Arquitectura e Implementación de Microservicios

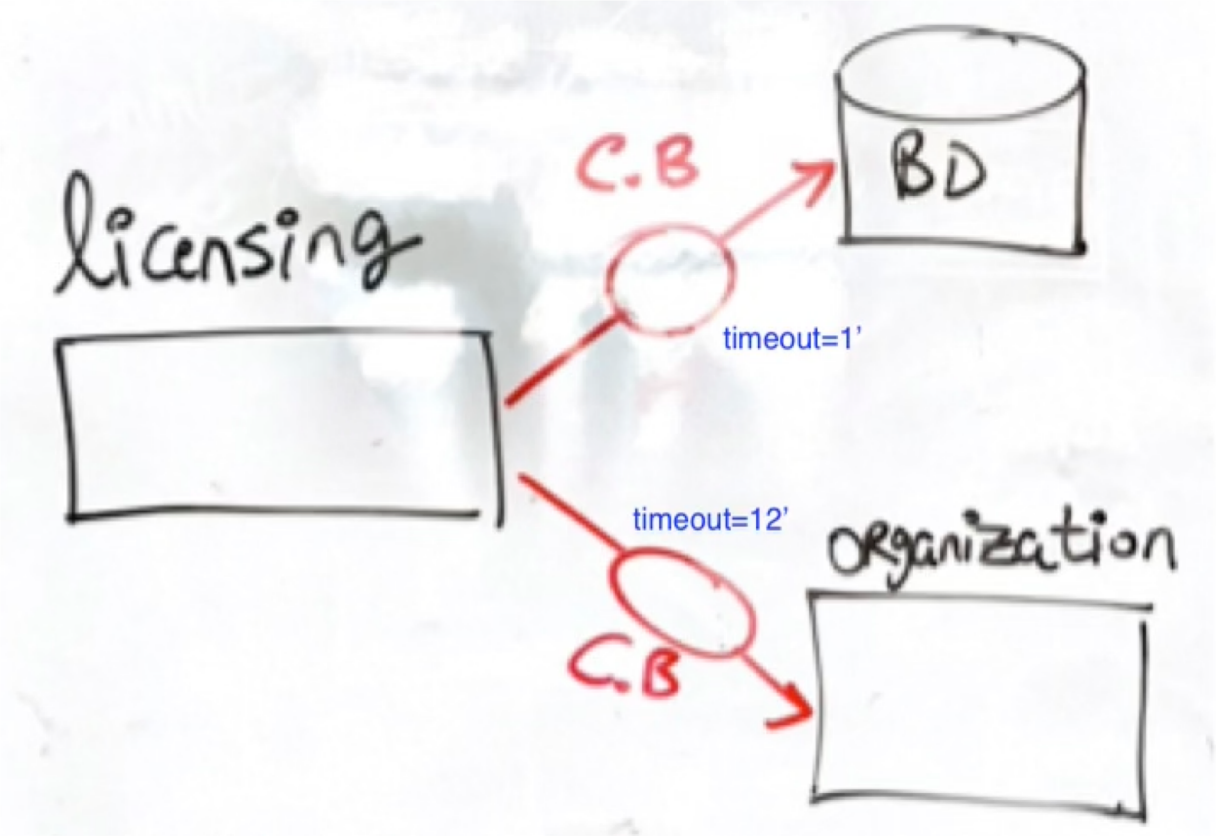
Sesion 4

OS ENTERPRISE CONSULTING

Hystrix

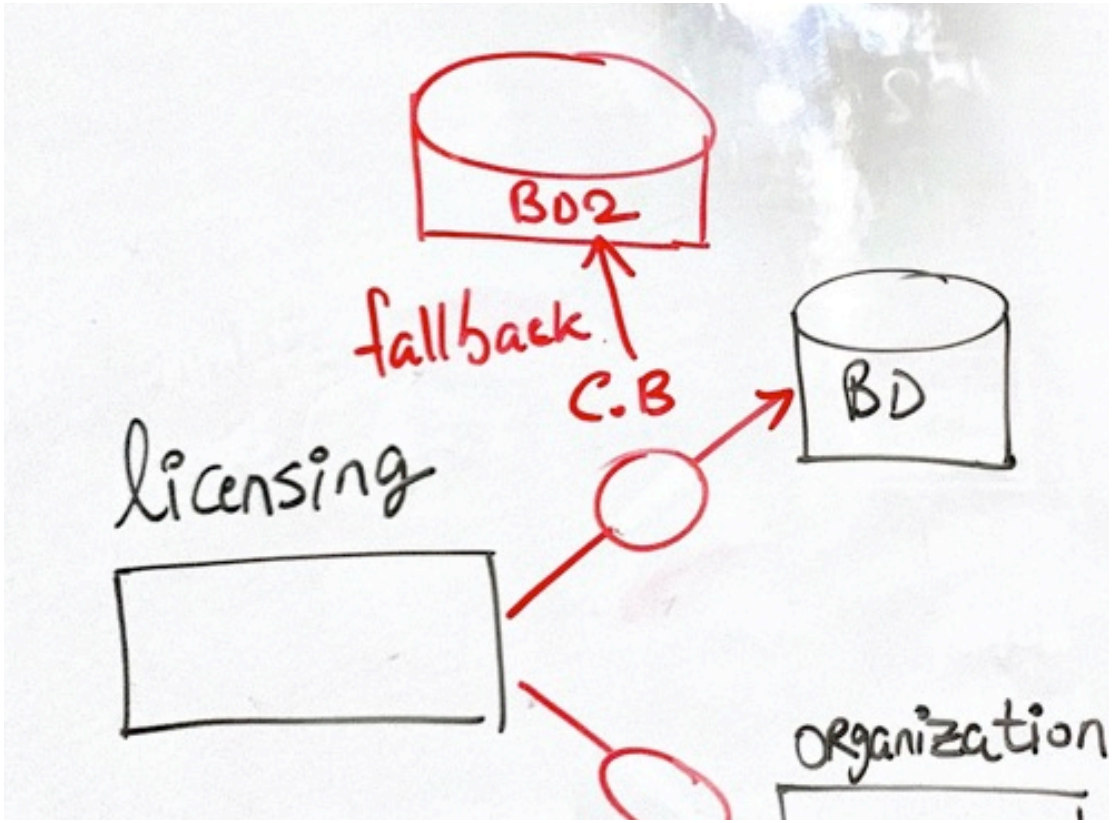
- implementa circuit breakers, fallbacks y bulkhead

