Kresil
Kotlin Resilience

Kotlin Multiplatform library for fault-tolerance with Ktor Integration

ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

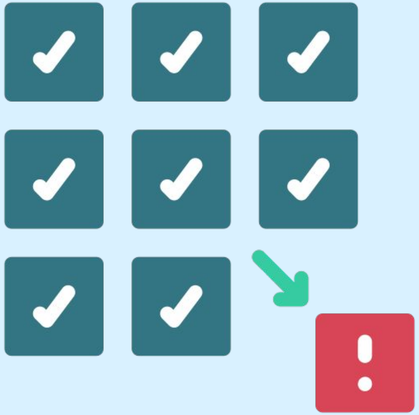
Supervisor: Prof. Pedro Félix
Author: Francisco Engenheiro - 49428
Project and Seminary
BSc in Computer Science and Engineering
Summer 2023/2024

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...

Existing Platform-Specific Solutions

Resilience4j

POLLY

Netflix’s Hystrix

Multiplatform Considerations

Concurrency Model

Time Management

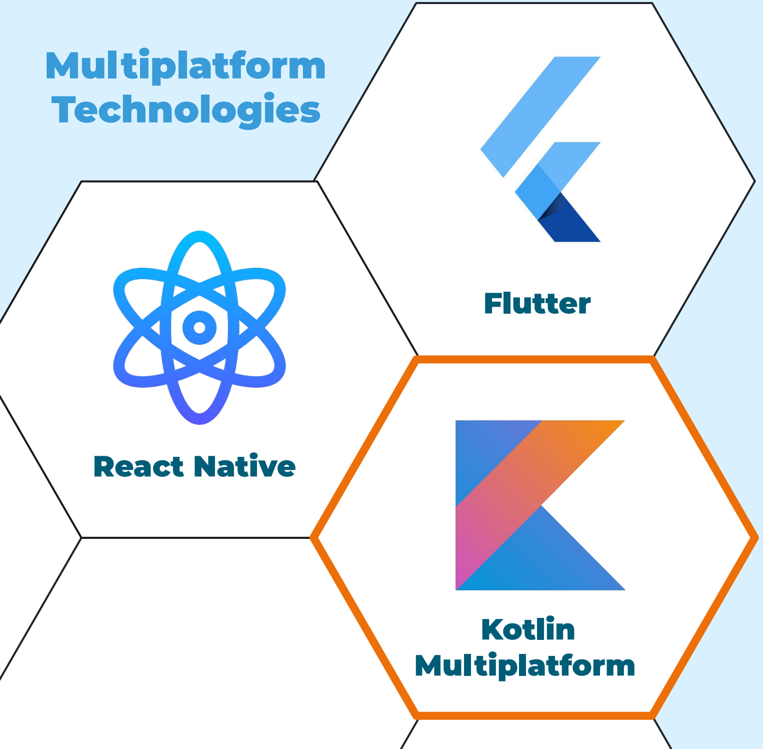
Synchronous vs Asynchronous

Mocking

Logging

Why Kotlin Multiplatform

Multiplatform Technologies


Flutter

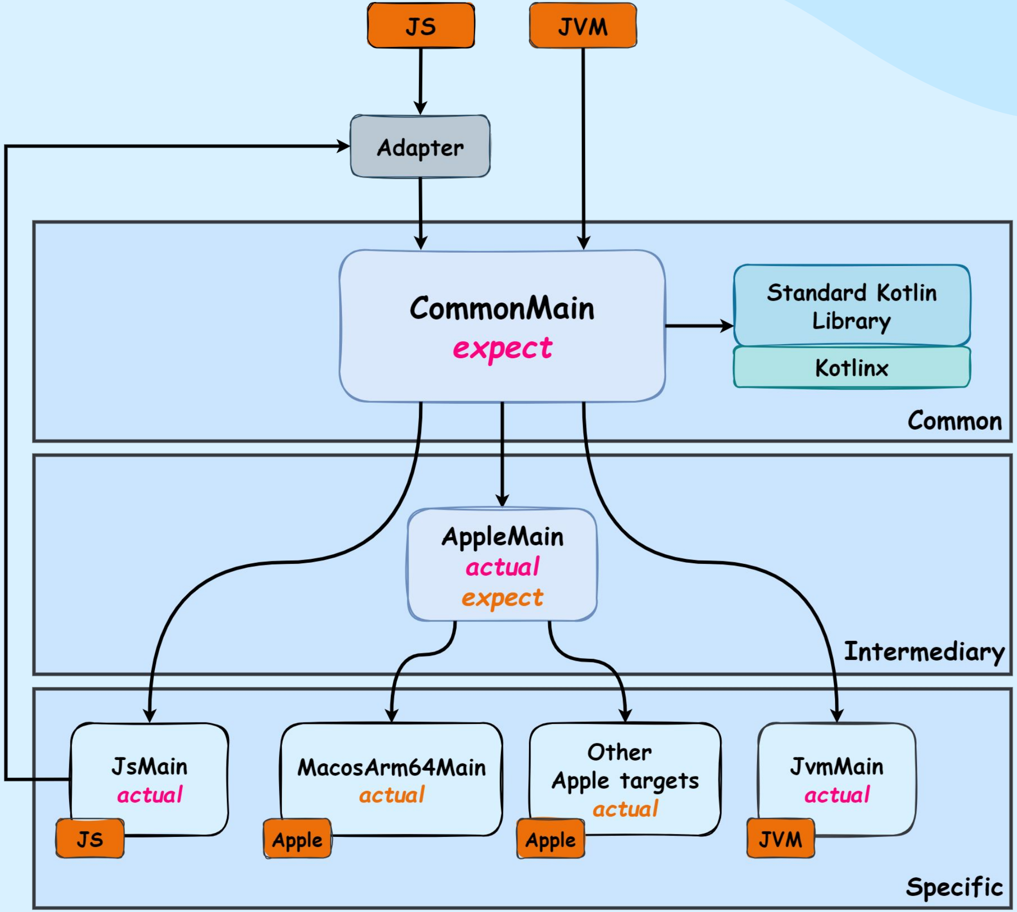
React Native

Kotlin Multiplatform

No libraries that provide resilience mechanisms in Kotlin Multiplatform with the same functionality of the platform-specific solutions

