

Напоследок, рассмотрим паттерн повышения надёжности вашей системы Circuit Breaker (Автоматический выключатель)

Что это за паттерн?

Представьте электрическую цепь в вашем доме. Если что-то идет не так, например, возникает перегрузка, автоматический выключатель "срабатывает", прерывая цепь и предотвращая возможный пожар.

В программировании Circuit Breaker работает по той же логике: если внешний сервис, на который мы полагаемся, начинает глючить, Circuit Breaker "срабатывает", прерывая вызовы к этому сервису, чтобы избежать дальнейших проблем.

Как это реализуется в монолите?

В монолитной архитектуре такой паттерн может быть реализован в виде специальных условий и таймеров в коде, которые следят за временем отклика и успешностью вызовов к внешним ресурсам, таким как база данных или другие внешние API.

Зачем этот паттерн в микросервисах?

В микросервисной архитектуре этот паттерн особенно полезен, потому что микросервисы часто зависят друг от друга. Если один из них начинает работать некорректно, это может привести к "снежному кому" проблем во всей системе. Circuit Breaker помогает избежать этого, "отключая" проблемный микросервис до его восстановления.

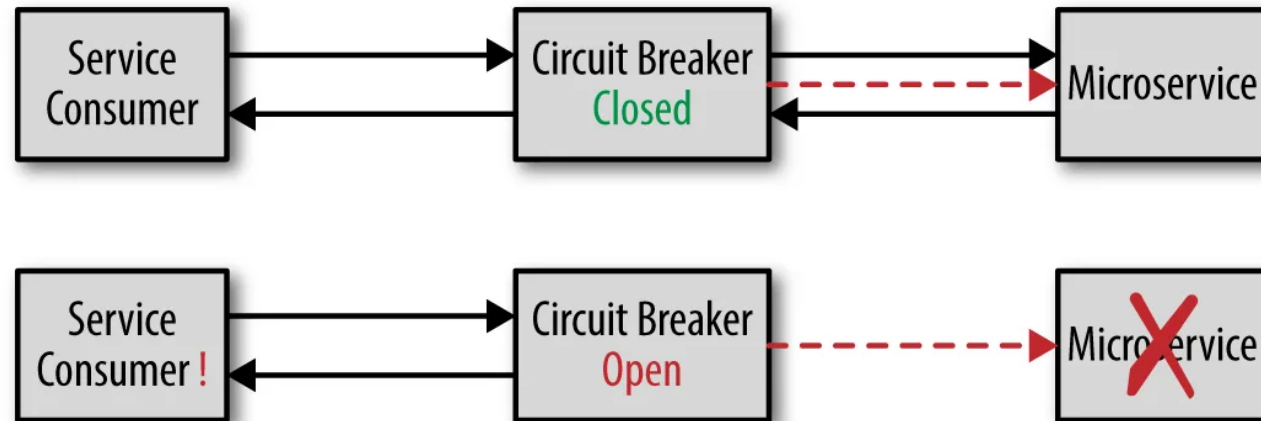
Как это реализуется?

Представим, что у нас есть онлайн-магазин с микросервисом для обработки платежей.

Когда пользователь нажимает "Купить", микросервис обращается к внешнему сервису для проведения платежа. Circuit Breaker (как дополнительный сервис-прокси) следит за этим процессом.

- 1. Закрытое состояние (Closed):** Все идет хорошо, платежи проходят. Circuit Breaker "закрыт", и вызовы к платежному сервису проходят свободно.
- 2. Открытое состояние (Open):** Если происходит сбой и несколько платежей подряд не удастся провести, Circuit Breaker "открывается". Теперь, если пользователь пытается сделать покупку, Circuit Breaker не дает этому вызову пройти, и пользователь видит сообщение об ошибке "Сервис временно недоступен".

3. **Полуоткрытое состояние (Half-Open):** После некоторого времени Circuit Breaker переходит в "полуоткрытое" состояние и пропускает один или несколько "тестовых" вызовов к платежному сервису, чтобы проверить, восстановился ли он.



Нюансы при использовании паттерна:

1. **Настройка лимтов:** Вам нужно аккуратно выбрать, после какого количества неудачных попыток или задержек Circuit Breaker должен "сработать".
2. **Время восстановления:** Как долго Circuit Breaker должен оставаться в "открытом" состоянии, прежде чем перейти в "полуоткрытое".
3. **Обратная связь:** Нужно определить, как система и пользователи будут информированы о срабатывании Circuit Breaker.
4. **Fallback-стратегии:** Что делать, когда Circuit Breaker срабатывает? Например, можно показать пользователю сообщение об ошибке или использовать кэшированные данные.