1. Circuit breaker



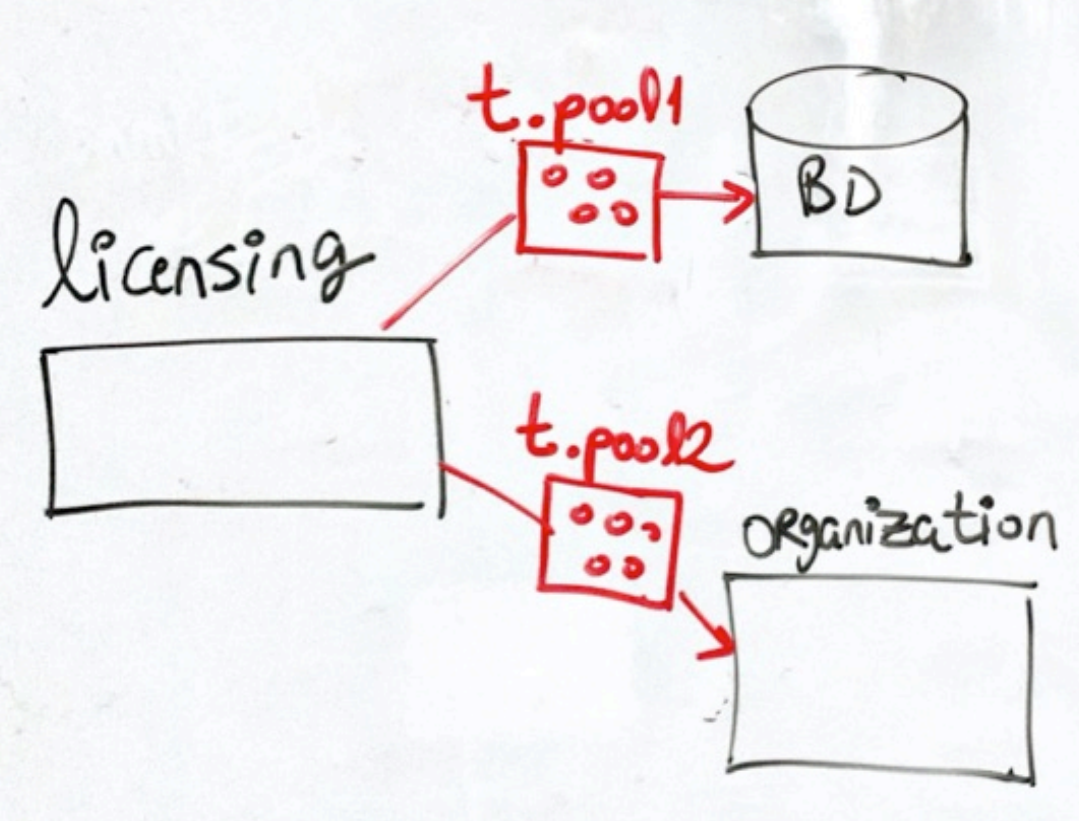
OS ENTERPRISE CONSULTING

2. Fallback



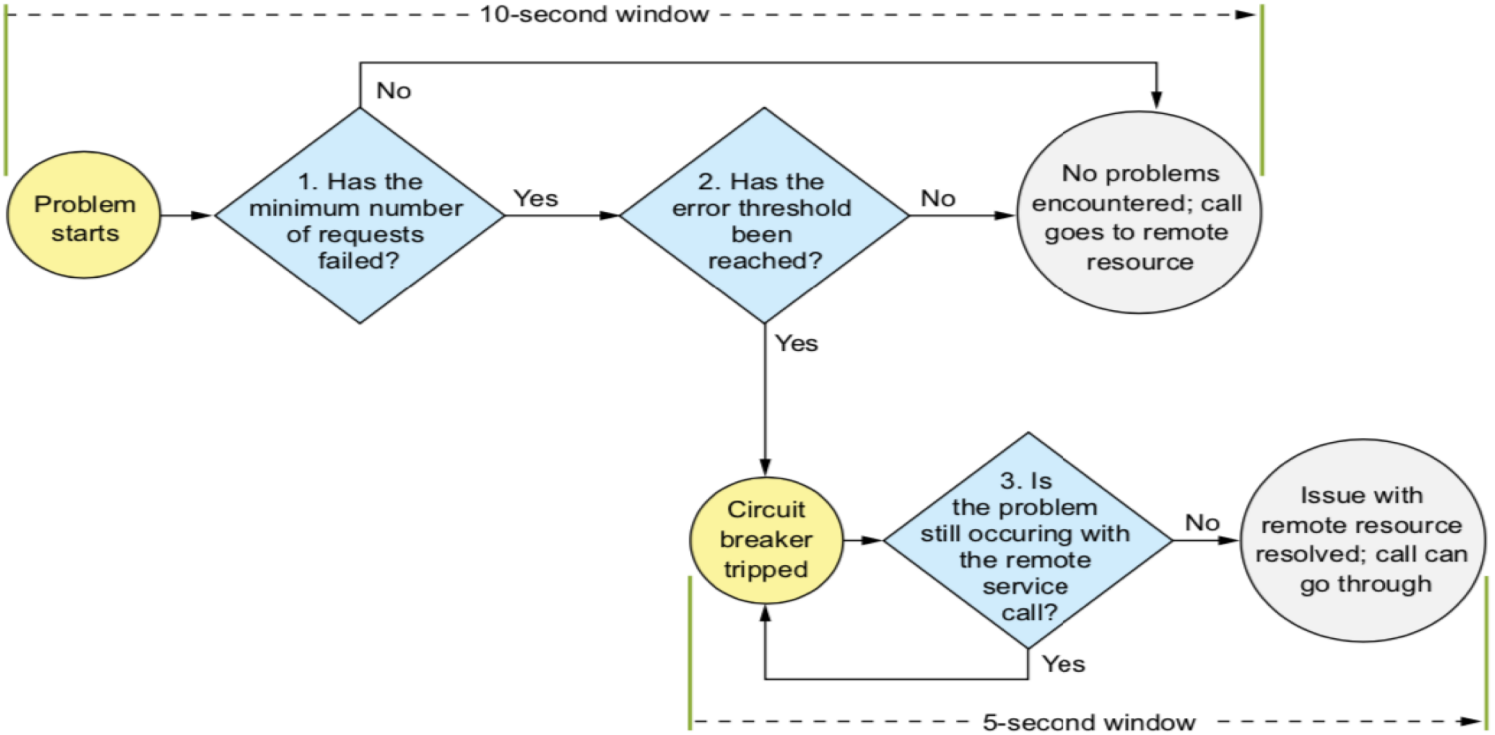
OS ENTERPRISE CONSULTING

Bulkhead



OS ENTERPRISE CONSULTING

Circuit breaker details



- Volume-threshold
- Sleep-Windows-inmiliseconds
- Metrics-rollingstats
- Threshold-percentage