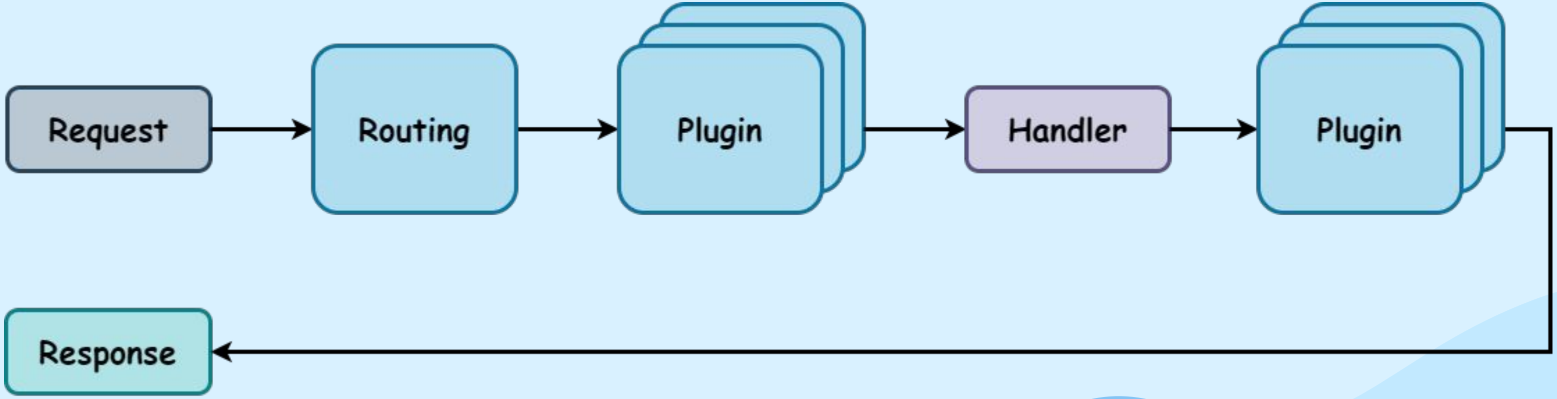
Arrow Library

Kotlin Multiplatform Architecture

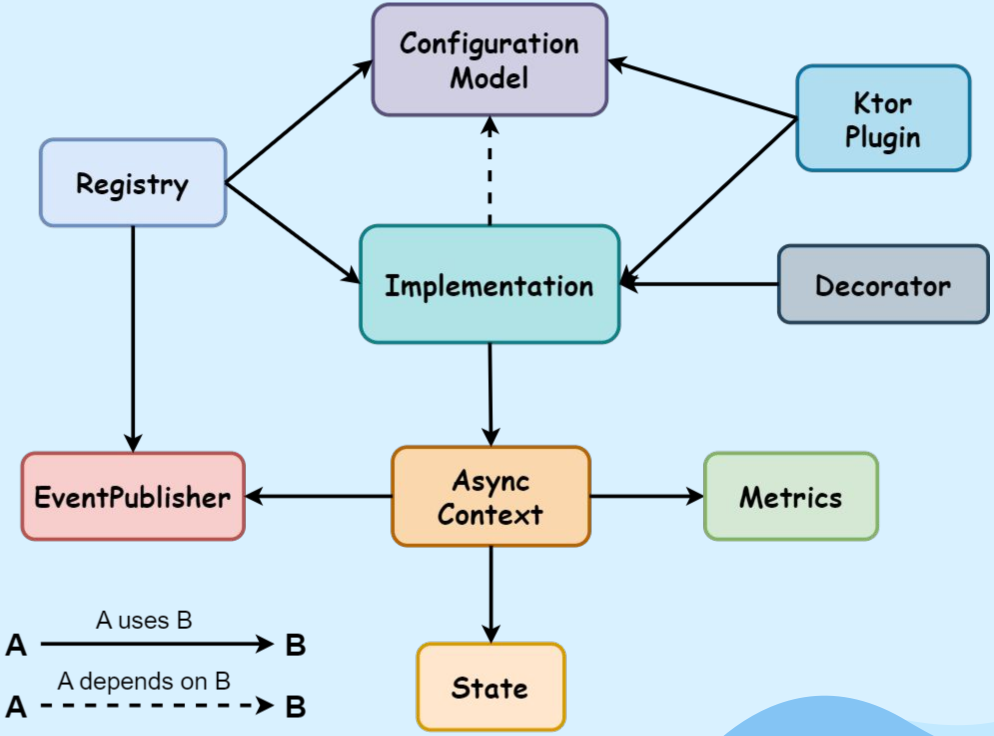


Ktor Framework

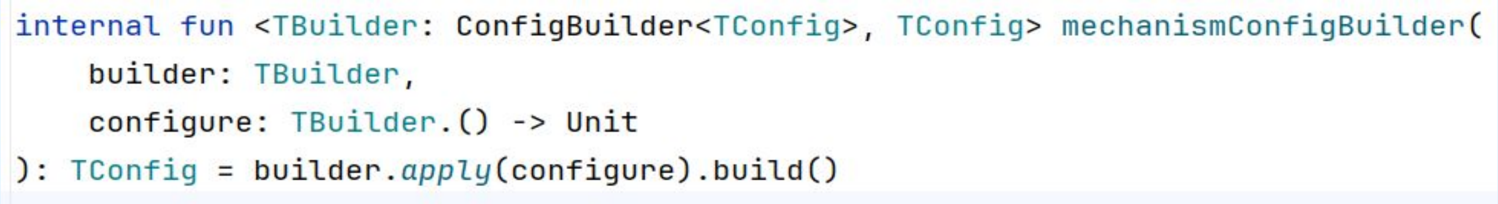
- Built with Kotlin Multiplatform;
- Asynchronous server and client development framework
- Based on the coroutines concurrency model;
- Modular



Mechanism Model

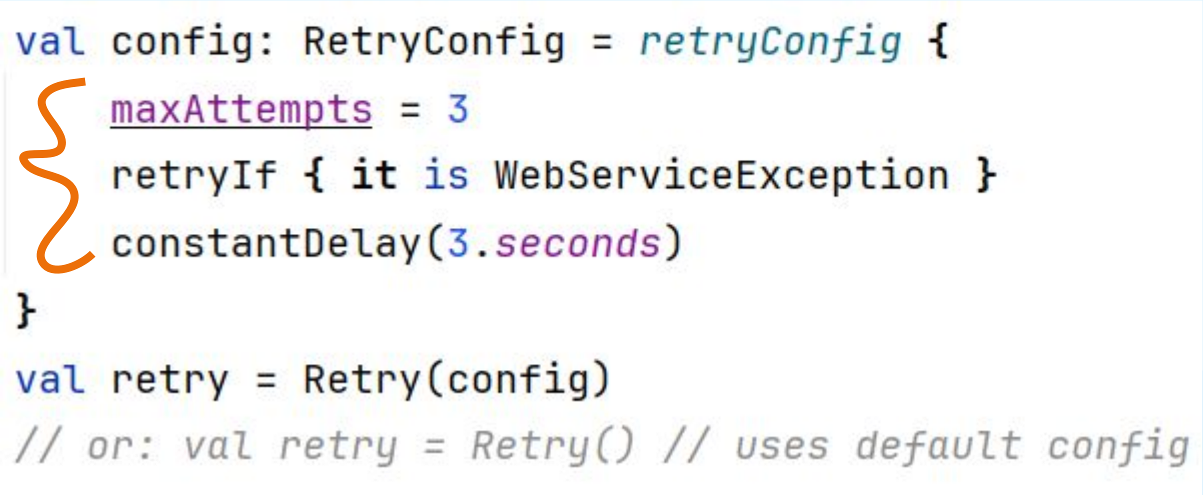


Mechanism Configuration



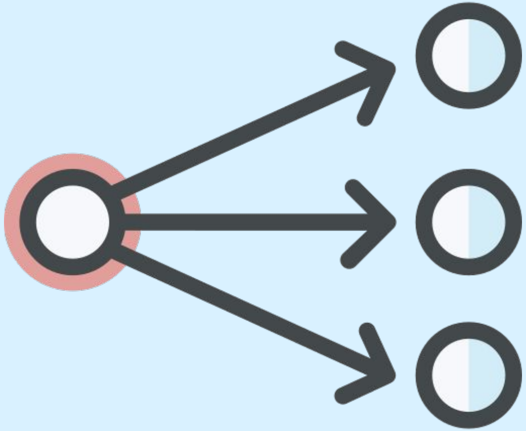
```kotlin
internal fun <TBuilder: ConfigBuilder<TConfig>, TConfig> mechanismConfigBuilder(
    builder: TBuilder,
    configure: TBuilder.() -> Unit
): TConfig = builder.apply(configure).build()
```

