Kresil

Kotlin Resilience

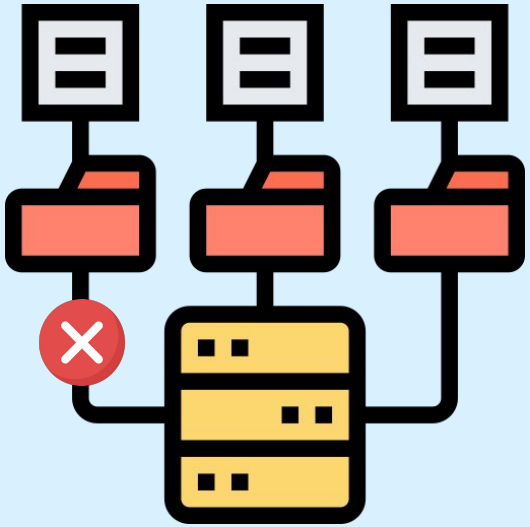
Kotlin Multiplatform library for fault-tolerance with Ktor Integration

ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Supervisor: Prof. Pedro Félix
Author: Francisco Engenheiro - 49428
Project and Seminary
BSc in Computer Science and Engineering
Summer 2023/2024

Resilience in Distributed Systems

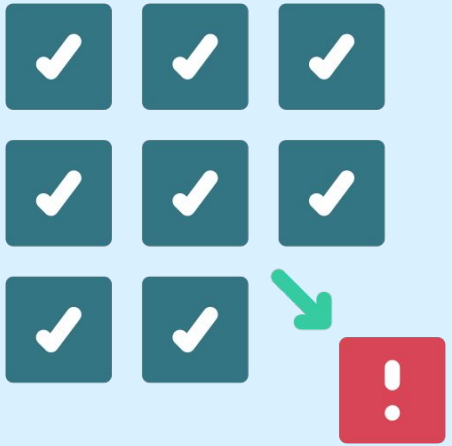
If failure is inevitable, what can be done?



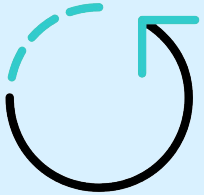
Fault-Tolerance Service

In the presence of faults:

- **Maintains all or part of its functionality**
- **Provides an alternative**

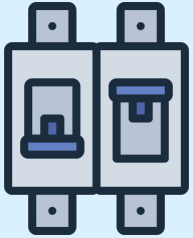


Resilience Mechanisms



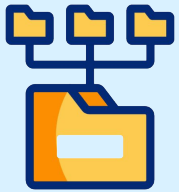
Retry - Repeats failed executions

Rate Limiter - Limits executions/period

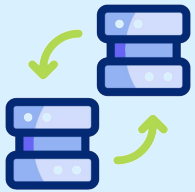


Circuit Breaker - Temporarily blocks possible failures

Time Limiter - Limits duration of execution



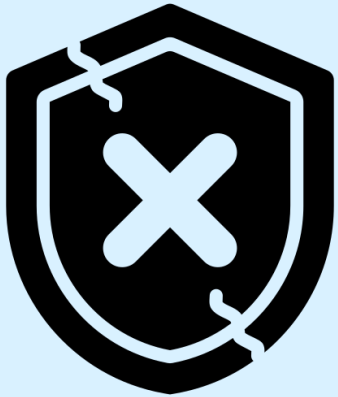
Cache - Memorizes a successful result



Fallback - Defines an action to fallback on failure

And many more...