OS ENTERPRISE CONSULTING

HystrixCommand

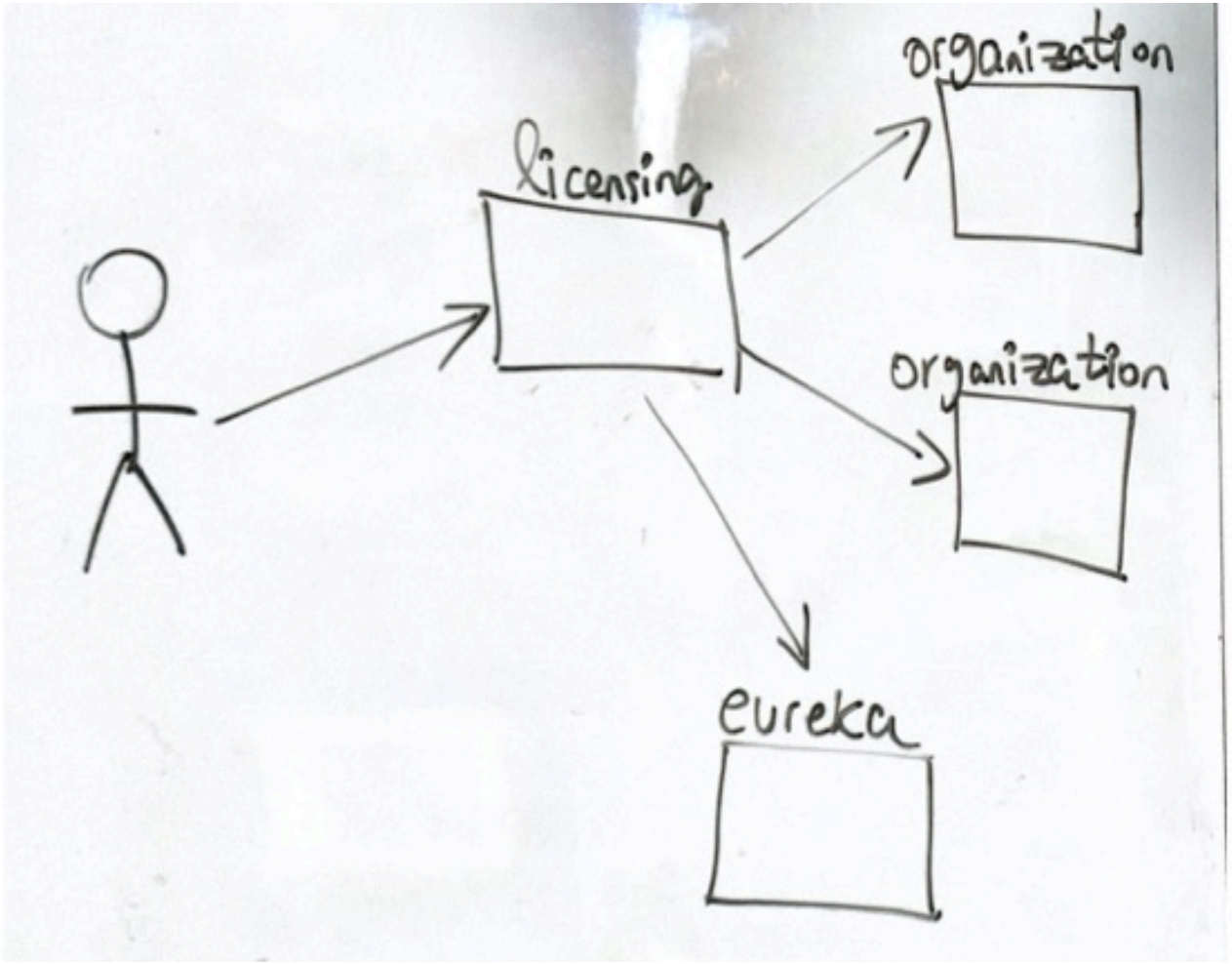
- Personalizar Hystrix

Table 5.1 Configuration Values for `@HystrixCommand` Annotations

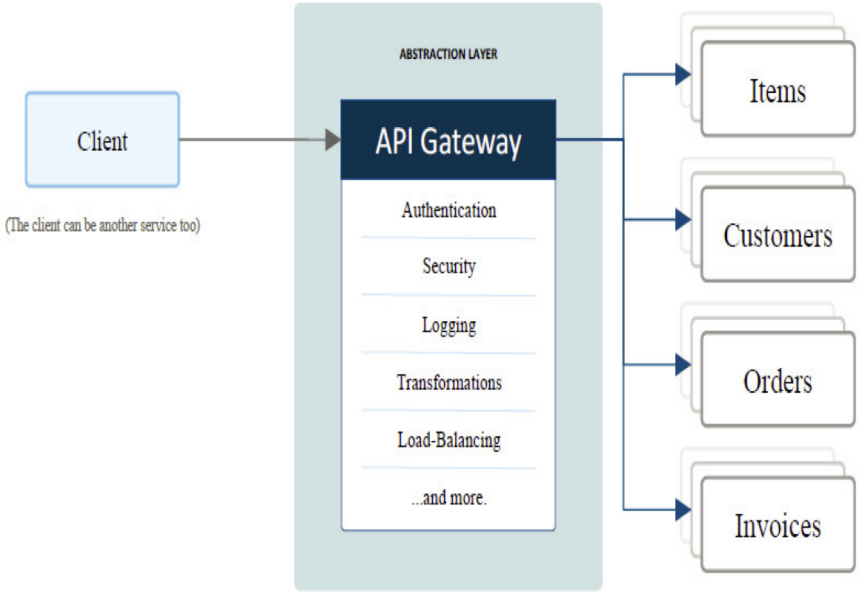
Property Name	Default Value	Description
`fallbackMethod`	None	Identifies the method within the class that will be called if the remote call times out. The callback method must be in the same class as the `@HystrixCommand` annotation and must have the same method signature as the calling class. If no value, an exception will be thrown by Hystrix.
`threadPoolKey`	None	Gives the `@HystrixCommand` a unique name and creates a thread pool that is independent of the default thread pool. If no value is defined, the default Hystrix thread pool will be used.
`threadPoolProperties`	None	Core Hystrix annotation attribute that's used to configure the behavior of a thread pool.
`coreSize`	10	Sets the size of the thread pool.
`maxQueueSize`	-1	Maximum queue size that will set in front of the thread pool. If set to -1, no queue is used and instead Hystrix will block until a thread becomes available for processing.
`circuitBreaker.request-VolumeThreshold`	20	Sets the minimum number of requests that must be processed within the rolling window before Hystrix will even begin examining whether the circuit breaker will be tripped. Note: This value can only be set with the `commandPoolProperties` attribute.
`circuitBreaker.error-ThresholdPercentage`	50	The percentage of failures that must occur within the rolling window before the circuit breaker is tripped. Note: This value can only be set with the `commandPoolProperties` attribute.
`circuitBreaker.sleep-WindowInMilliseconds`	5,000	The number of milliseconds Hystrix will wait before trying a service call after the circuit breaker has been tripped. Note: This value can only be set with the `commandPoolProperties` attribute.
`metricsRollingStats.timeInMilliseconds`	10,000	The number of milliseconds Hystrix will collect and monitor statistics about service calls within a window.
`metricsRollingStats.numBuckets`	10	The number of metrics buckets Hystrix will maintain within its monitoring window. The more buckets within the monitoring window, the lower the level of time Hystrix will monitor for faults within the window.

