UNIVERSIDAD AUTONOMA DE CHIAPAS

CONTADURÍA Y ADMINISTRACIÓN CAMPUS I

LICENCIATURA EN INGENIERIA EN DESARROLLO Y TECNOLOGIAS DE SOFTWARE

LUIS EDUARDO GONZALEZ GUILLEN – 6M – A211397

ACT.1.2 ARQUITECTURA ORIENTADA A SERVICIOS

TALLER 4

MTRO LUIS ALFARO GUTIERREZ

19 DE AGOSTO DEL 2023

ARQUITECTURA DE MICROSERVICIOS:

Características:

La arquitectura de microservicios es un enfoque de diseño de software que descompone una aplicación en una colección de servicios independientes y desacoplados, que se ejecutan en su propio proceso y se comunican a través de mecanismos como HTTP, mensajes o API (Newman, 2015). A continuación se describen algunas de sus características clave:

- Desacoplamiento: Los microservicios son entidades independientes que pueden funcionar y evolucionar de forma autónoma (Dragoni et al., 2017).
- Escalabilidad: Cada microservicio puede escalarse de forma independiente, lo que permite una mayor flexibilidad y eficiencia en el uso de recursos (Villamizar et al., 2016).
- Resiliencia: La falla en un microservicio no necesariamente afecta el funcionamiento de toda la aplicación, lo que mejora la disponibilidad y la tolerancia a fallos (Buyya & Srirama, 2019).
- Despliegue Independiente: Los microservicios pueden desplegarse, actualizarse y escalar de forma independiente, lo que facilita la implementación de nuevas características y correcciones (Richardson, 2018).
- Descentralización: La arquitectura de microservicios favorece un enfoque descentralizado en el desarrollo y la gestión, lo que permite a los equipos trabajar de forma más autónoma (Lewis & Fowler, 2014).
- Alta Cohesión y Bajo Acoplamiento: Los microservicios están diseñados para ser altamente cohesivos y con bajo acoplamiento, lo que facilita su mantenimiento y evolución (Fraser et al., 2017).

Beneficios:

- Agilidad en el Desarrollo: La modularidad de los microservicios permite a los equipos de desarrollo trabajar de forma más ágil y rápida, facilitando la implementación de nuevas funcionalidades (Newman, 2015).
- Escalabilidad: La capacidad de escalar servicios de manera independiente permite un uso más eficiente de los recursos y mejora el rendimiento de la aplicación (Villamizar et al., 2016).
- Resiliencia y Tolerancia a Fallos: La independencia entre microservicios mejora la resiliencia del sistema, ya que una falla en un servicio no necesariamente afecta a los demás (Buyya & Srirama, 2019).
- Despliegue y Mantenimiento Simplificados: Los microservicios pueden desplegarse de forma independiente, lo que facilita las actualizaciones y el mantenimiento del sistema (Richardson, 2018).

Técnicas:

- API RESTful: Una de las técnicas más comunes para la integración de microservicios es el uso de API RESTful, que permite la comunicación a través de HTTP(S) (Richardson, 2018).

- Mensajería Asíncrona: Utilizando colas de mensajes o buses de eventos, los microservicios pueden comunicarse de manera asíncrona, lo que mejora la resiliencia y la escalabilidad (Newman, 2015).
- GraphQL: Ofrece una alternativa a REST para la consulta de datos entre microservicios, permitiendo solicitudes más flexibles y optimizadas (Schrock, 2017).
- gRPC: Este protocolo de alto rendimiento permite la comunicación entre microservicios utilizando HTTP/2 y Protocol Buffers, y es especialmente útil para microservicios que requieren baja latencia (Buyya & Srirama, 2019).

Balanceo de carga:

El balanceo de carga es una técnica utilizada para distribuir el tráfico entrante de una red o sistema entre múltiples servidores, enlaces, núcleos de CPU, discos u otros recursos. El objetivo es optimizar la utilización de recursos, maximizar el rendimiento, minimizar los tiempos de respuesta y evitar la sobrecarga de cualquier recurso individual (Kaur & Luthra, 2014).

Despliegue:

El despliegue, también conocido como implementación, es el proceso mediante el cual una aplicación o sistema de software se instala, configura y pone en funcionamiento en un entorno específico para su uso y operación (Humble & Farley, 2010). Este proceso puede implicar diversas etapas, desde la preparación del entorno hasta la verificación post-despliegue para asegurar que el sistema funcione como se espera.

Arquitectura monolítica vs arquitectura de microservicios

MONOLITICA	MICROSERVICIOS
Estructura Unificada: En una arquitectura monolítica, todas las funcionalidades de la aplicación se desarrollan como una unidad indivisible (Parnas, 1972).	Desacoplamiento: Los microservicios son entidades independientes que pueden funcionar y evolucionar de forma autónoma (Dragoni et al., 2017).
Despliegue: Todo el código se despliega como una única base de código, lo que puede hacer que el proceso de despliegue sea más sencillo pero menos flexible (Humble & Farley, 2010).	Despliegue: Cada microservicio se despliega de forma independiente, lo que facilita la implementación de nuevas características y correcciones (Richardson, 2018).
Escalabilidad: La escalabilidad se logra replicando la aplicación monolítica completa, lo que puede ser ineficiente (Vogels, 2008).	Escalabilidad: Cada microservicio puede escalarse de forma independiente, lo que permite una mayor flexibilidad y eficiencia en el uso de recursos (Villamizar et al., 2016).
Acoplamiento: Las diferentes partes de la aplicación están estrechamente acopladas, lo que puede complicar el mantenimiento	Acoplamiento: Los microservicios están diseñados para ser altamente cohesivos y con bajo acoplamiento, lo que facilita su

y la introducción de cambios (Parnas, 1972).	mantenimiento y evolución (Fraser et al., 2017).

REFERENCIAS BIBLIOGRAFICAS:

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: yesterday, today, and tomorrow. Present and Ulterior Software Engineering, 195–216.

Pahl, C., & Jamshidi, P. (2016). Microservices: A Systematic Mapping Study. 6th International Conference on Cloud Computing and Services Science, 137–146.