Policies - **Define the mechanism behaviour**



```kotlin
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

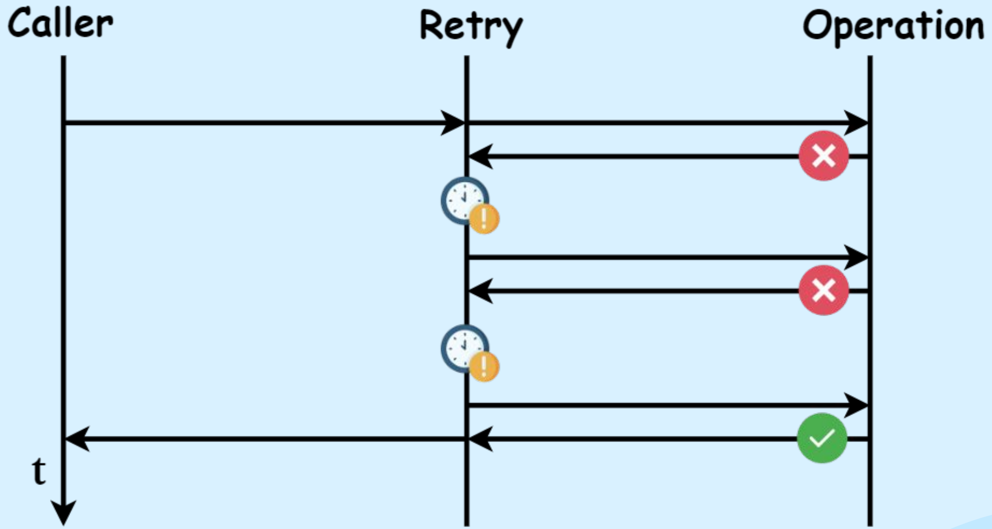


Each mechanism:

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

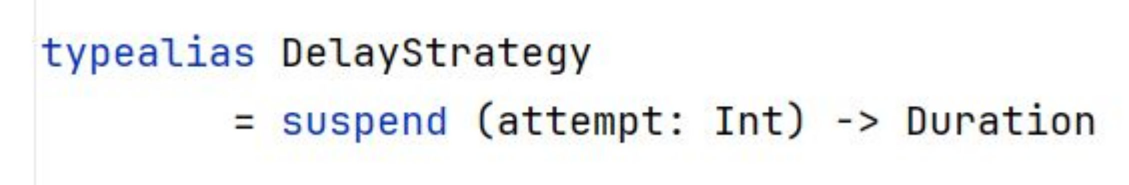
Retry Mechanism

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



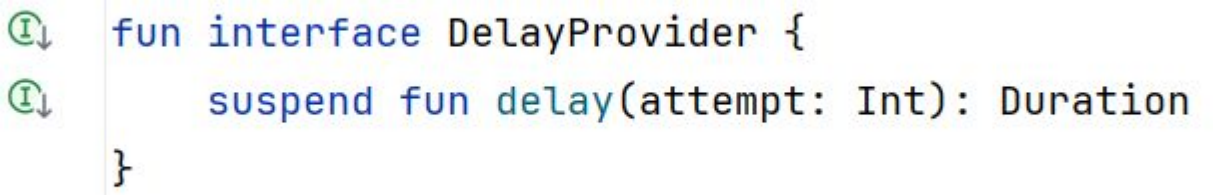
Delay Handling

Delay Strategy



```
typealias DelayStrategy
        = suspend (attempt: Int) -> Duration
```

Delay Provider

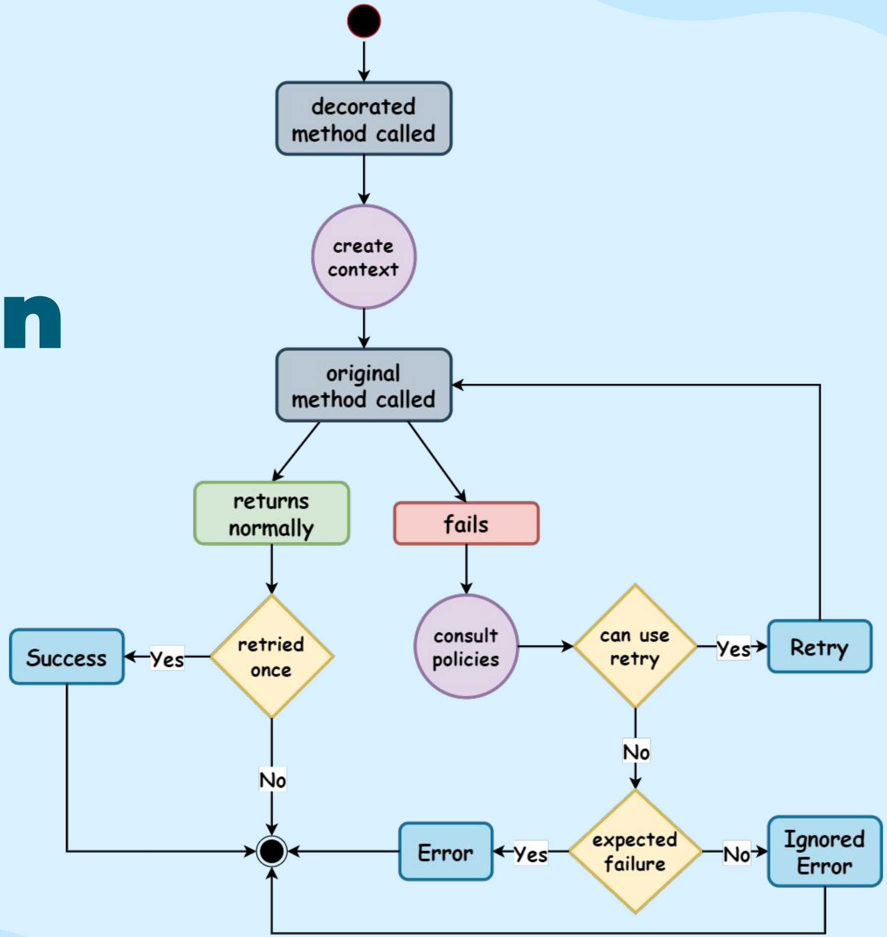