API Gateway con Netflix Zuul

OS ENTERPRISE CONSULTING


organization
licensing
organization
eureka

API Gateway

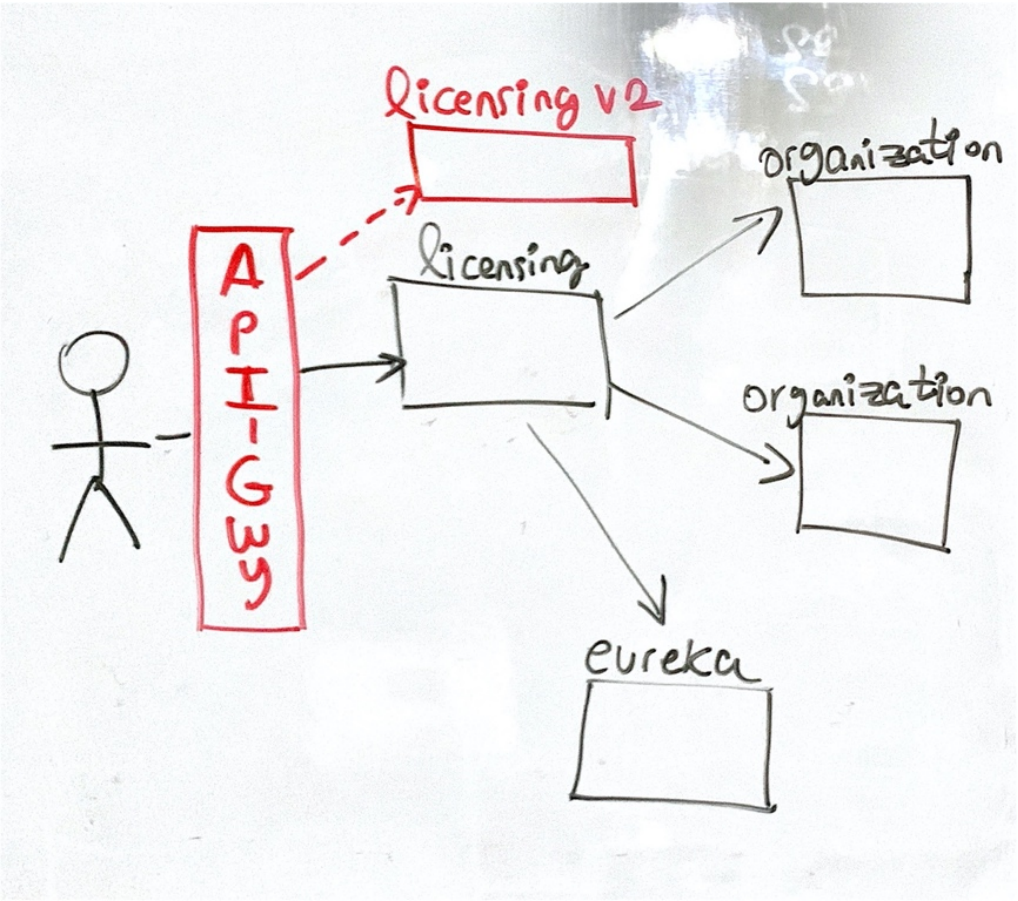


- Intermediario cliente – servicio invocado
  - Unica url
- Determina servicio a invocar
- Policia transito
  - Rutea al servicio-instancia
  - Guardian request ingresan
- Cliente no llama directamente servicio
  - Punto central de aplicacion de reglas
  - Aspectos transversales, 1 solo lugar
  - Libera equipo desarrollo de Aspectos transversales

OS ENTERPRISE CONSULTING

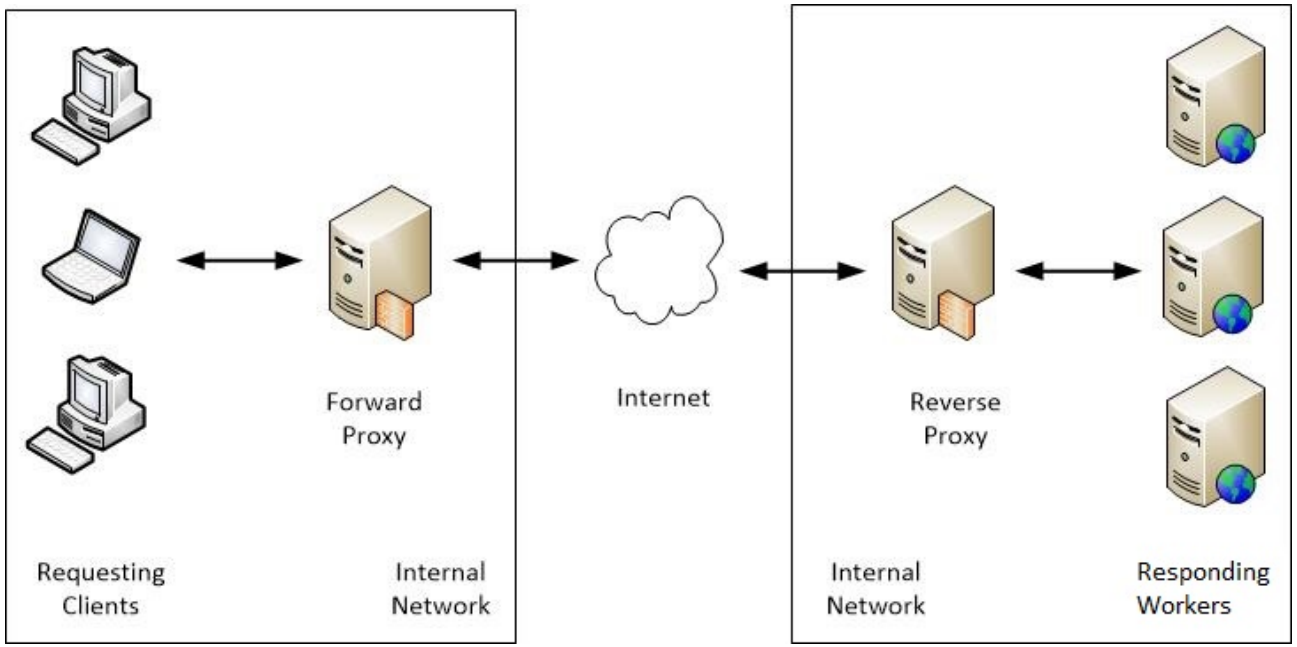
Aspectos transversales

- Ruteo estático
  - Único endpoint para todos los servicios
- Ruteo dinamico
  - Ruteos inteligentes basado clientes
    - Ejemplo: ruteados a una versión diferente servicio
- Autenticación y autorización
  - Punto natural de verificación
- Métricas y logging
  - colectar métricas y escribir logs