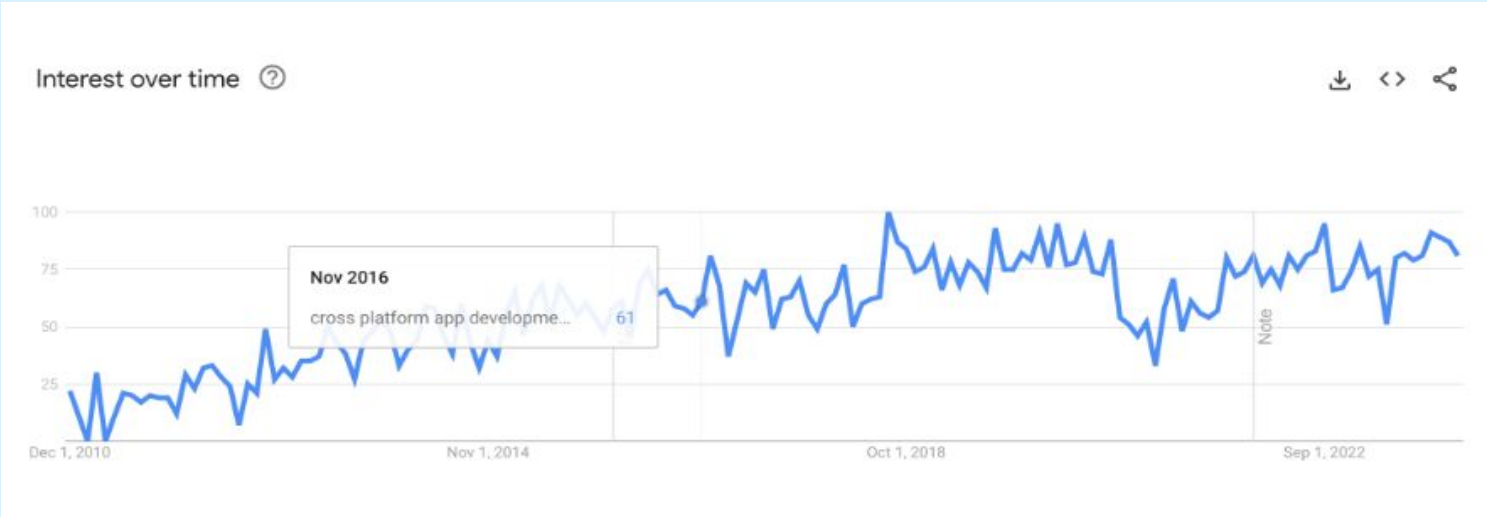
Resilience Types



Reactive - Mitigates impact from failures

Proactive - Prevents failures from happening

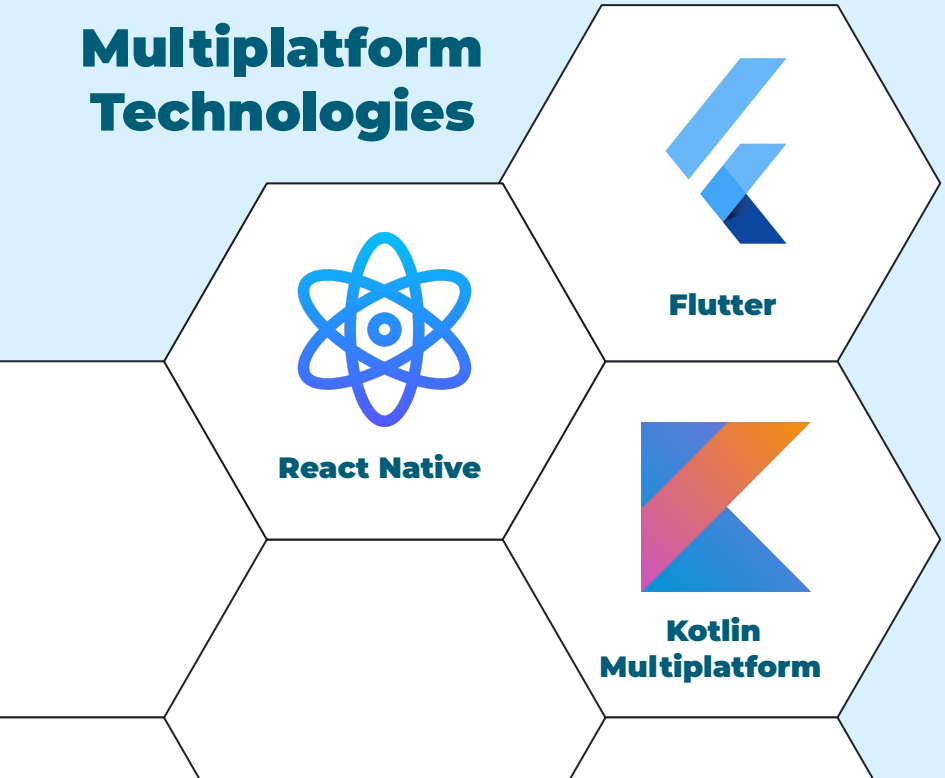
Multiplatform Development Interest



Font:
https://www.jetbrains.com/help/kotlin-multiplatform-dev/cross-platform-frameworks.html

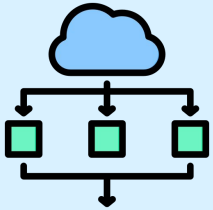
Problem

Multiplatform Technologies



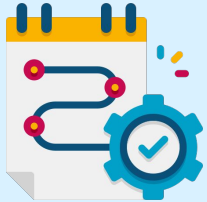
No libraries that provide resilience mechanisms in a multiplatform context