```
fun interface DelayProvider {
    suspend fun delay(attempt: Int): Duration
}
```

Options

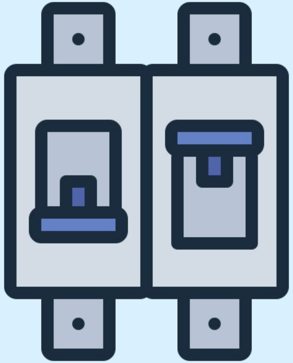
- No delay
- Constant delay
- Linear delay
- Exponential delay
- Custom Delay

Retry Execution Flow

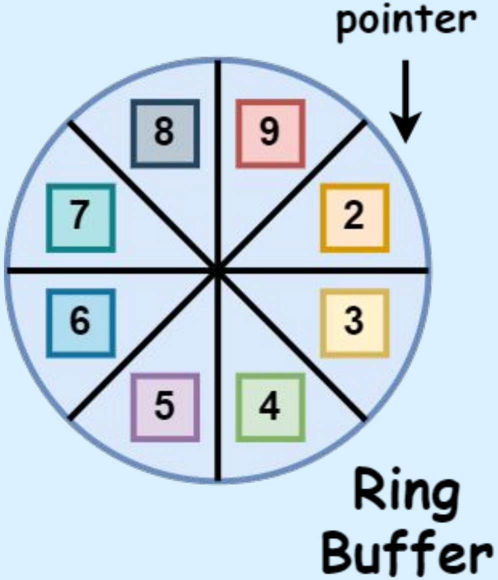


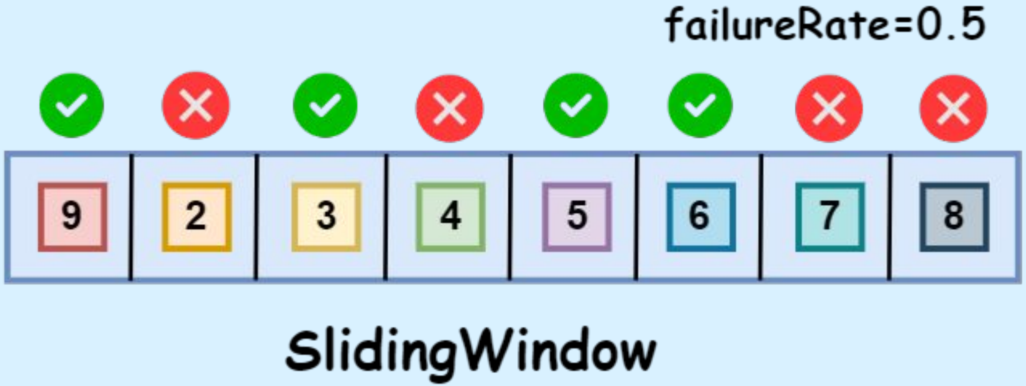
Circuit Breaker Mechanism

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



Circuit Breaker Sliding Window





Available Types

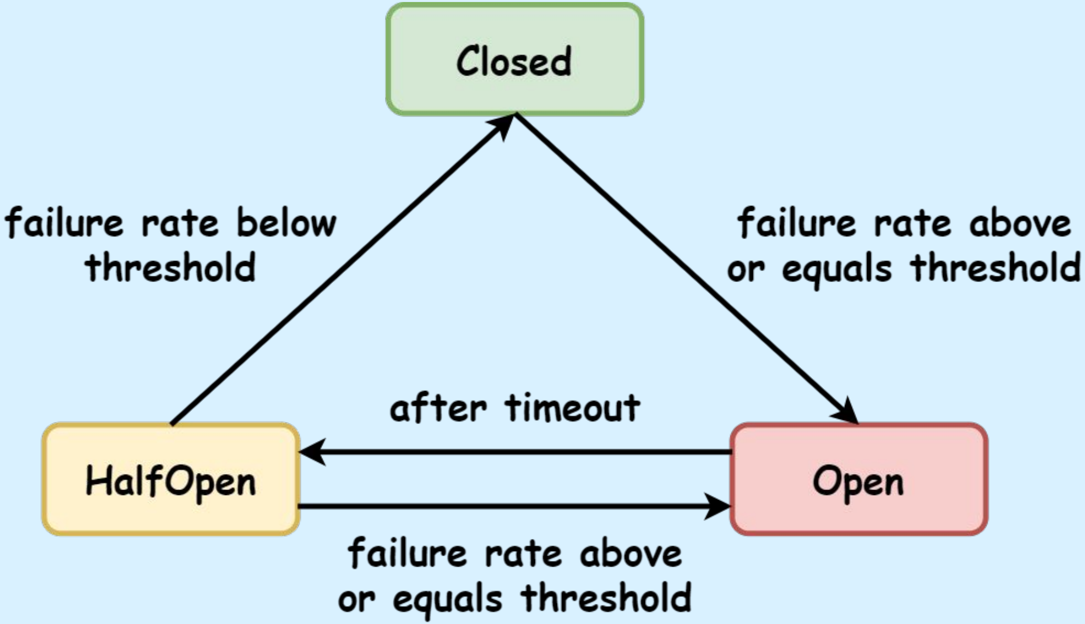
Count Based

Last N calls

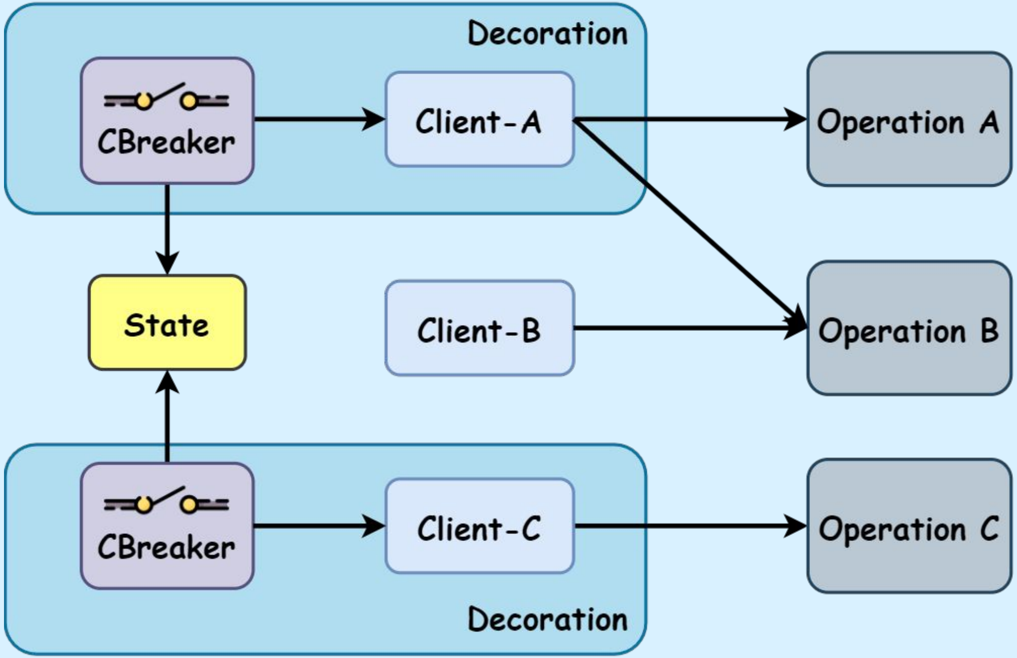
Time Based

Last N time units

Circuit Breaker States

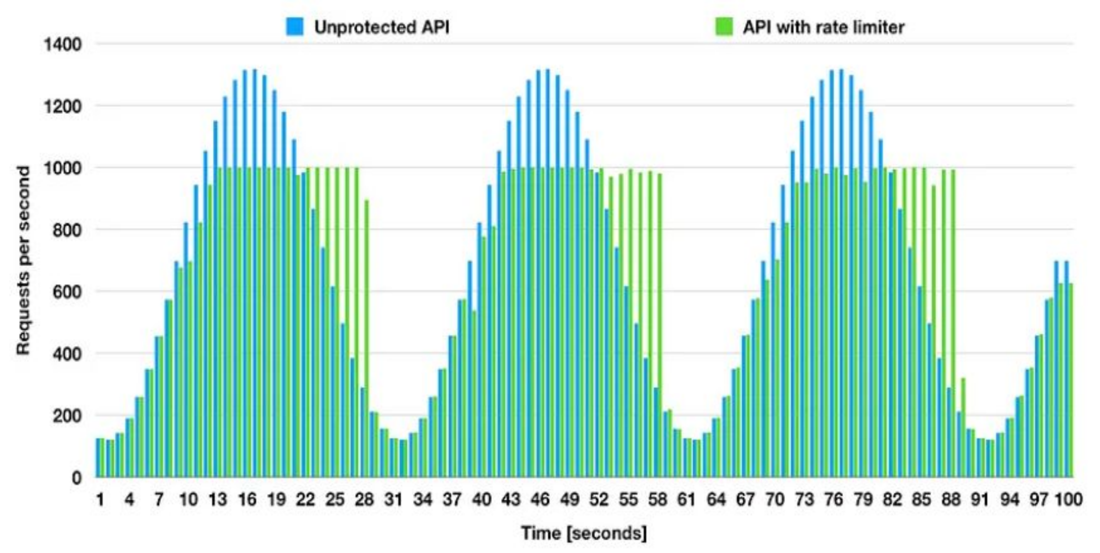

Closed
failure rate below threshold
failure rate above or equals threshold
after timeout
HalfOpen
Open
failure rate above or equals threshold

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

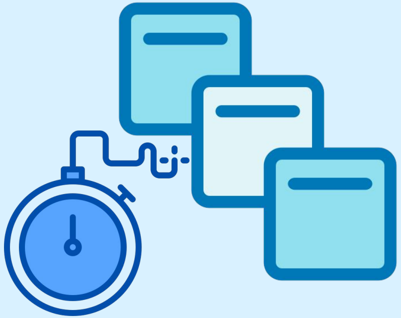
Total Requests

Key-Based

Rate Limit Exceeded

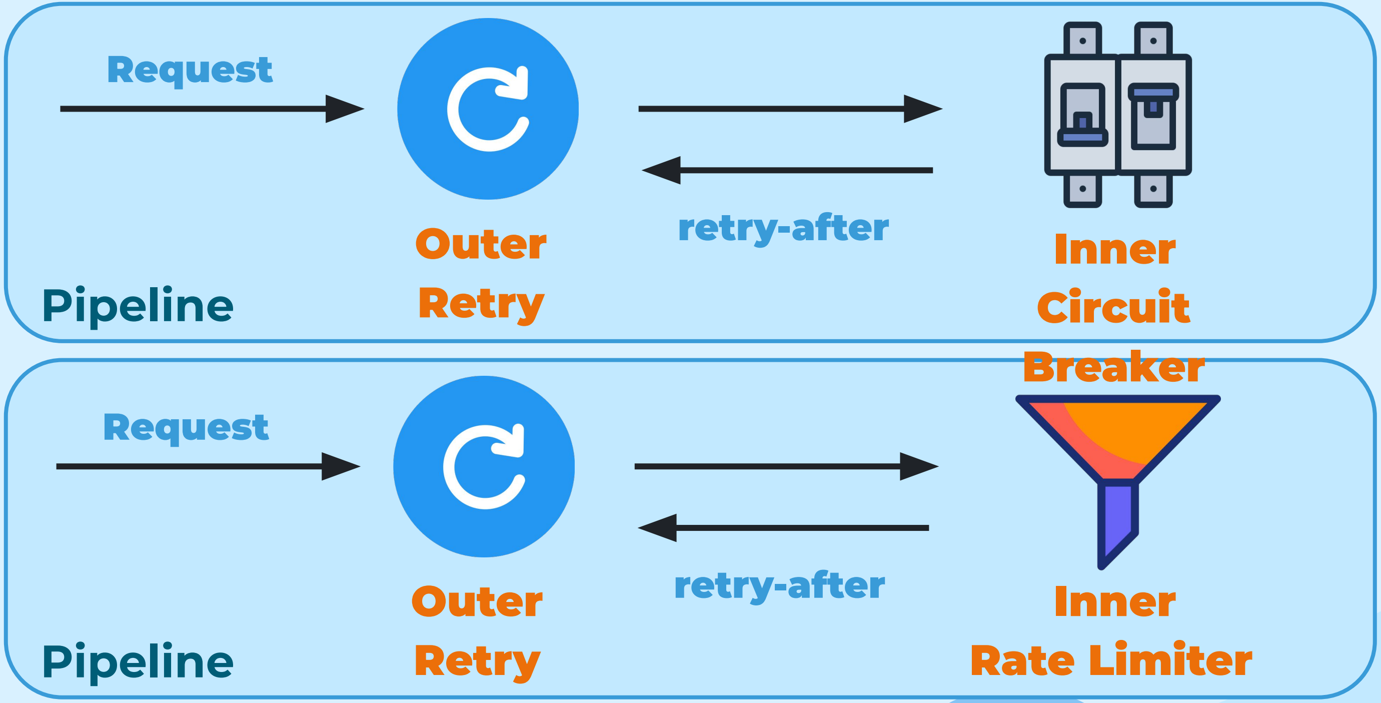
Reject: Immediately deny the request and return an error response message

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

Mechanism Combination