OS ENTERPRISE CONSULTING

proxy



Api Gateway es un caso especial de reverse proxy

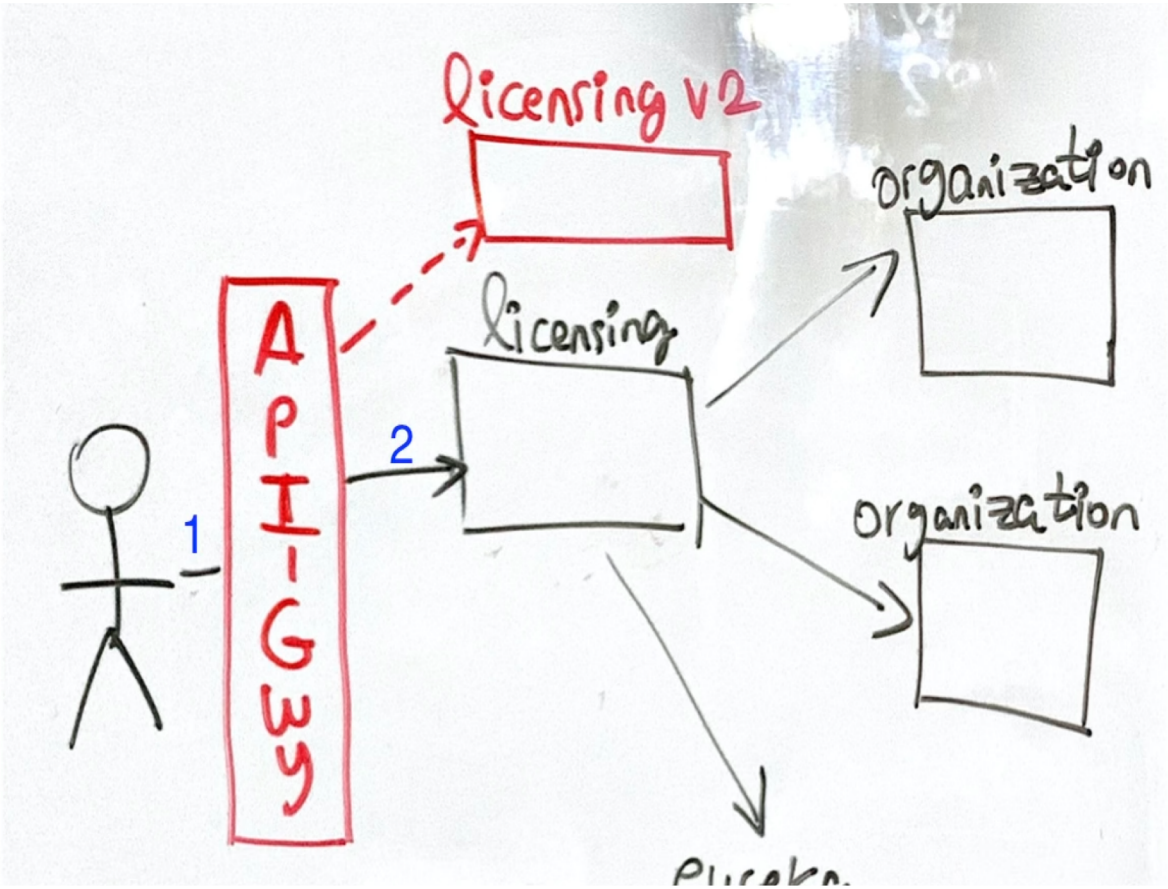
OS ENTERPRISE CONSULTING

Netflix Zuul

- Integración con Spring cloud
- Administra ruteos de todos los servicios
  - Único punto de entrada
- Construye filters: inspeccionan y actúan sobre request