Multiplatform Considerations

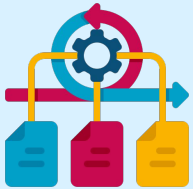


Concurrency Model

Time Management



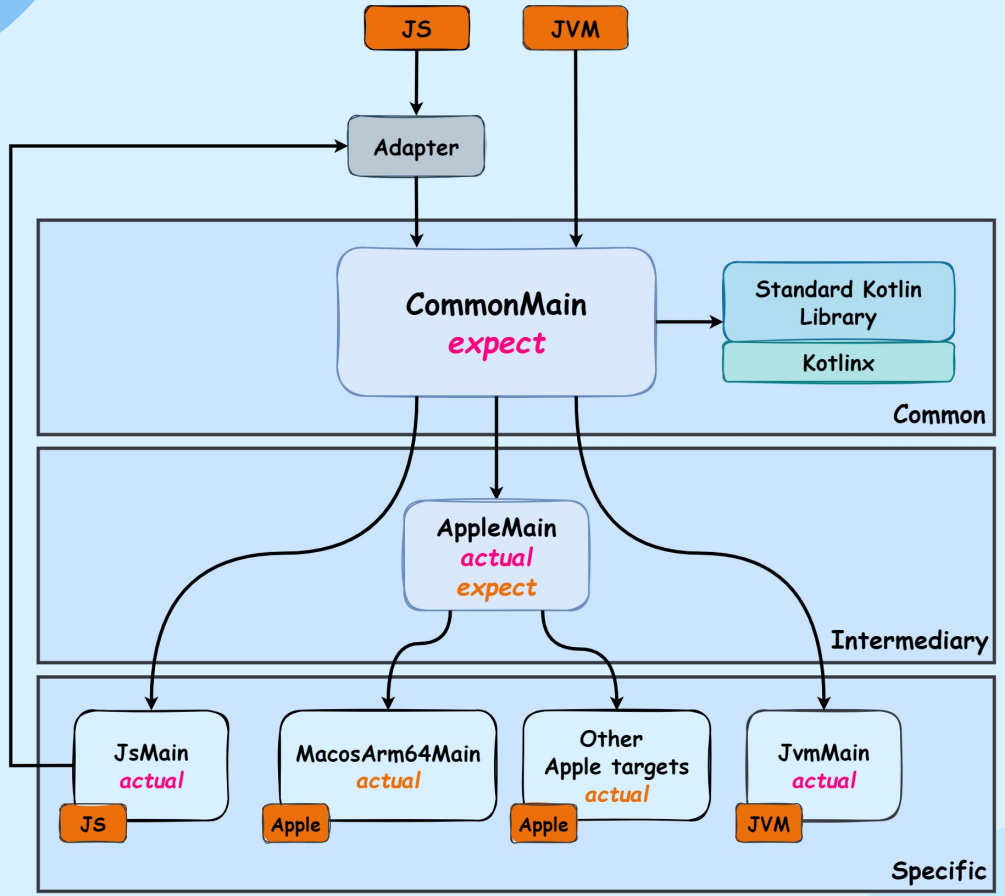
Synchronous vs Asynchronous



Mocking

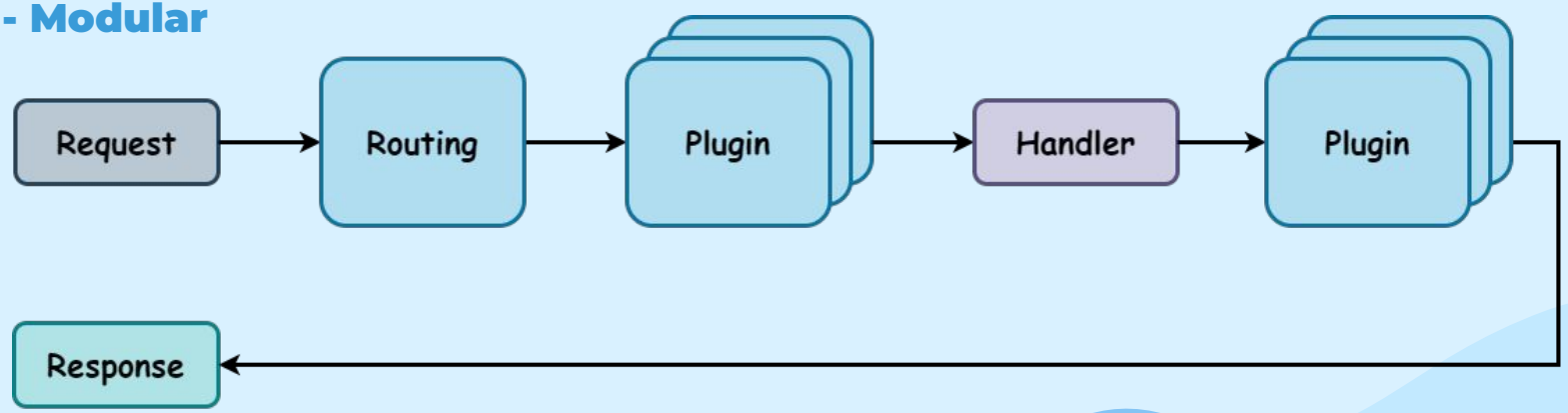
Logging

Kotlin Multiplatform Architecture



Ktor Framework

- Built with Kotlin Multiplatform;
- Only known asynchronous server and client development framework
- Based on the coroutines concurrency model;
- Modular



Existing Solutions

Platform-Specific

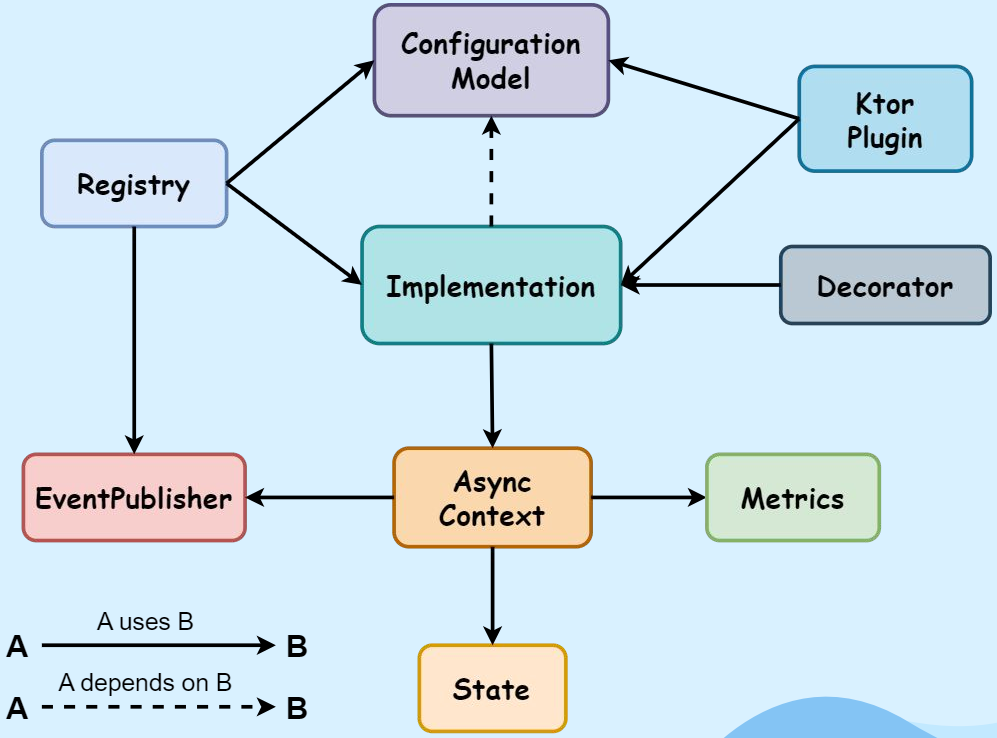
Resilience4j

Netflix's Hystrix

Multiplatform

Arrow

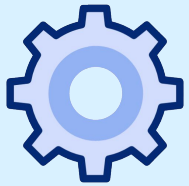
Mechanism Model



Mechanism Configuration

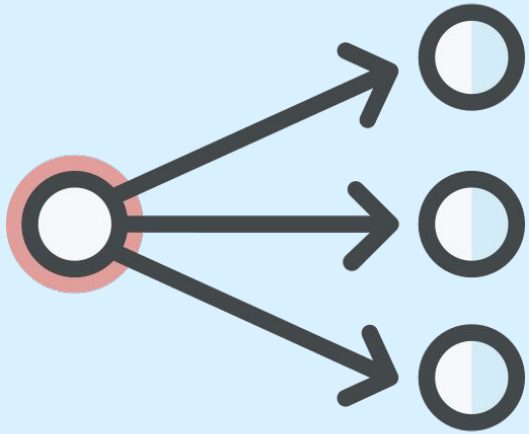
```
internal fun <TBuilder: ConfigBuilder<TConfig>, TConfig> mechanismConfigBuilder(
    builder: TBuilder,
    configure: TBuilder.() -> Unit
): TConfig = builder.apply(configure).build()
```

Policies - Define the mechanism behaviour



```
val config: RetryConfig = retryConfig {
    maxAttempts = 3
    retryIf { it is WebServiceException }
    constantDelay(3.seconds)
}
val retry = Retry(config)
// or: val retry = Retry() // uses default config
```

