TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN UTN-FRC

TUP

UTN Facultad Regional Córdoba

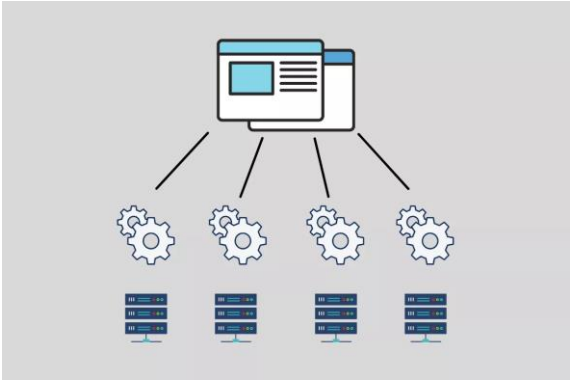
LABORATORIO DE COMPUTACIÓN III

2024

Microservicios

ESTRUCTURA TEMÁTICA

- ¿Qué son los microservicios?
- Monolitos vs Microservicios
- Ventajas/Desventajas de microservicios
- Tolerancia a fallos y gestión de errores
- Patrones y Buenas Prácticas en microservicios

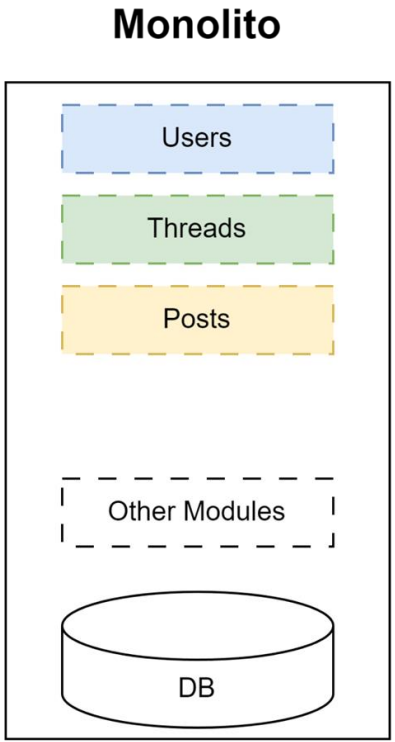


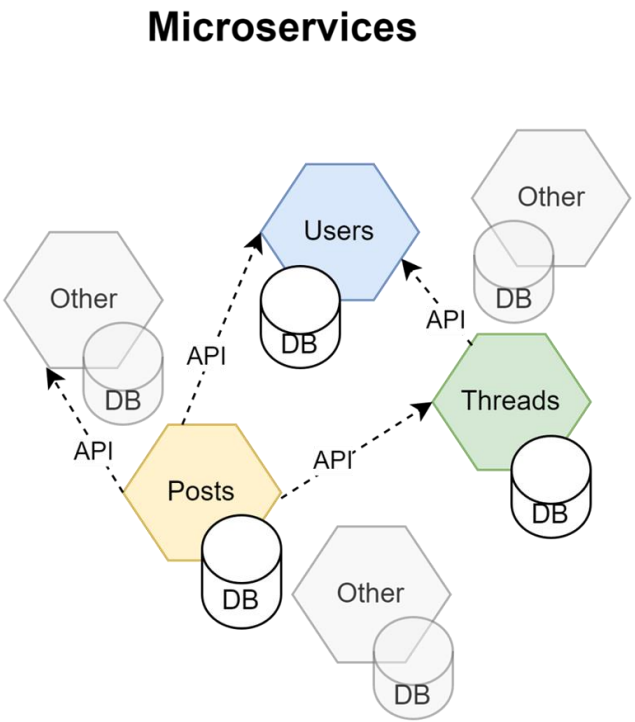
¿Qué son los microservicios?

Un Microservicios representa **una funcionalidad específica** y bien definida dentro de la aplicación general, lo que permite un enfoque modular y un desacoplamiento entre las distintas partes del sistema. Cada microservicio es responsable de una tarea o una función del negocio y **opera de manera independiente**.

¿Qué son los microservicios?

