https://github.com/mercadolibre/java-meli-toolkit
https://github.com/mercadolibre/fury_java-melitk-restclient#retry-strategies

1. Para esto debemos usar una estrategia que tome en cuenta los reintentos.

Con new Relic podemos saber el promedio en función a estadísticas el tema de las llamadas.

Simple Retry Strategy

```
Response response = restClient
    .withRetryStrategy(new SimpleRetryStrategy(MAX_RETRIES, WAIT_MS))
    .get("http://yourdomain.com/resource");
```

○

Limited Retry Strategy

```
RetryStrategy inner = new SimpleRetryStrategy(MAX_RETRIES, WAIT_MS); //if rate limiter allows a retry, delegate retry decision to this strategy.
RetryStrategy limited = new LimitedRetryStrategy(MAX_RETRY_RPM, inner); -
```

o

Exponential Backoff Retry Strategy

It'll increase wait time between retries between a min and a max value, growing exponentially between runs

```
Response response = restClient
    .withRetryStrategy(new ExponentialBackoffRetryStrategy(MIN_MS, MAX_MS))
    .get("http://yourdomain.com/resource");
```

Esto va a depender mucho del tráfico esperado y del que queramos q la app tenga. Ya que el objetivo es que el time de respuesta no sea más de 500ms y a la vez tenga un error rate menor a 0,5%.

500%2 = 250

Se puede hacer un máximo de 1 re intentos con una espera de 250ms. Con un total de timeout de nuestra app de 500ms. Esto aplica exponetial backoff retry strategy.

Sino, se puede poner un timeout al principio de 150ms, y luego de 300ms.

Sino, otra solución puede ser enfocarse en el P95 (con re intentos de 150ms hasta 2 reintentos), quedando con un error de 0,5 que es el máximo permitido.

2. Se podría aplicar un limit strategy. Bajando la cantidad de re intentos a 1 y aumentando la espera a 250ms.

Cambiar el status code a 504 que es más específico.

Si devuelve 429 dejar de re intentar. Sería aplicar un circuit breaker.