Zuul routes

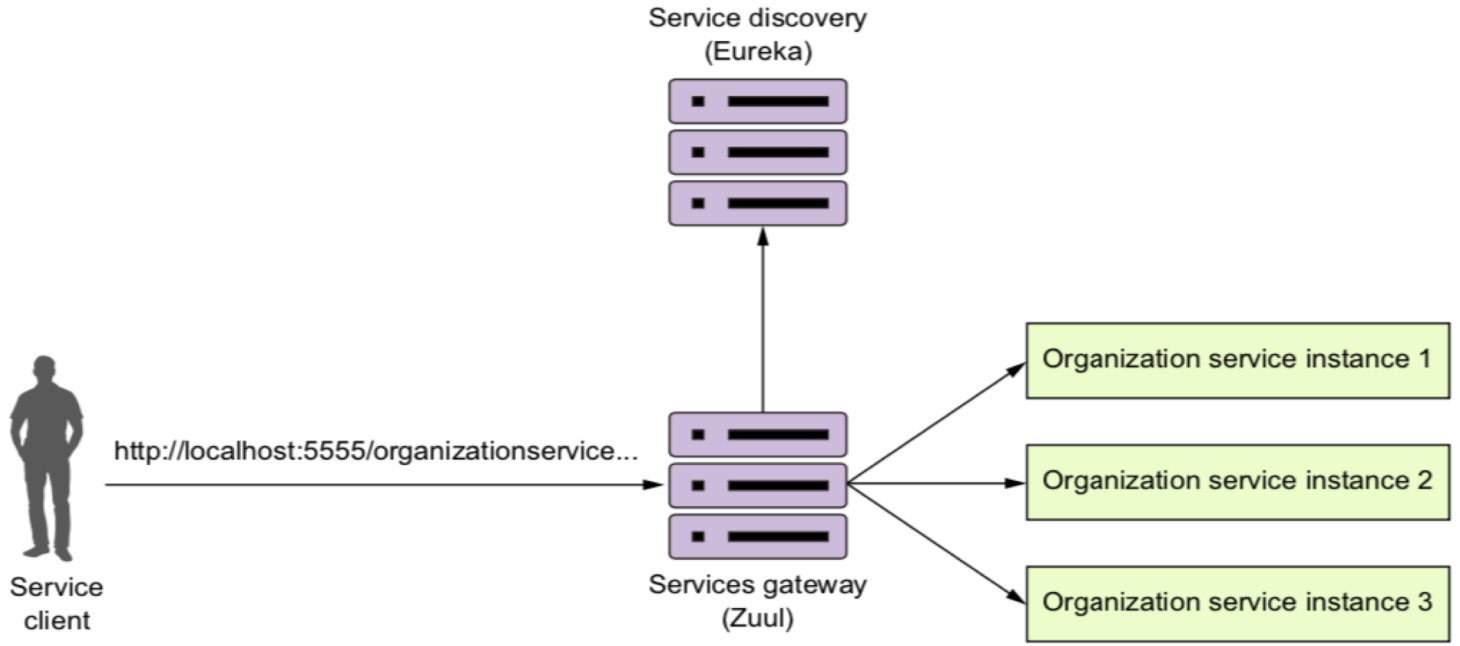
- Zuul debe maperar request y rutear a servicios
- Tipos de ruteo
  - ruteo automático via service Discovery
  - mapeo manual via service Discovery
  - mapeo manual usando url estáticas



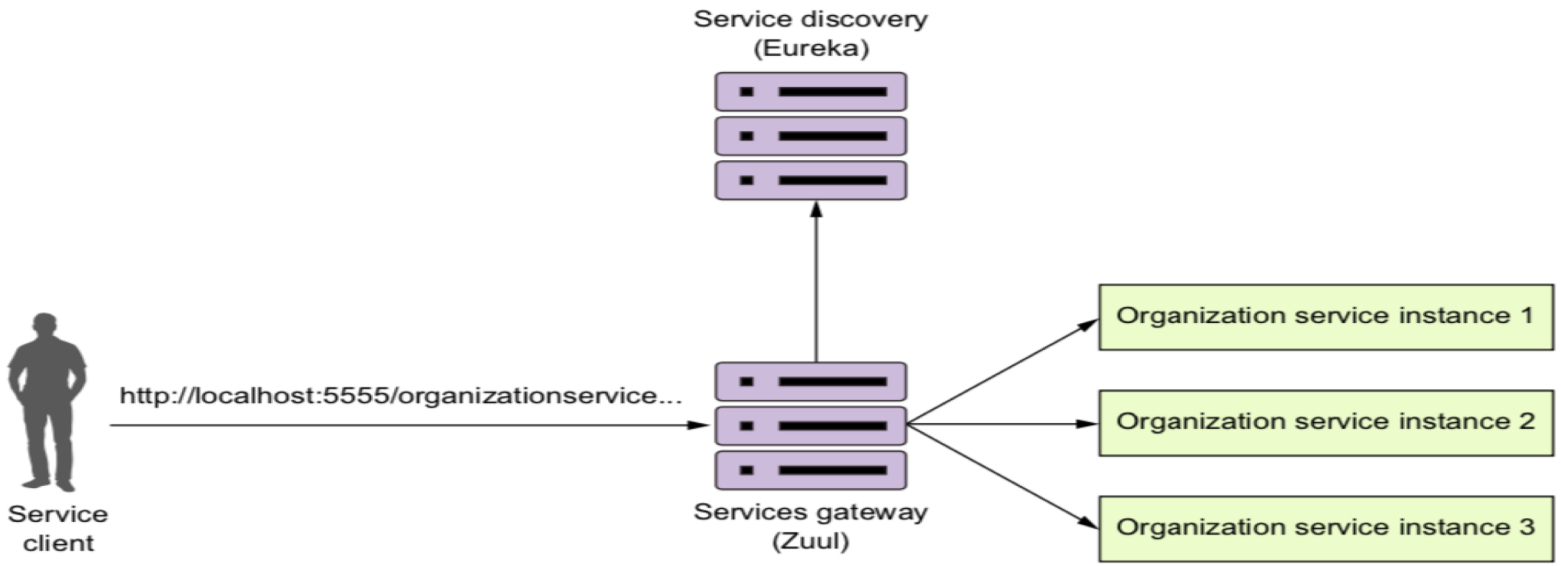
OS ENTERPRISE CONSULTING

Ruteo automático via service Discovery

- Basado en serviceid
- 0 configuración
- Esta por defecto



http://localhost:5555/**organizationservice**/v1/organizations/e254f8c-c442-4ebe-a82a-e2fc1d1ff78a



