funciona como esperan. Esto facilita la actualización del código y acelera el tiempo de salida al mercado de nuevas funciones. Además, es fácil aislar y corregir fallos y errores en los servicios individuales.

Ventajas de los microservicios



Implementación independiente:

los microservicios son unidades individuales, por lo que permiten una implementación independiente rápida y sencilla de funciones individuales.

Flexibilidad tecnológica:

con las arquitecturas de microservicios, los equipos pueden elegir con libertad las herramientas que desean

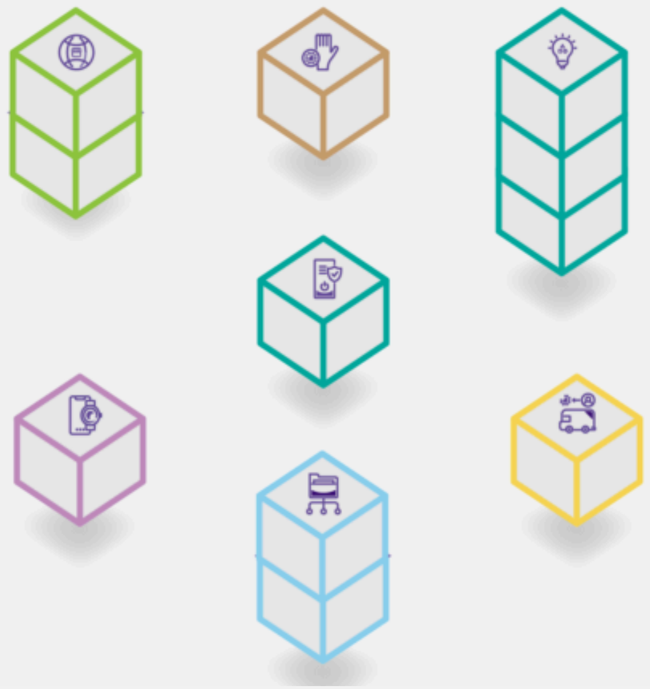
Alta fiabilidad

puedes implementar cambios para un servicio en concreto sin el riesgo de que se caiga toda la aplicación.

Equipos más satisfechos

los equipos de Atlassian que trabajan con microservicios están mucho más satisfechos, ya que son más autónomos y pueden compilar e implementar de forma independiente, sin esperar semanas a que se apruebe una solicitud de incorporación de cambios.

Desventajas de los microservicios



Desarrollo descontrolado

los microservicios suman complejidad en comparación con las arquitecturas monolíticas, ya que hay más servicios en más lugares creados por varios equipos. Si el desarrollo descontrolado no se gestiona adecuadamente, se reduce la velocidad de desarrollo y el rendimiento operativo.

La propiedad no está clara

a medida que se introducen más servicios, también lo hace el número de equipos que los ejecutan. Con el tiempo, se hace difícil conocer los servicios disponibles que un equipo puede aprovechar y con quién hay que hablar para obtener asistencia.

Más sobrecarga organizativa

los equipos deben agregar otro nivel de comunicación y colaboración para coordinar las actualizaciones e interfaces.

Desafíos para la depuración:

cada microservicio tiene su propio conjunto de registros, lo que complica la depuración. Además, un solo proceso empresarial puede ejecutarse en varias máquinas, lo que aún lo dificulta más.

Falta de estandarización

sin una plataforma común, puede haber una proliferación de idiomas, de estándares de registro y de supervisión.

	Monolítica	Microservicios
Diseño	Código base único con múltiples funciones interdependientes.	Componentes independientes con funcionalidad autónoma que se comunican entre sí mediante API.
Desarrollo	Al principio requiere menos planificación, pero se hace cada vez más compleja de entender y mantener.	Al principio requiere más planificación e infraestructura, pero se hace más fácil de administrar y mantener con el tiempo.
Implementacion	Toda la aplicación se implementó como una sola entidad.	Cada microservicio es una entidad de software independiente que requiere una implementación individual en contenedores.
Modificacion	Pequeños cambios introducen mayores riesgos, ya que afectan a todo el código base.	Puede modificar microservicios individuales sin afectar a toda la aplicación.
Escalado	Debe escalar toda la aplicación, incluso si solo ciertas áreas funcionales experimentan un aumento de la demanda.	Puede escalar microservicios independientes según sea necesario, lo que ahorra costos generales de escalado.
Inversion	Baja inversión inicial a costa de aumentar los esfuerzos continuos y de mantenimiento.	Inversión adicional de tiempo y costos para desarrollar la competencia del equipo. Sin embargo, ahorro de costos a largo plazo, mantenimiento y adaptabilidad.