Event Listeners

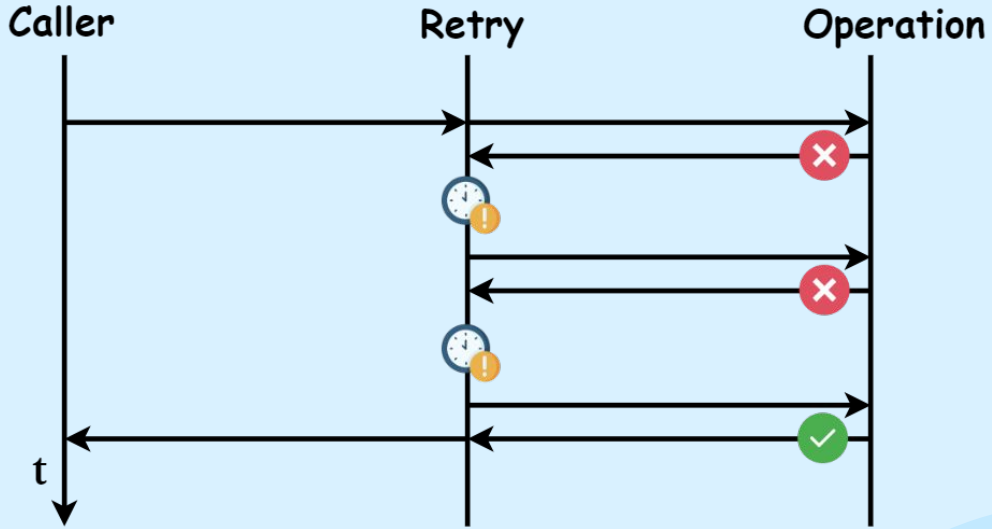


Each mechanism:

- **Provides listeners for specific and undiscriminated events**
- **Uses asynchronous coroutine primitive Flow**
- **Supports cancellation of registered listeners**

Retry Mechanism

- **Reactive mechanism**
- **Retries operation on failure**
- **Addresses transient faults**