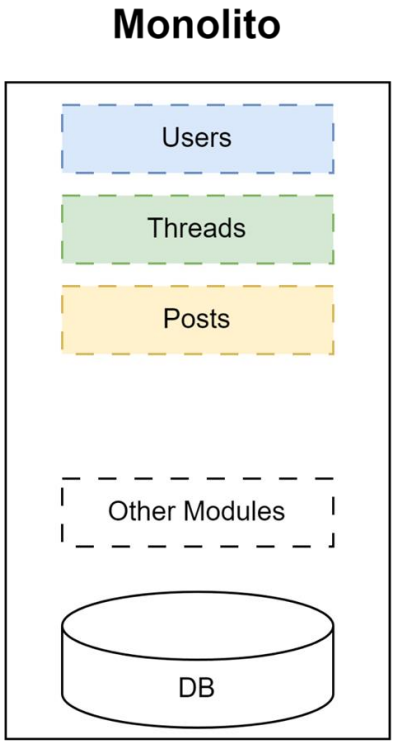
La principal premisa detrás de los microservicios es la división del sistema en componentes más manejables y especializados, en lugar de tener un gran monolito donde todas las funcionalidades están fuertemente acopladas.

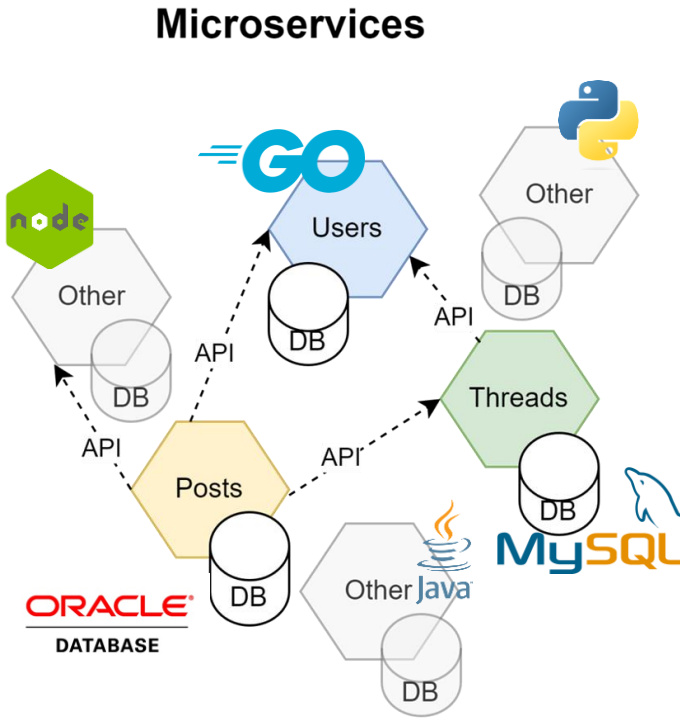




¿Qué son los microservicios?

Los microservicios permiten que los equipos utilicen diferentes tecnologías y lenguajes de programación para desarrollar cada servicio según las necesidades específicas. Esto proporciona una gran flexibilidad y evita que un equipo quede limitado a una única tecnología para toda la aplicación.