Request

Outer Retry

retry-after

Inner Circuit Breaker

Pipeline

Request

Outer Retry

retry-after

Inner Rate Limiter

Pipeline

Plugin Demos Architecture

Server

Client

Features:

- Runs **Kotlin in the browser**
- Attest **mechanism implementation**

Retry Plugin Demo

JS

Client
With Retry

JS

Client
Without
Retry

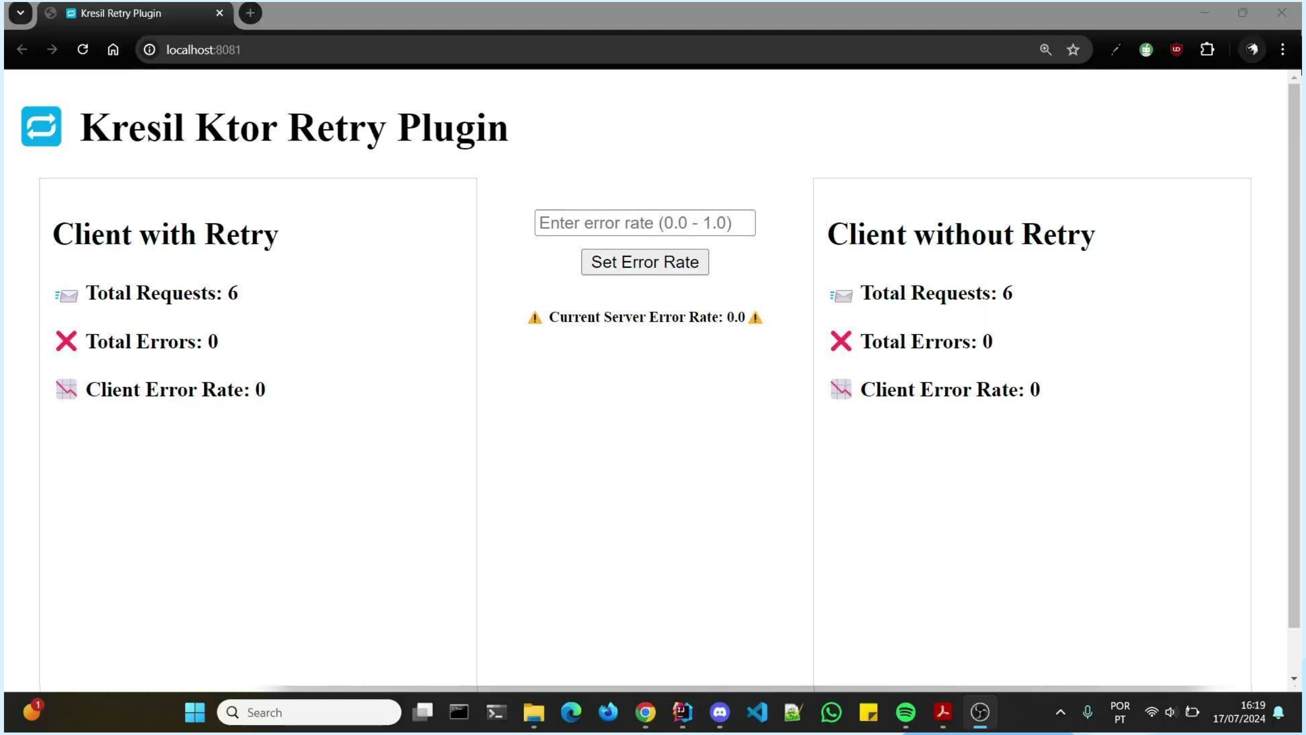
Overview:

- 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



Kresil Ktor Retry Plugin

Client with Retry

Total Requests: 6

❌ Total Errors: 0

Client Error Rate: 0

Enter error rate (0.0 - 1.0)

Set Error Rate

⚠ Current Server Error Rate: 0.0 ⚠

Client without Retry

Total Requests: 6

❌ Total Errors: 0

Client Error Rate: 0

Circuit Breaker Plugin Demo

Client

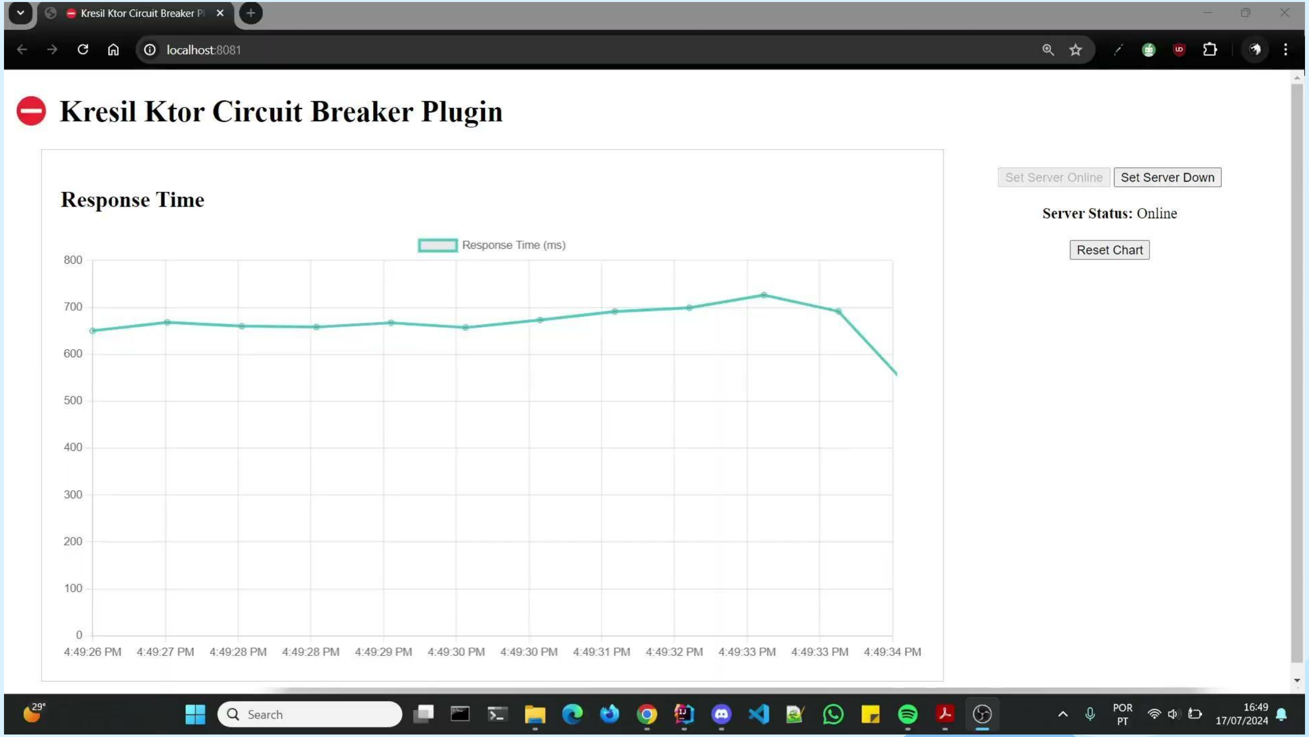
Overview:

- Server allows configuration of response behavior (OK/NOK)
- Server responds in a constant 500ms delay
- Client monitors response time from server
- Circuit Breaker deploys an exponential delay strategy in the Open State

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

Server

Circuit Breaker Plugin Demo