Delay Handling

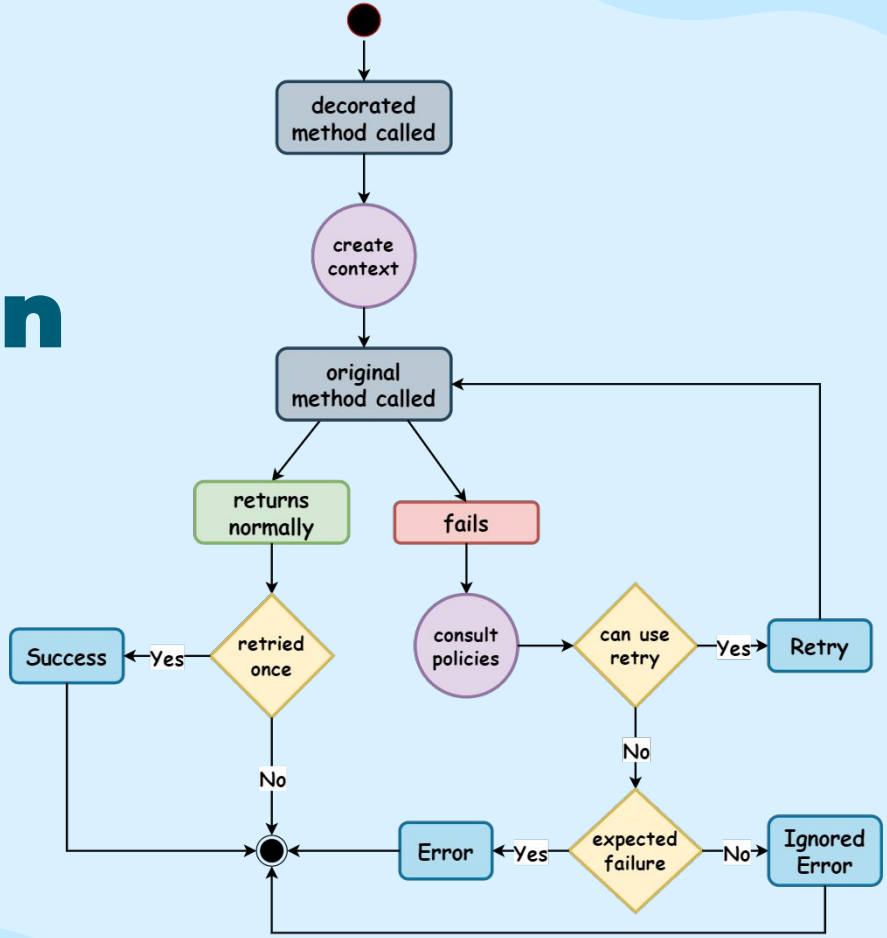
Delay Strategy

Options

- No delay
- Constant delay
- Linear delay
- Exponential delay
- Custom Delay

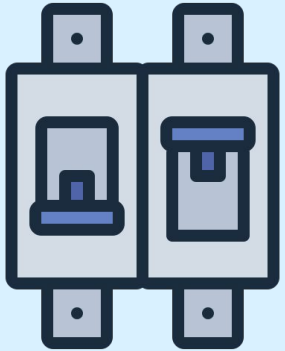
Delay Provider

Retry Execution Flow

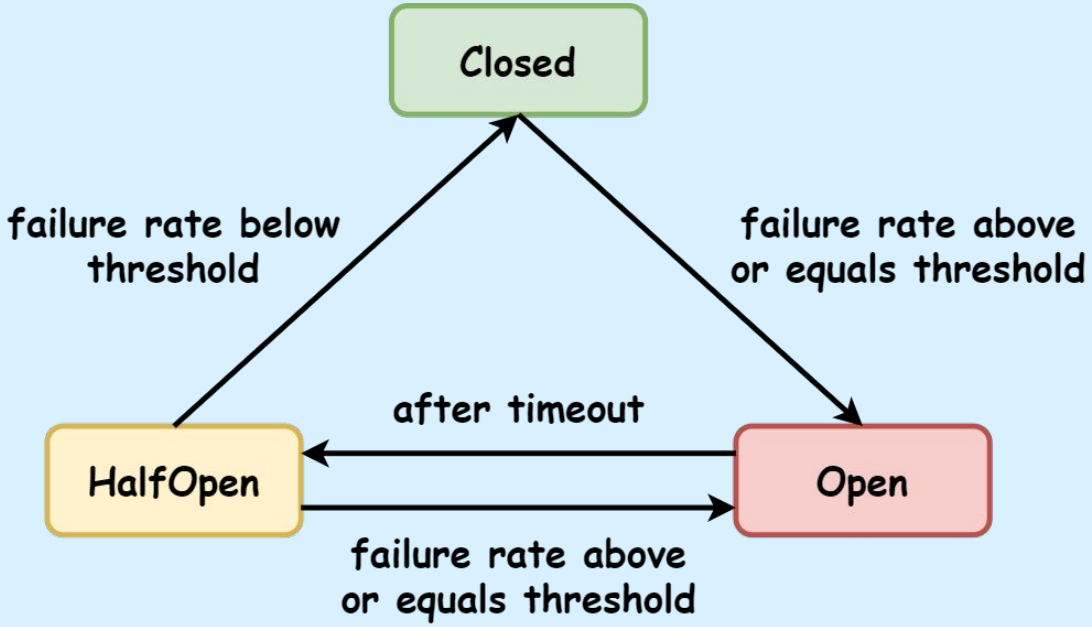


Circuit Breaker Mechanism

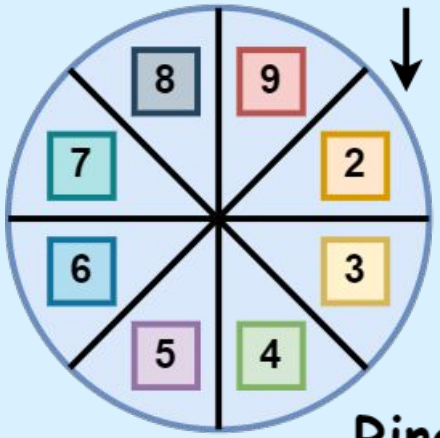
- **Reactive** resilience mechanism
- **Protects** component from overload/failure
- **Short-circuits requests** when component misbehaves
- **Monitors system health**



Circuit Breaker States



Circuit Breaker Sliding Window



Ring Buffer

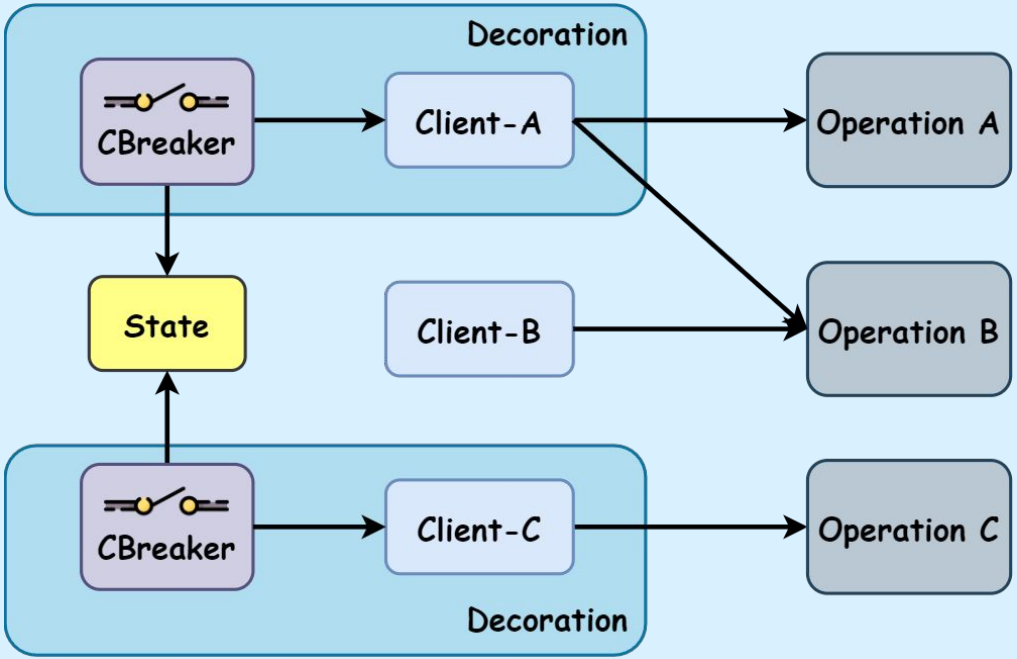
SlidingWindow

Available Types

Count Based
Last N calls

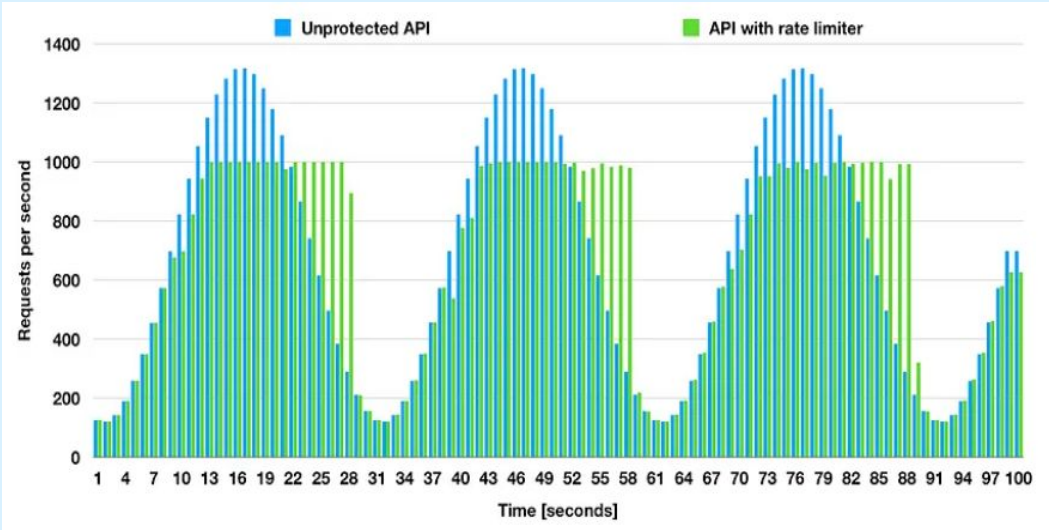
Time Based
Last N seconds

Circuit Breaker Decoration

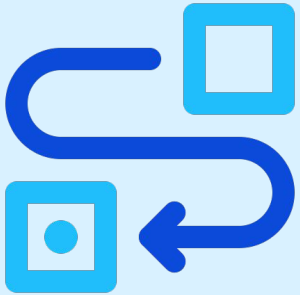


Rate Limiter Mechanism

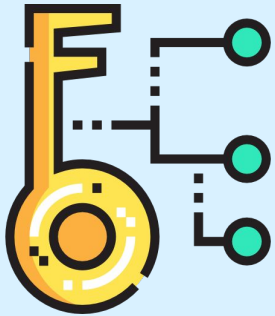
- **Proactive mechanism**
- **Limits requests to a component**
- **Could be bound to a time unit**
- **Protects systems from overloading**



