# Definición

Los patrones de diseño son planos útiles para construir aplicaciones en la nube confiables, escalables y seguras. El patrón de diseño Circuit Breaker es utilizado en el desarrollo de software para mejorar la posible resiliencia y tolerancia a fallo de un sistema [1]. La regla principal seguida por este patrón es evitar que una aplicación intente repetidamente ejecutar una operación que es probable que falle, permitir que continúe sin esperar a que se solucione el fallo y detectar si el fallo se ha resuelto [2].

En este contexto Circuit Breaker se puede implementar de dos maneras: el patrón Circuit Breaker del lado del cliente y del lado del servicio.

* En el lado del cliente, cada cliente incluye un Circuit Breaker separado para interceptar llamadas a servicios externos. Esto permite al cliente monitorear el estado del servicio externo y decidir si realizar la llamada o no [3].
* En el lado del servicio implica el uso de un Circuit Breaker interno en el servicio para procesar las invocaciones del cliente. Este diseño interno determina si la invocación debe procesarse o no en función del estado del servicio [3].

Es importante indicar que este patrón no sigue la misma lógica que el patrón Retry. Retry simplemente reintenta el proceso anteriormente fallado esperando una respuesta positiva, mientras que Circuit Breaker evita que una aplicación realice una operación que es probable que falle [2].

# Caso de Implementación

implementación: Citio de E-Commerce

Un nuevo sitio de e-commerce da uso de un servicio externo para poder realizar sus pagos con la mayor cantidad de usuarios como sea posible. El único problema que han tenido últimamente es una continua cantidad de quejas relacionadas en momentos donde el servicio externo de pasarela de pagos no estaba funcional.

Aparentemente, este servicio tiene una alta cantidad de tiempos de mantenimiento. Por esto se implementa un sistema Circuit Breaker por parte del sitio de e-commerce, evitando que los usuarios intenten repetidamente realizar pagos si la pasarela de pago no está disponible, así ayudando a evitar que el sistema se sobrecargue y se bloquee.

# Bibliografía

[1] Montesi, F., & Weber, J. (2016). Circuit breakers, discovery, and API gateways in microservices. arXiv preprint arXiv:1609.05830.

[2] Martinekuan. (n.d.). Circuit breaker pattern - azure architecture center. Azure Architecture Center | Microsoft Learn. <https://learn.microsoft.com/en-us/azure/architecture/patterns/circuit-breaker>

[3] Saleh Sedghpour, M. R., Klein, C., & Tordsson, J. (2022, April). An Empirical Study of Service Mesh Traffic Management Policies for Microservices. In *Proceedings of the 2022 ACM/SPEC on International Conference on Performance Engineering* (pp. 17-27).