CIRCUIT BREAKER PATTERN

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

What is it?

Its goal is to behave as an actual electric circuit breaker

How does it work?

```
// Client code

breaker.call();
```

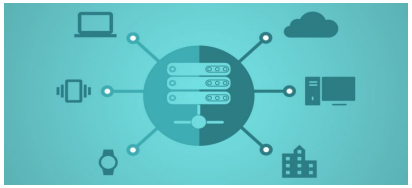
```
private void call(){
 if(closedState){

   doCall();

   if(fail){
      failures++;
   }
   if(failures >= failures_threshold){
      setState(openState)
   }
}
```

```
else if(openState){
   timeout = now();
   if(timeout >= time_threshold){
   setState(halfOpenState);
   }
   throw Error;
} else if(halfOpenState)
      doCall();
      if(fail){
         setState(openState);
      }
      else{
         setState(closedState);
      }
}
```

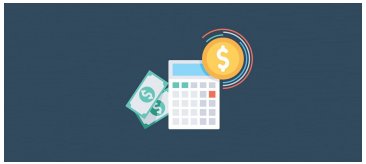
Aspects before implementing a short circuit

1. Will it truly increase your system capability?

2. Is it possible to deploy it in current system?

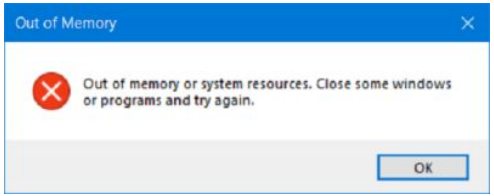


3. Are you able to afford its maintenance cost?



Constraints during implementation

- Useful self healing software, preventing the overload of systems (Thundering herds, OutOfMemory error…)



- Important considerations during implementation:
- Adapt correct threshold based on system requirements.
- Adapt maximum time for circuit to be open
- Configure excepcional triggers to close circuits
- Configure each constraint: per environment/endpoint/request.

Essential maintaining proactively these configurations as system evolves.

Consequences of a wrongly configured implementation

- Possibles problems of a poorly adapted circuit breaker:
  - System’s rate to provide services is decreased.
  - Excess resources consumption if circuit open time is too elongated.
  - Appearance of differents errors due to Circuit breaker thread management.
  - Confusion among downstream service owners due to episodic request patterns.

Scenario

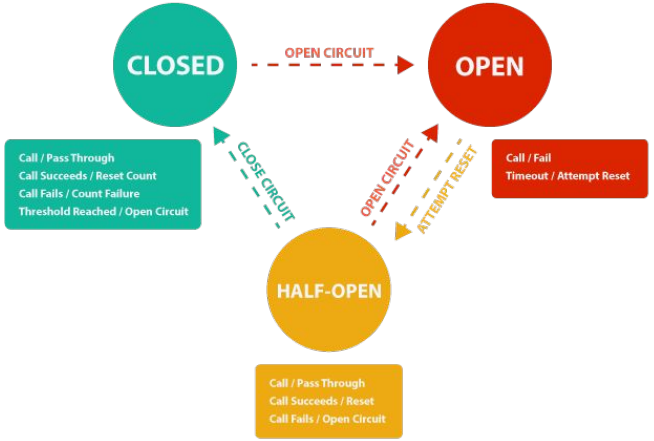
- You develop a system for your company that allows users to check data from your customers
- You use an API to list customers and display their details when selected
- The system providing the API your system relies on goes down
- Calls to the API timeout after 30 seconds and the system is unresponsive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

Solution: Circuit breaker pattern

- 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

General overview of the solution

- 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 in 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



QUESTIONS TIME