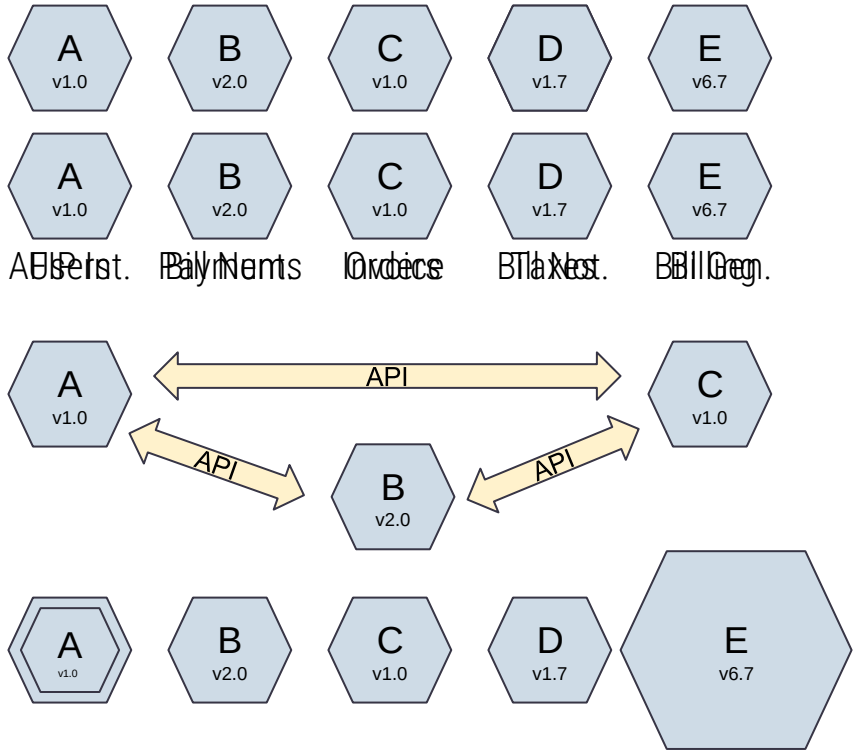
Autónomos

Despliegue independiente

Lógica de negocio específica

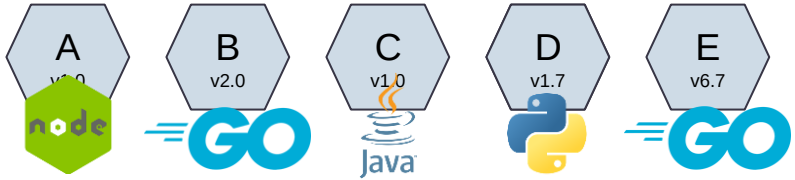
Comunicación a través de interfaces

Escalabilidad independiente

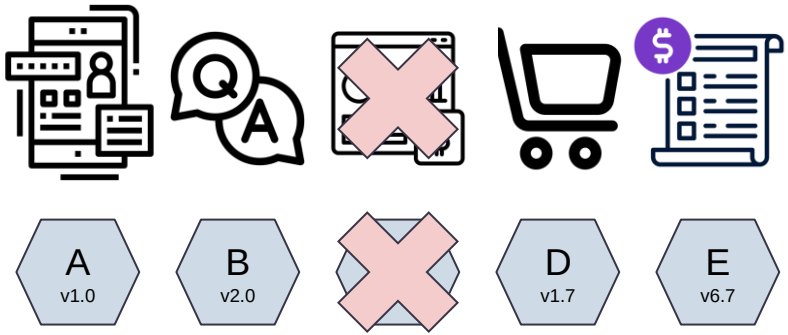


Autónomos

Tecnología y plataforma independiente

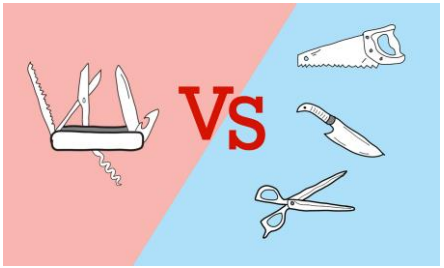


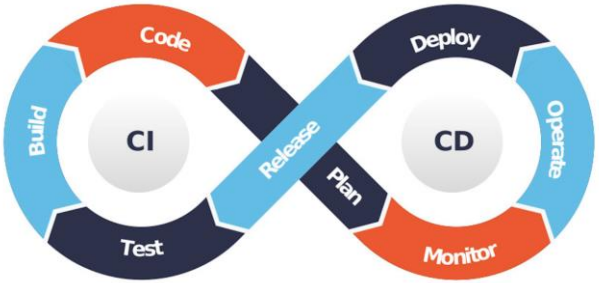
Aislamiento de fallos



Especializados

Enfoque en una sola responsabilidad



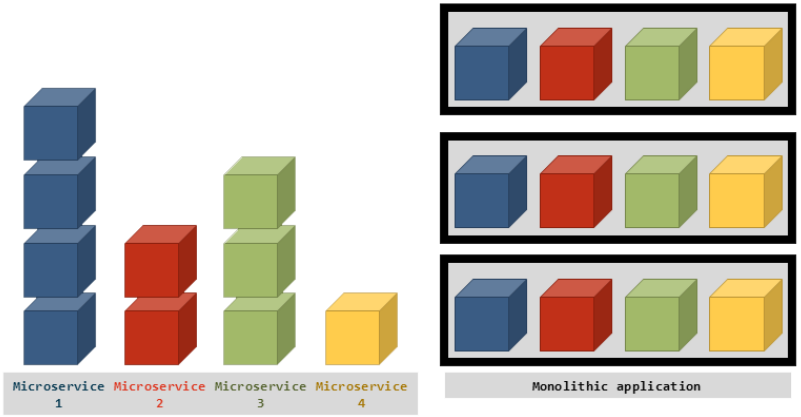


