**Eureka Server :** localhost:8761

**Zipkin:** localhost:9411

**RabbitMq :** localhost:15672 username:guest

Password:guest

After creating review , review service updates the company ms rating by taking the average as well as job microservice as well

**CircuitBraker using resiliency4j:** <http://localhost:8082/actuator/circuitbreakers>

**✅ Resilience4j Implementation - Circuit Breaker, Fallback, and Retry**

In my microservices-based project, I implemented **Resilience4j** to handle fault tolerance and ensure high availability between services. Specifically, I used the **Circuit Breaker**, **Fallback**, and **Retry** patterns to make the system more resilient in case of service failures or latency spikes.

**🔌 Circuit Breaker**

I applied the @CircuitBreaker annotation to service calls that communicate with other microservices — for instance, when the **Job Service** calls the **Company Service**. The circuit breaker monitors these external calls, and if the failure rate exceeds a configured threshold (like 50% of 5 recent calls), it "opens" the circuit to prevent further attempts for a fixed time period. This avoids overloading an already failing service and helps maintain system stability.

**🛠 Fallback Mechanism**

To provide a graceful user experience even when downstream services are down, I implemented **fallback methods**. If the company microservice is unavailable or times out, the fallback logic returns a default response — such as a "Company not found" placeholder — instead of propagating the failure to the user. This ensures the system degrades gracefully rather than crashing or returning a 500 error.

**🔁 Retry Pattern**

In addition to the circuit breaker, I also applied the @Retry annotation to automatically **retry failed requests** a few times before falling back. This is especially useful for transient errors like temporary network issues or service start-up delays. The retry mechanism is configurable in terms of the number of attempts and the wait duration between them.

**Postgres:** localhost:5050

Pass: postgres