Kresil Ktor Circuit Breaker Plugin

Server Status: Online

Rate Limiter Plugin Demo

Client

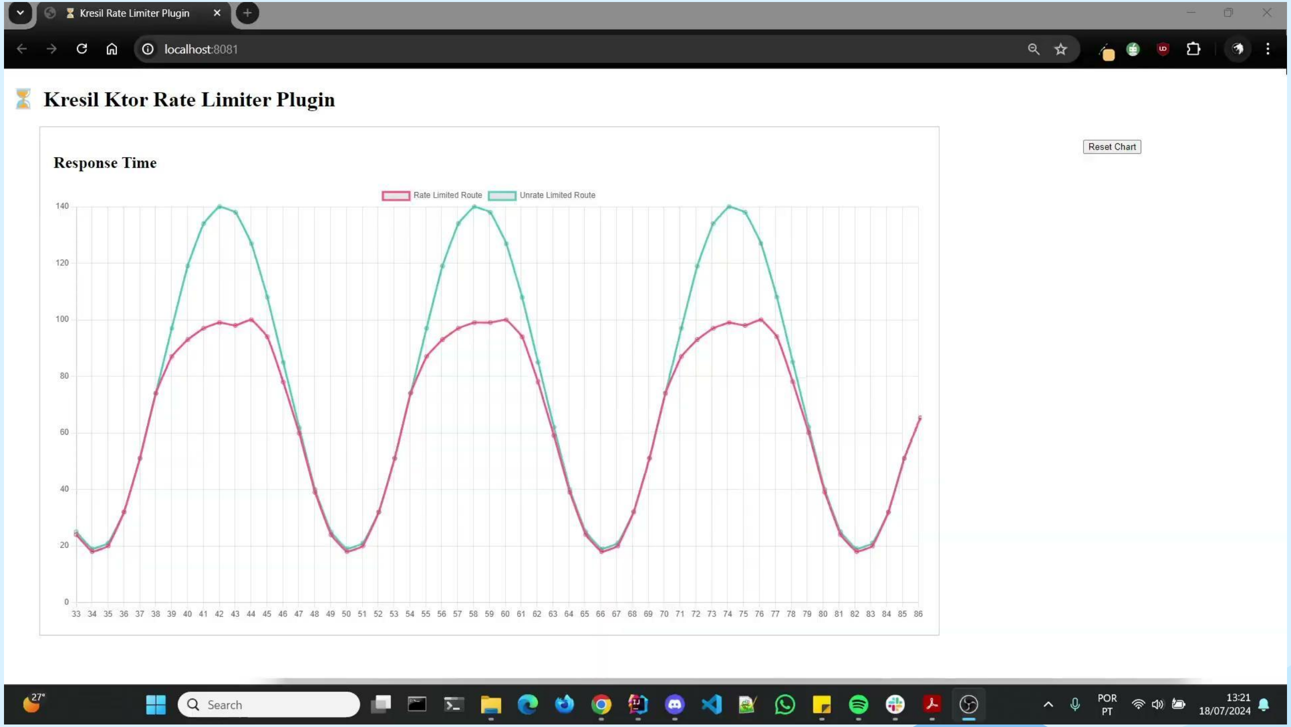
Overview:

- Server has rate-limited and unlimited routes
- Client plots requests made to both routes over a time period

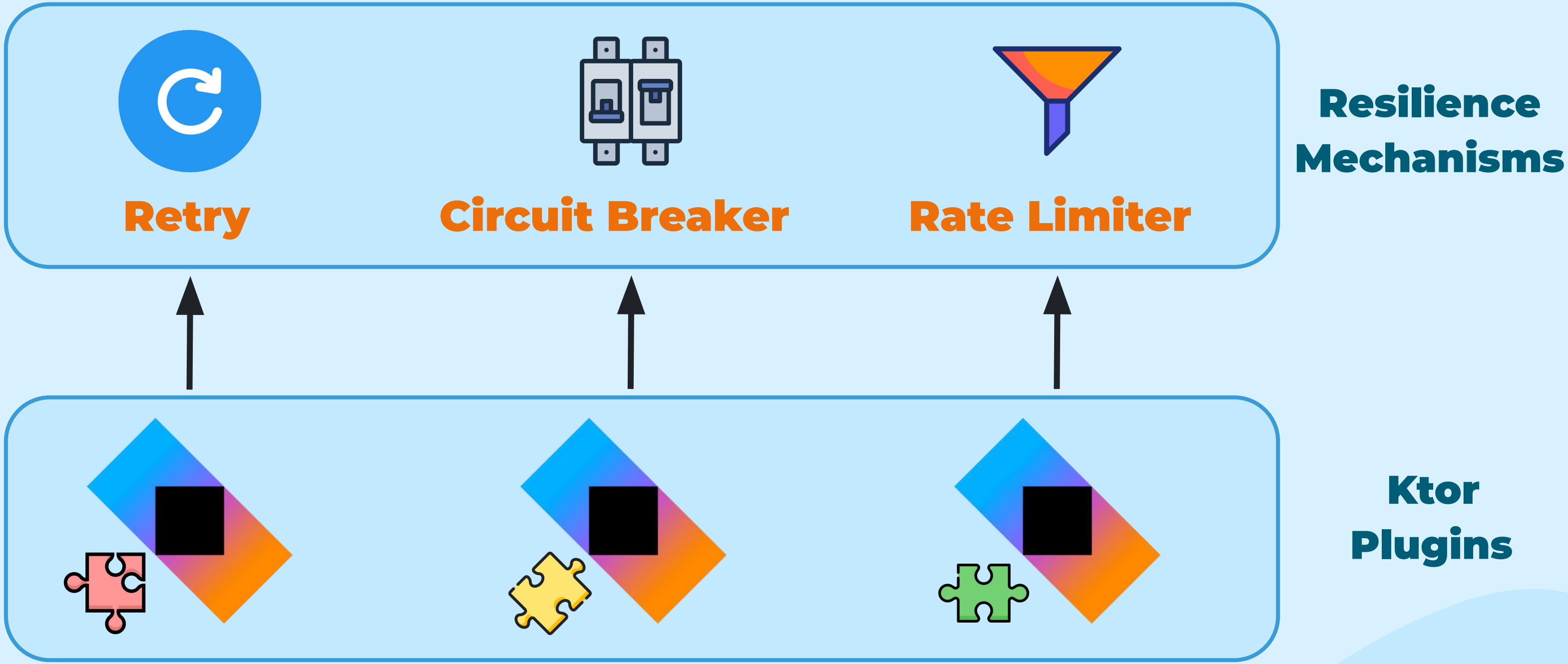
Objective: Observe rate-limited requests do not exceed a predefined limit; while unrated do

Server

Rate Limiter Plugin Demo

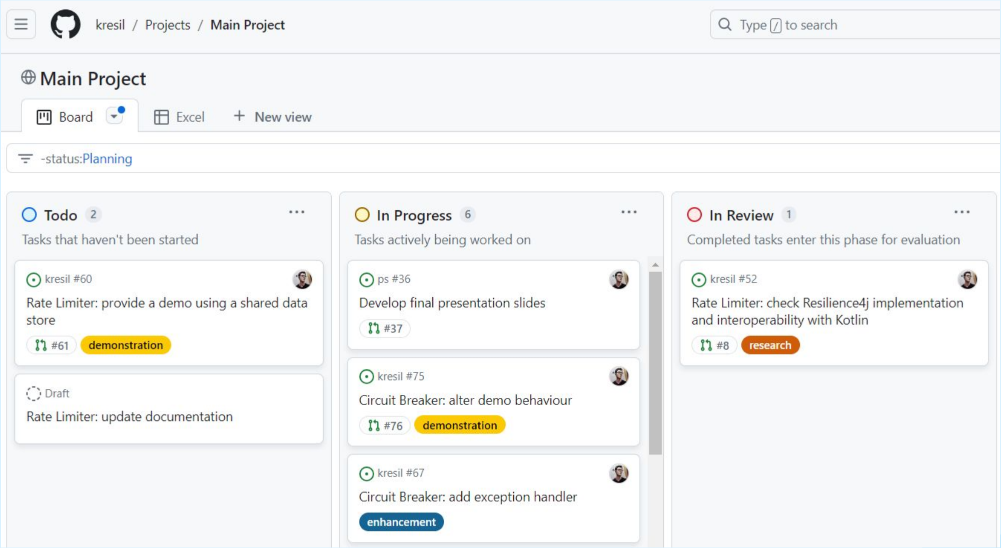


Conclusions



Software Engineering Practices

- Task management
- Descriptive issues
- PR's and branches
- Workflows
- Documentation and examples of usage
- Tests



kresil / Projects / Main Project

Type / to search

Main Project

Board | Excel | + New view

-status:Planning

Todo 2

Tasks that haven't been started

- kresil #60 — Rate Limiter: provide a demo using a shared data store — #61 — demonstration
- Draft — Rate Limiter: update documentation

In Progress 6

Tasks actively being worked on

- ps #36 — Develop final presentation slides — #37
- kresil #75 — Circuit Breaker: alter demo behaviour — #76 — demonstration
- kresil #67 — Circuit Breaker: add exception handler — enhancement