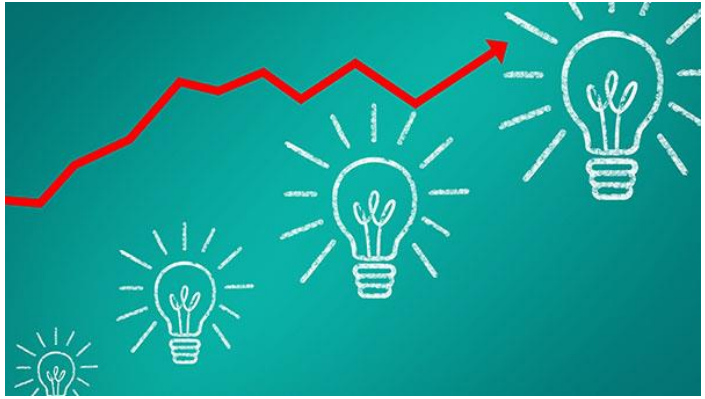
Enfoque en una sola responsabilidad

Acoplamiento reducido

Especializados

Escalabilidad dirigida





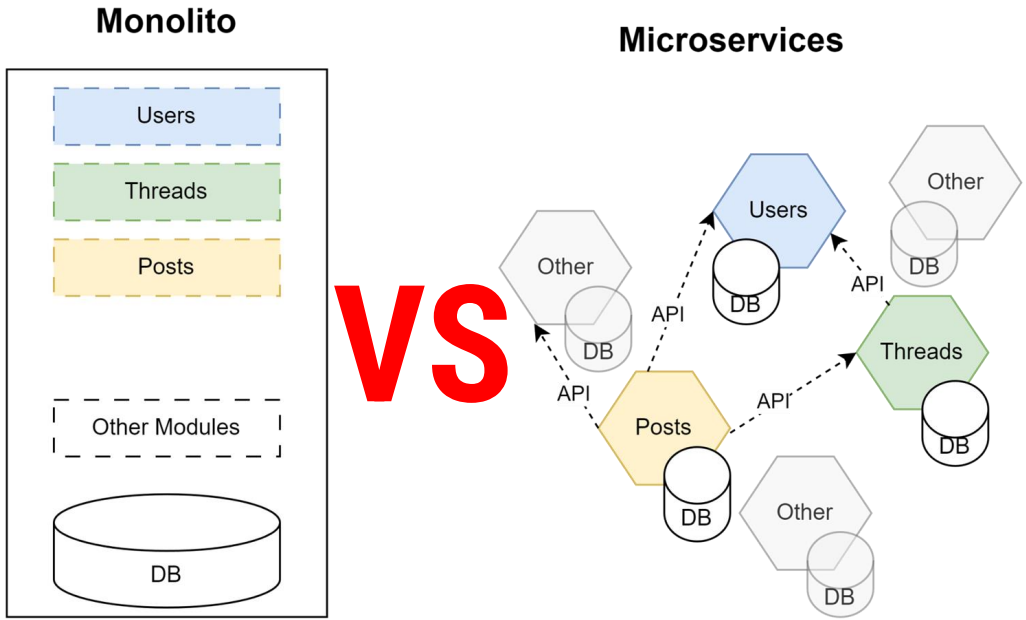
Mejor adaptación a la evolución del negocio

Monolitos vs Microservicios

Los procesos están estrechamente asociados y se ejecutan como un solo servicio.

Agregar o mejorar las características se vuelve más complejo a medida que crece la base de código.

Aumentan el riesgo de indisponibilidad de la aplicación.