Types of Rate Limiting



Total Requests

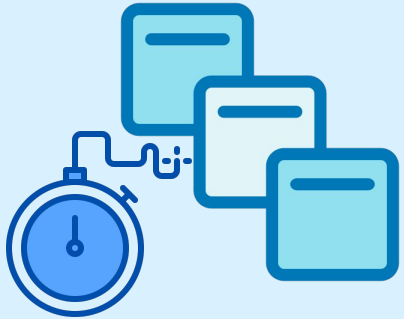


Key-based

Rate Limiting Exceeded

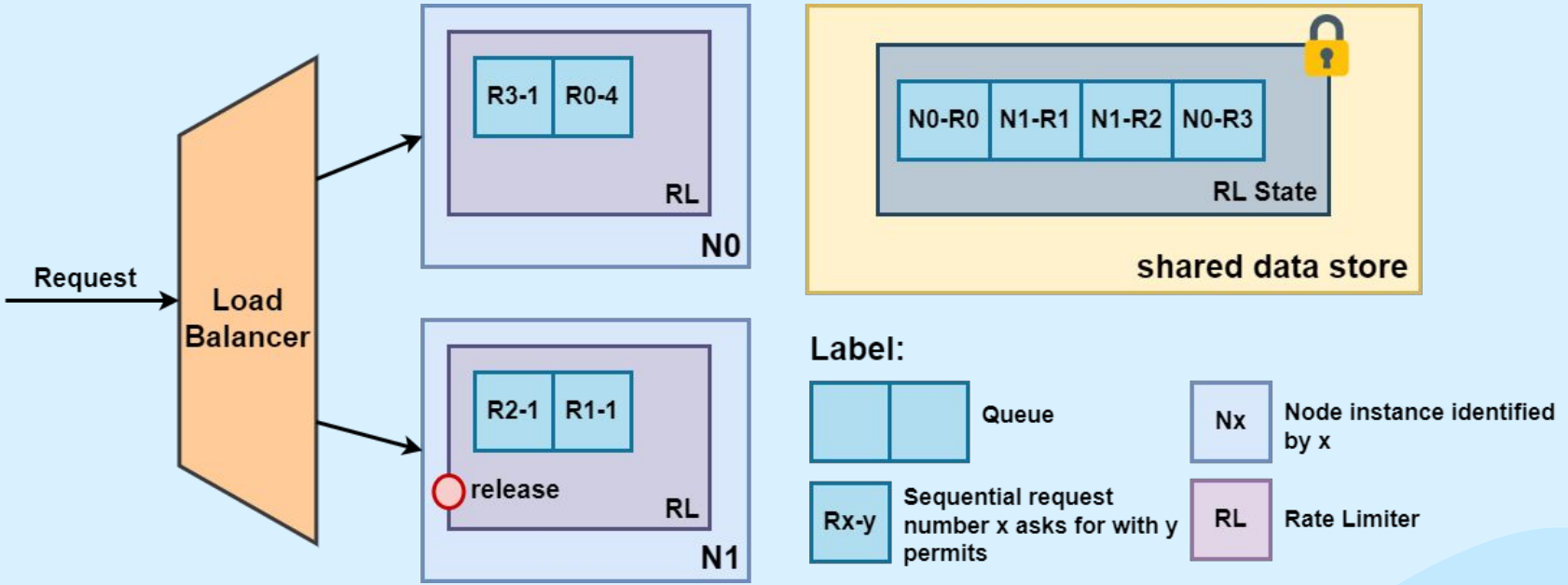
Reject: Immediately deny the request and return an error response message

Wait: Place the request in a queue to be processed later when the rate limit allows