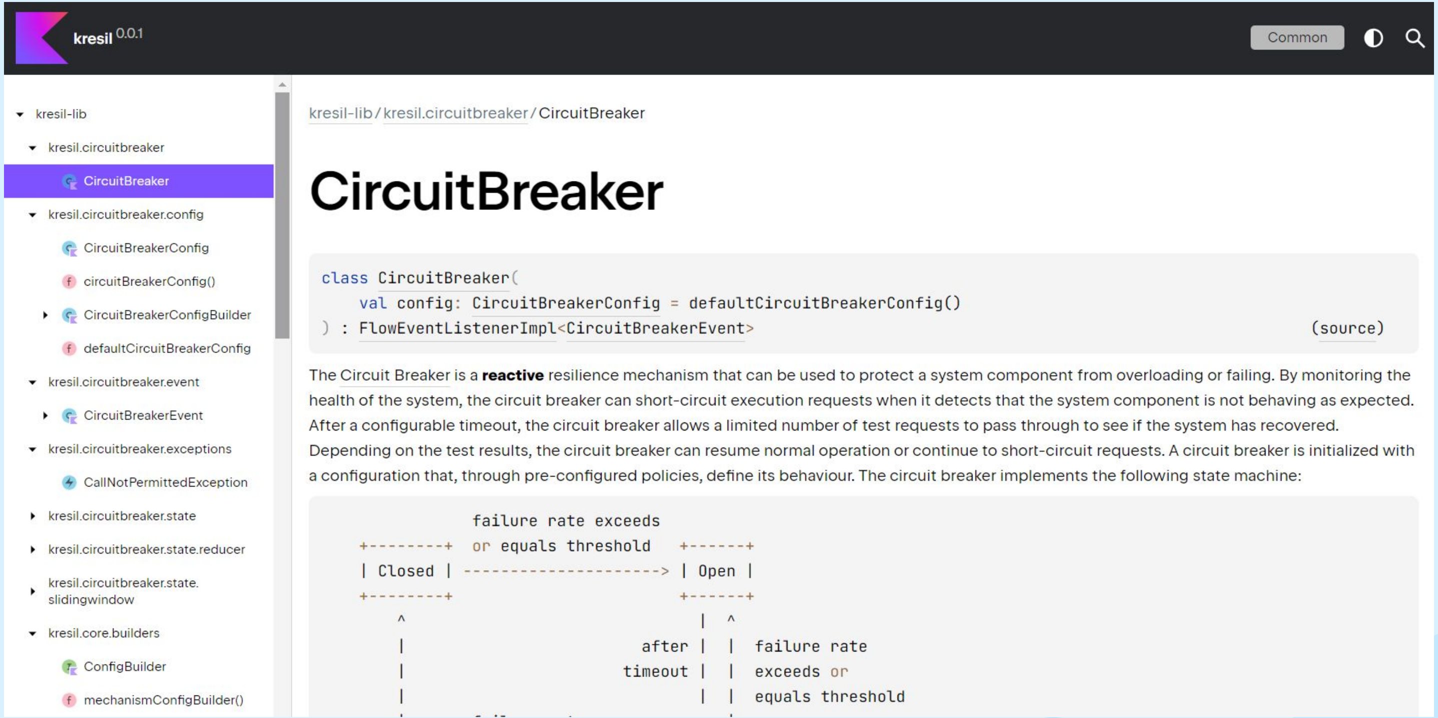
In Review 1

Completed tasks enter this phase for evaluation

- kresil #52 — Rate Limiter: check Resilience4j implementation and interoperability with Kotlin — #8 — research

All to promote asynchronous collaboration and simulate real work environments

API Documentation


kresil 0.0.1
Common
kresil-lib
kresil.circuitbreaker
CircuitBreaker
kresil.circuitbreaker.config
CircuitBreakerConfig
circuitBreakerConfig()
CircuitBreakerConfigBuilder
defaultCircuitBreakerConfig
kresil.circuitbreaker.event
CircuitBreakerEvent
kresil.circuitbreaker.exceptions
CallNotPermittedException
kresil.circuitbreaker.state
kresil.circuitbreaker.state.reducer
kresil.circuitbreaker.state.slidingwindow
kresil.core.builders
ConfigBuilder
mechanismConfigBuilder()

kresil-lib / kresil.circuitbreaker / CircuitBreaker

CircuitBreaker

```
class CircuitBreaker(
    val config: CircuitBreakerConfig = defaultCircuitBreakerConfig()
) : FlowEventListenerImpl<CircuitBreakerEvent>                                          (source)
```

The Circuit Breaker is a **reactive** resilience mechanism that can be used to protect a system component from overloading or failing. By monitoring the health of the system, the circuit breaker can short-circuit execution requests when it detects that the system component is not behaving as expected. After a configurable timeout, the circuit breaker allows a limited number of test requests to pass through to see if the system has recovered. Depending on the test results, the circuit breaker can resume normal operation or continue to short-circuit requests. A circuit breaker is initialized with a configuration that, through pre-configured policies, define its behaviour. The circuit breaker implements the following state machine:

```
                failure rate exceeds
    +--------+  or equals threshold   +------+
    | Closed | ---------------------> | Open |
    +--------+                        +------+
        ^                               |  ^
        |                         after |  |  failure rate
        |                       timeout |  |  exceeds or
        |                               |  |  equals threshold
```

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