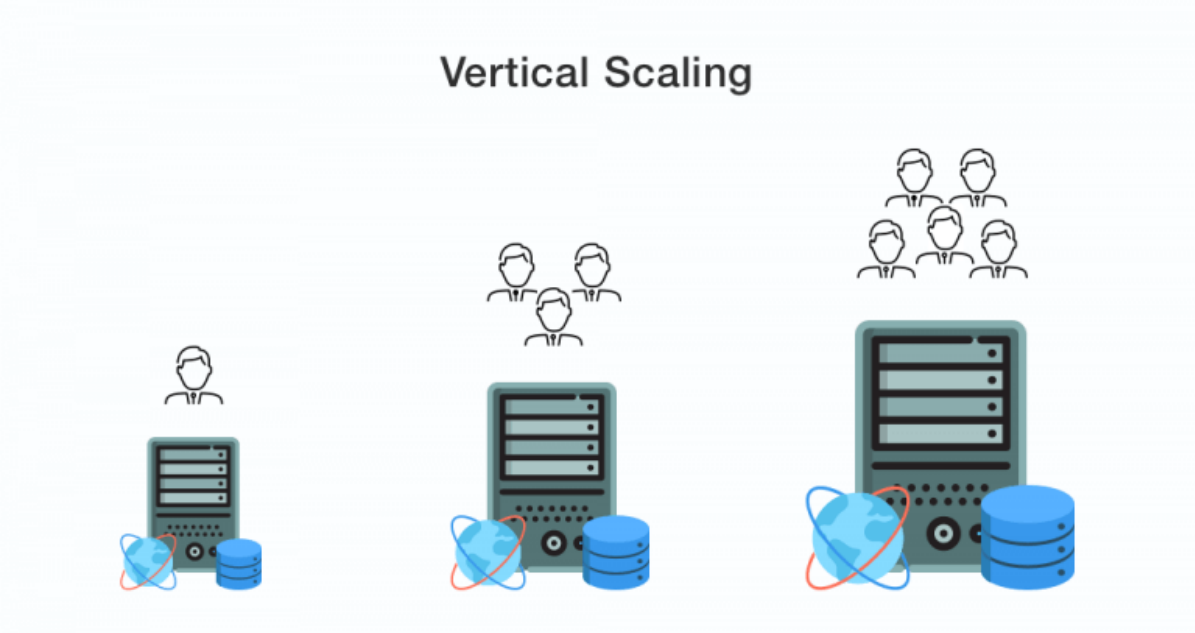
Una aplicación se crea con componentes independientes.

Se comunican a través de una interfaz bien definida mediante API ligeras.

Cada servicio se puede actualizar, implementar y escalar de manera independiente.