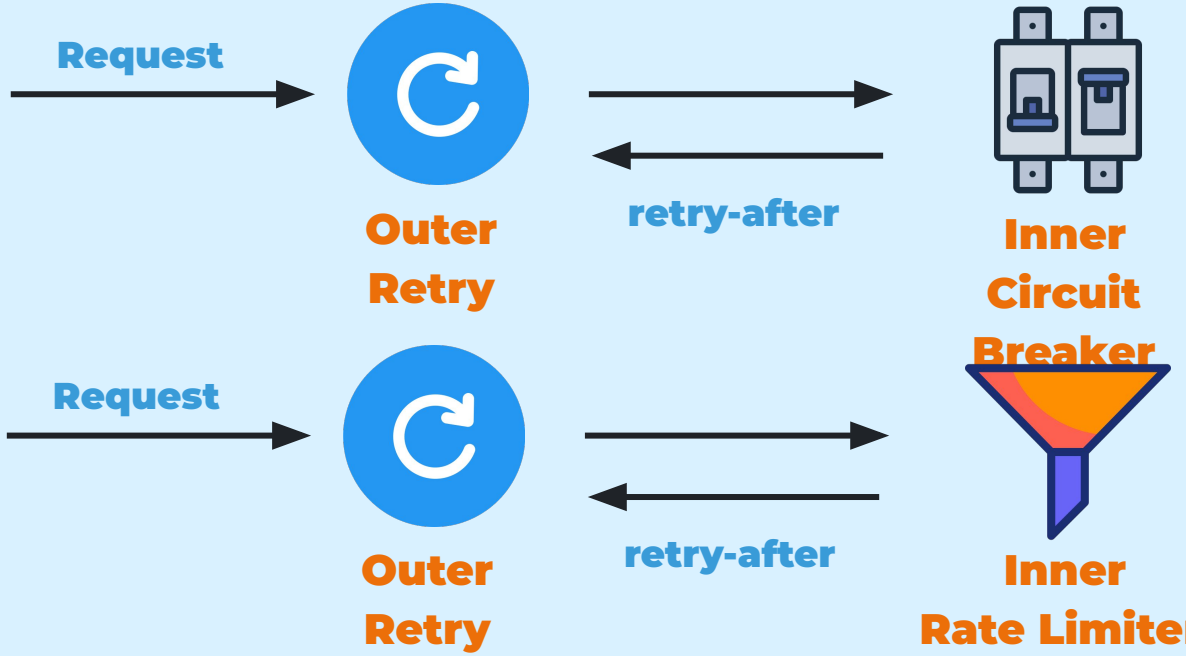


Both: Place request in the queue and reject after timeout expires

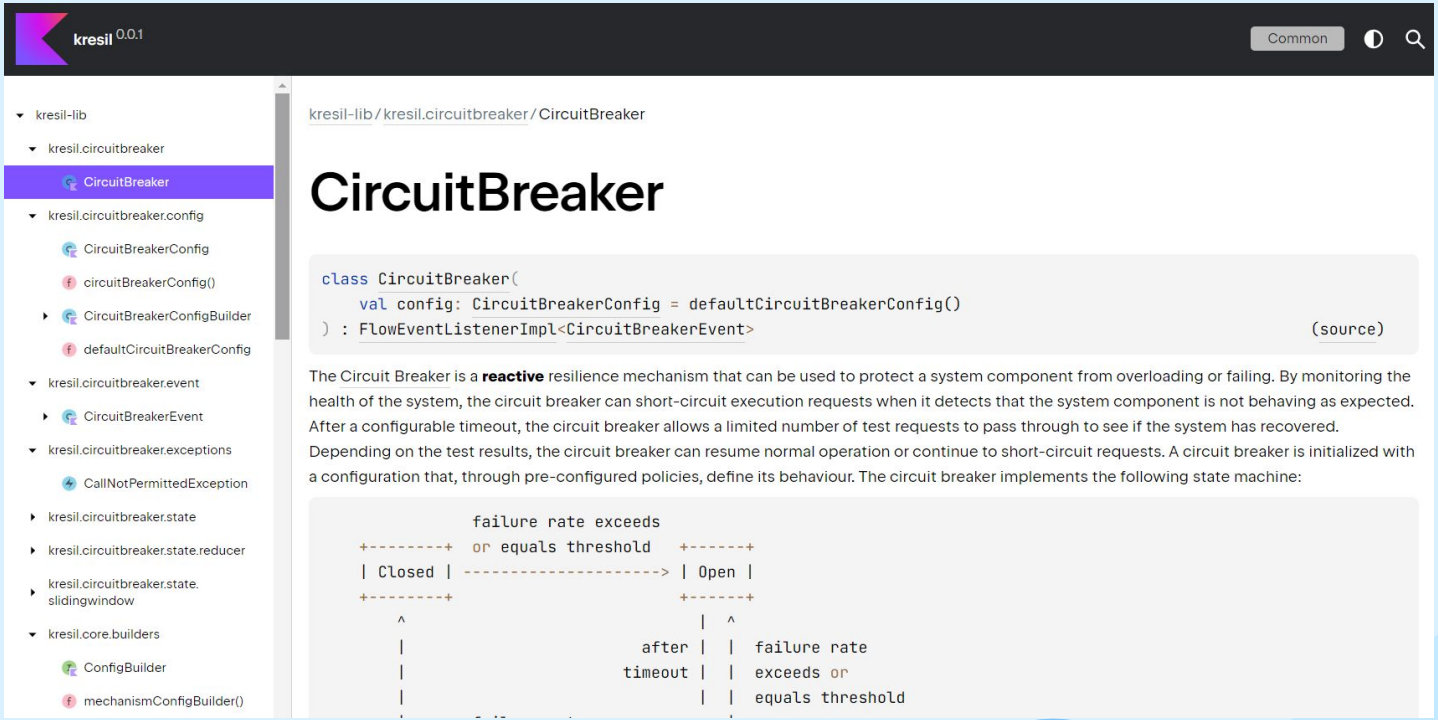
Distributed Rate Limiting



Mechanism Combination



API Documentation



Link: https://kresil.github.io/kresil/

Plugin Demos Architecture

Server

Client

Retry Plugin Demo

Client With Retry

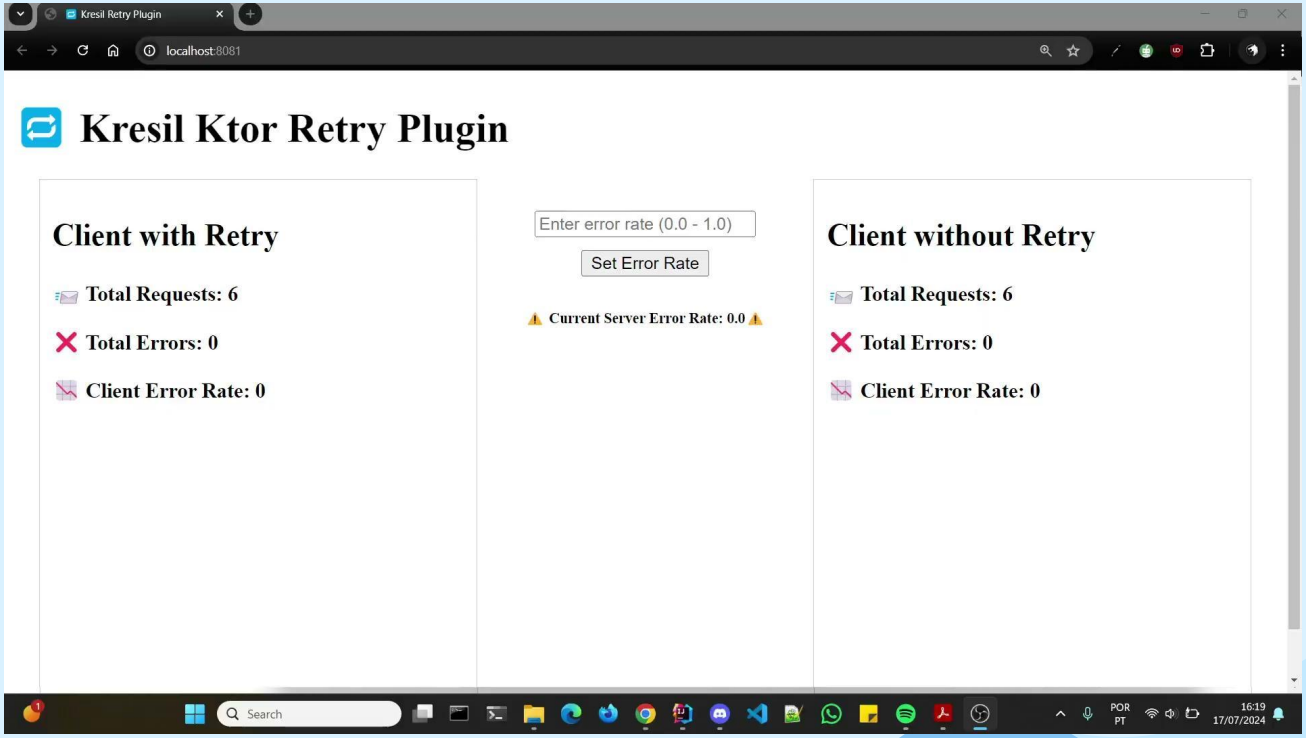
Client Without Retry

Overview:
- Server allows for adjustment of error rate
- Error rate affects request failure probability
- Clients display request error rate

