# Report on Circuit Breaker Pattern

Alonso Gago Suárez - UO269424
Jairo García Castro - UO271449
Diego Cabas Álvarez - UO271506

**Report Index**



# Introduction

The main goal of this pattern is to act like a real electric circuit breaker, enabling and disabling a request into a system once we know that it is gonna fail or that it has already failed several times. Allowing the service to stop consuming resources for an useless petition, encapsulating its logic so it can't make more requests, as we already know that it's going to fail.

## Aspects to take into account during implementation

Three main aspect need to be taken into consideration before the implementation of this pattern:

- Will it really increase system capability?
- Is it possible to be deployed in our system?
- Are you able to afford its maintenance cost?

# Constraints during the implementation

Circuit Breaker Pattern is a really useful healing software that could prevent some overloading scenarios in systems such as Thundering or OutOfMemory errors. But in order to achieve this we need to correctly adapt this pattern to the requirements of our system or we won't optimize all of its benefits:

- Adapt threshold base on system requirement
- Adapt maximum time for circuit to be open
- Configure exceptional triggers to close circuits
- Configure constraint per environment, endpoint and request

It's essential to keep updating proactively all this threshold as our system keeps evolving, otherwise the old threshold could significantly reduce the outcome of our system.

## Consequences of wrongly configured implementation

As stated before, is essential in this pattern to correct configure and adapt the threshold of this pattern to our system, otherwise some of these problems could possible arise:

- System's outcome rate of services is decreased
- Excess resources consumption if a circuit keep open for too much time
- Appearance of errors due to Circuit breaker thread management
- Confusion among downstreams services if episodic requests are produced in our patter.

# Scenario

Given the following scenario of some developed application we can normally find the unresponsive API call problems:

- You develop a system for your company that allows users to check data from your customers
- You use an API to list customers and display their details when selected
- The system providing the API your system relies on goes down
- Calls to the API timeout after 30 seconds and the system is unresponsive

The solution we can provide to it thanks to circuit breaker:

- A system through which all API calls go through that continually monitors for failures.
- In case of a timeout failure, the circuit breaker moves from closed to open state, and all further calls to the API don't reach the external system.
- When the service exceeds a failure threshold that we set, the circuit breaker enters the "Open" state

## Scenario Solution Overview

This is the overview of a possible solution that can be applied in our scenario regarding circuit breaker architecture:

- Now we don't send requests to the API that fails, our servers don't overload and we have time to figure a solution
- After a time we enter the "Half Open" state which will determine if we stay open or closed.
- This way the infrastructure is safe from being stalled waiting for a system that is down and so your resources are not being pointlessly consumed