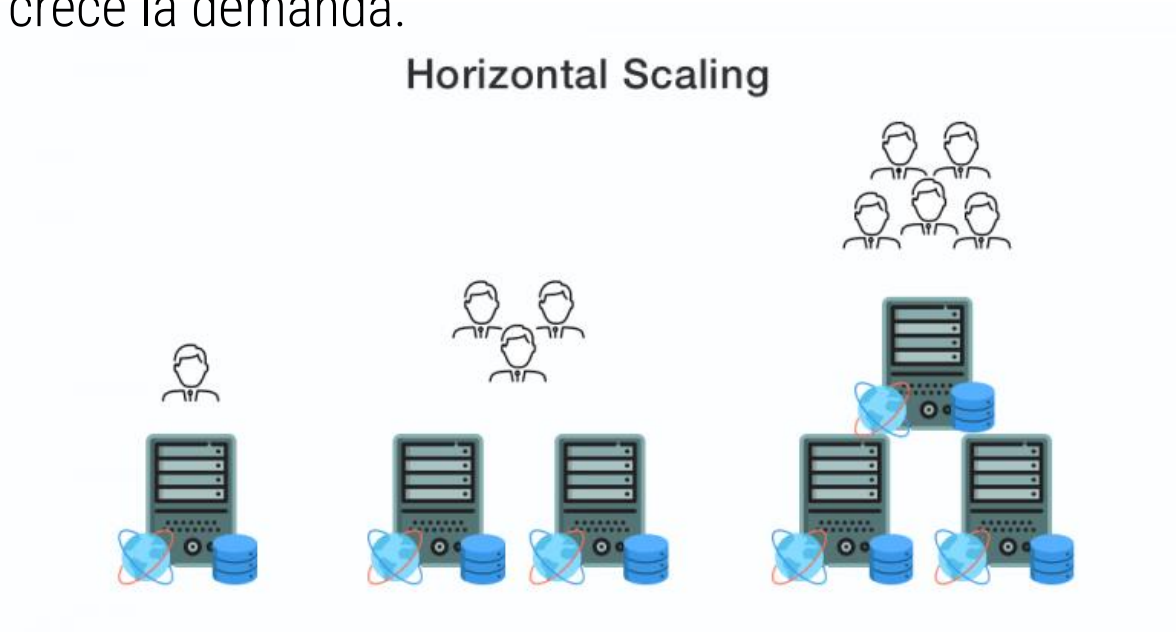
Escalabilidad Vertical

Consiste en aumentar los recursos (como CPU, memoria RAM o almacenamiento) en un solo servidor o máquina para mejorar su rendimiento y capacidad



Escalabilidad Horizontal

Se aumenta la capacidad del sistema agregando más instancias o servidores idénticos para distribuir la carga de trabajo. Permite agregar recursos adicionales a medida que crece la demanda.



Ventajas y Desventajas

Ventajas	Desventajas
Desarrollo ágil y rápido despliegue	Complejidad en la gestión y coordinación
Escalabilidad y flexibilidad	Overhead de comunicación
Mantenibilidad simplificada	Mayor esfuerzo de pruebas y monitoreo
Independencia tecnológica	Posible degradación del rendimiento
Mayor facilidad para equipos distribuidos	Consistencia de datos
Resistencia a fallos	Requerimientos de infraestructura adicionales

Tolerancia a fallos y gestión de errores

La tolerancia a fallos y la gestión de errores son aspectos críticos en el diseño y desarrollo de sistemas, especialmente en entornos distribuidos como las arquitecturas de microservicios. Estos conceptos se refieren a cómo un sistema maneja situaciones inesperadas o problemas en su funcionamiento para asegurar la continuidad del servicio y la recuperación ante fallos.

Tolerancia a fallos

Es la capacidad de un sistema para continuar funcionando de manera adecuada y proporcionar una funcionalidad degradada o una recuperación automatizada en caso de que uno o varios de sus componentes (como microservicios) experimenten fallos o errores.

Redundancia y replicación: Mantener copias redundantes de los microservicios y sus datos en diferentes nodos o servidores, de modo que si uno falla, otros puedan tomar su lugar.

Circuit Breaker: Implementar el patrón de circuit breaker para detectar errores en los microservicios y evitar solicitudes adicionales hasta que el servicio se recupere.

Respuestas degradadas: En lugar de fallar por completo, los microservicios pueden proporcionar respuestas degradadas.

Gestión de colas y reintentos: Si un microservicio no responde temporalmente, se puede implementar una gestión de colas y reintentos para volver a intentar la solicitud en un momento posterior.

Manejo adecuado de excepciones: Los microservicios deben capturar y manejar adecuadamente las excepciones para evitar que los errores se propaguen sin control.

Gestión de errores

Se refiere a la forma en que el sistema maneja los errores y las situaciones excepcionales cuando ocurren. Es fundamental implementar una gestión de errores robusta para proporcionar información significativa sobre los fallos y facilitar la resolución de problemas.

Registro y monitoreo de errores: Registrar y monitorear los errores en los microservicios para identificar patrones y tendencias de fallos y facilitar su resolución.

Respuestas con códigos de estado adecuados: Devolver códigos de estado HTTP o códigos de error significativos para que los clientes puedan entender la naturaleza del problema y responder en consecuencia.

Notificación y alertas: Configurar sistemas de notificación y alertas para que el equipo de desarrollo sea informado inmediatamente cuando ocurra un error crítico.

Reintentos y recuperación: Implementar mecanismos de reintentos y recuperación para intentar nuevamente la operación o proporcionar alternativas en caso de errores temporales.

Respuestas amigables para el usuario: Proporcionar mensajes de error claros y comprensibles para los usuarios finales, evitando mensajes técnicos o poco informativos.

TUP UTN Facultad Regional Córdoba

Gracias