Objective: Demonstrate improved success rate to a unreliable server

Server

Retry Plugin Demo



Circuit Breaker Plugin Demo

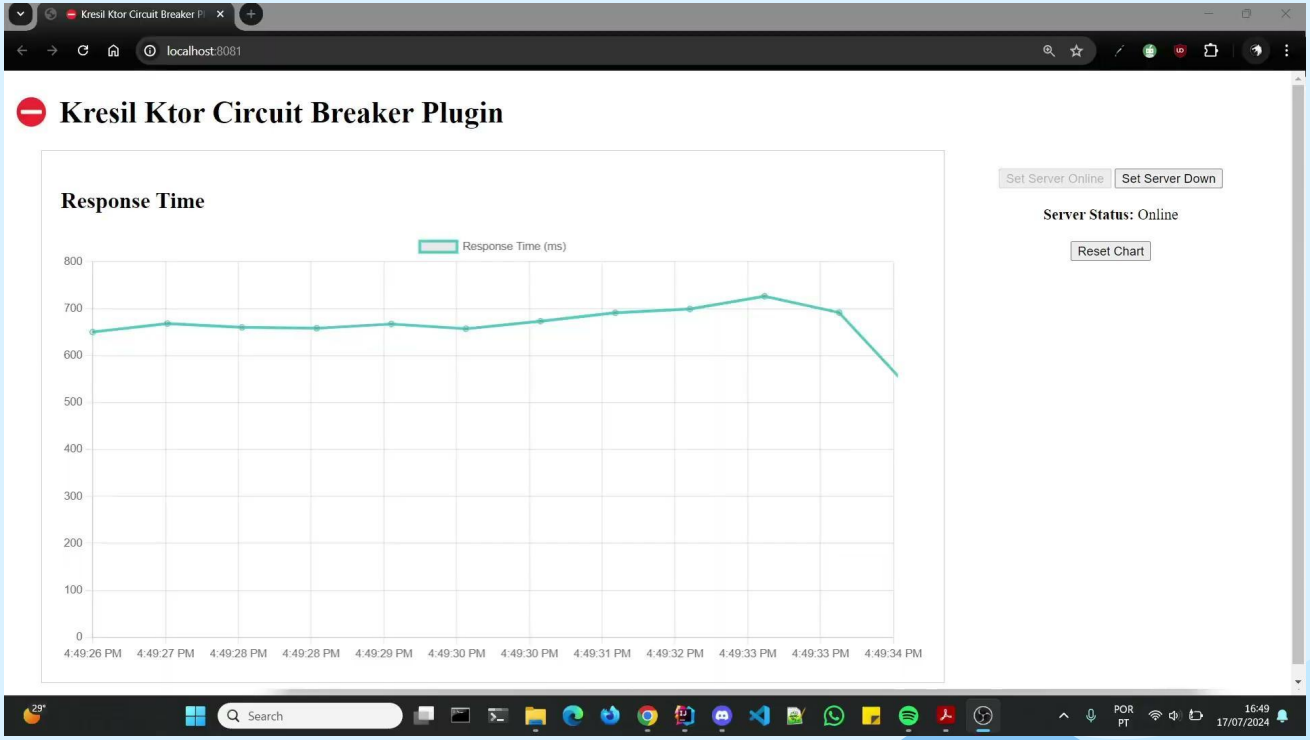
Client

Overview:
- Server allows configuration of response behavior (OK/NOK)
- Client monitors response time from server

Objective: Minimize request response time to a considered failing component (fail-fast)

Server

Circuit Breaker Plugin Demo



Rate Limiter Plugin Demo

Client

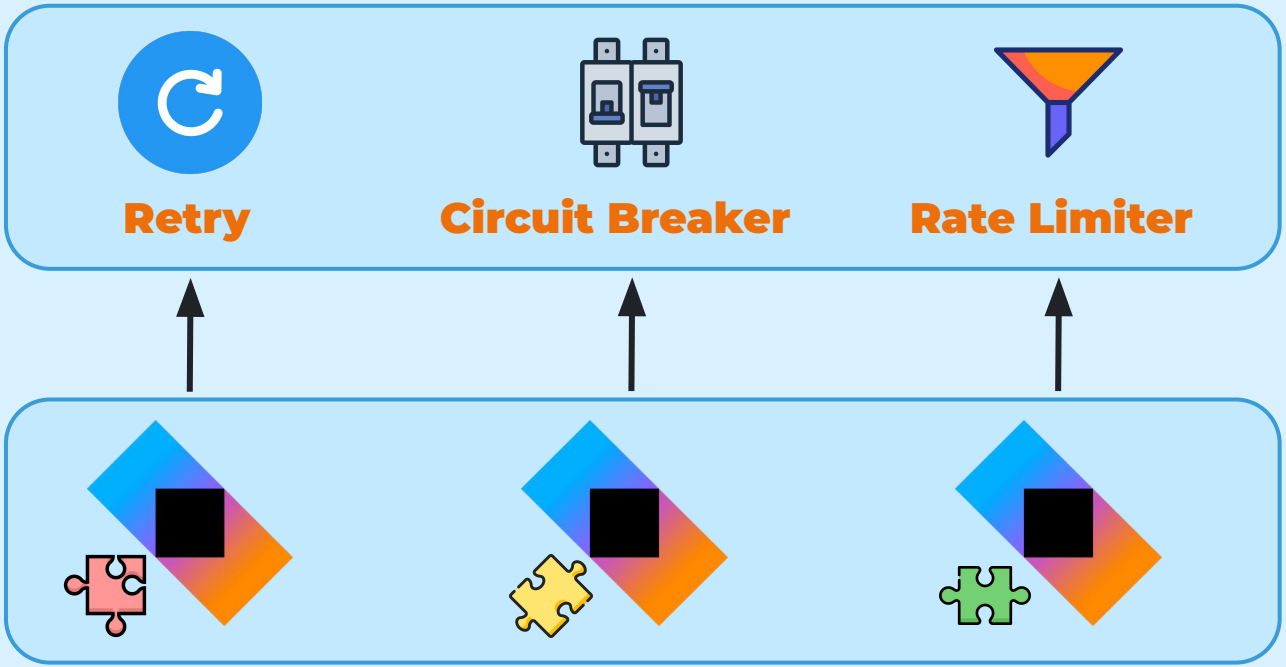
Overview:
- Server has rate-limited and unlimited routes
- Client plots requests made to both routes over a specific time unit

Objective: Observe rate-limited requests do not exceed a predefined limit; while unrated do

Server

Rate Limiter Plugin Demo

Conclusions



Software Engineering Practices