OS ENTERPRISE CONSULTING

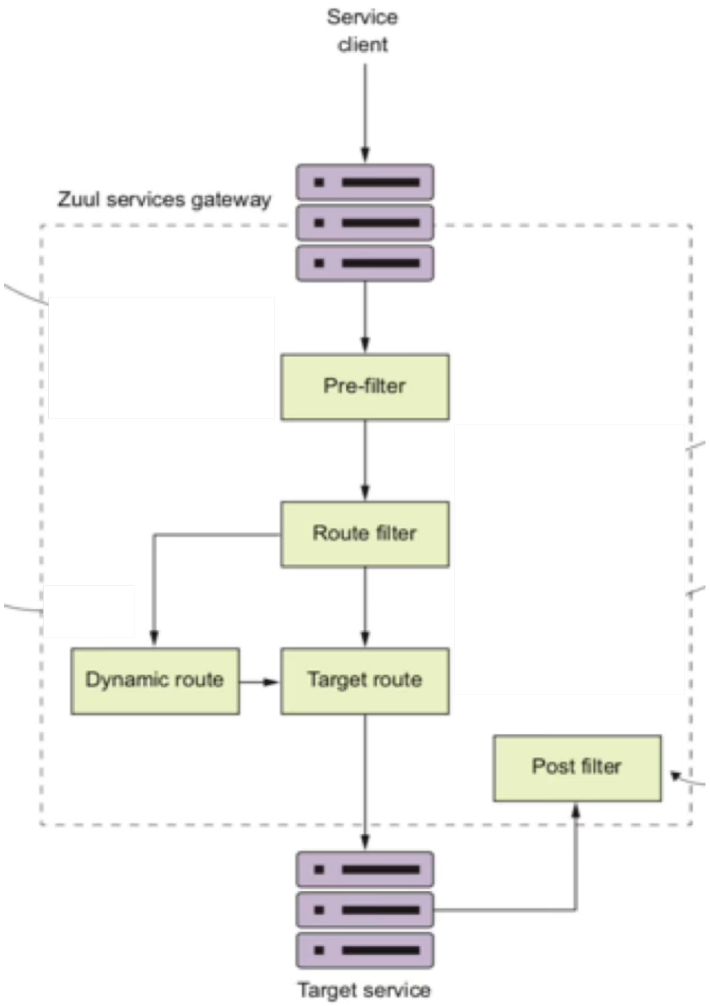
Laboratorio

Zuul filters

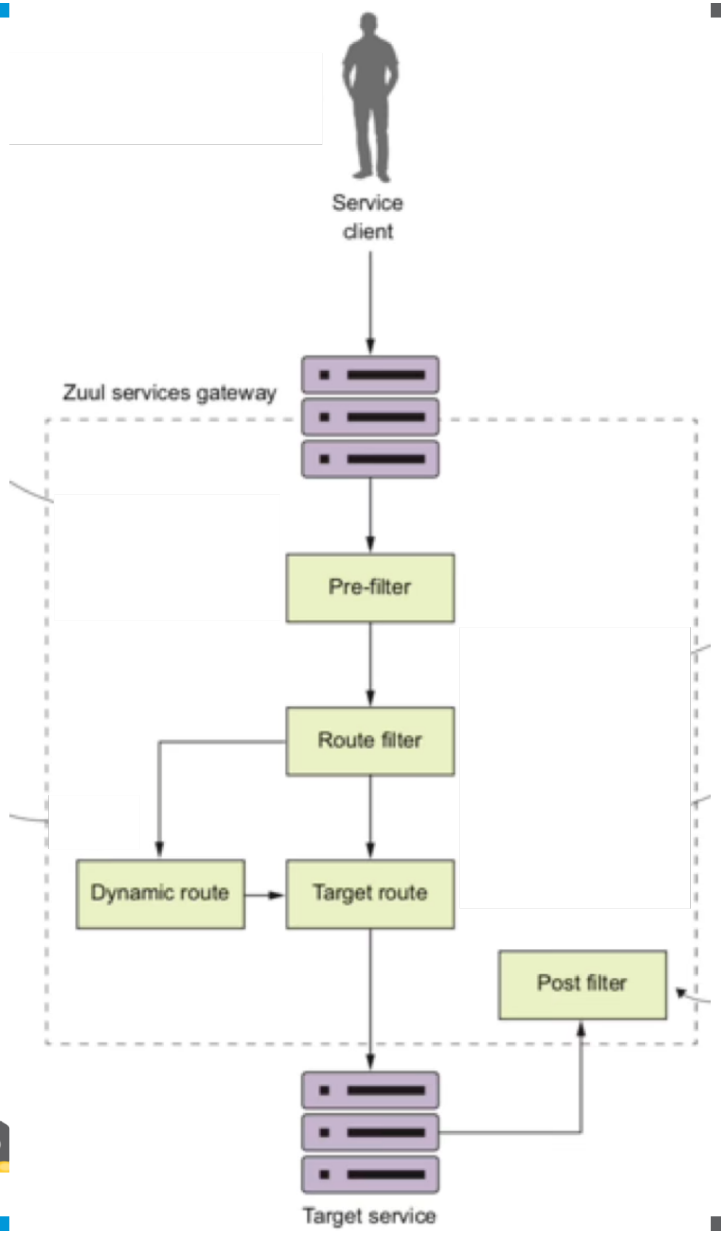
- Zuul es mas que un reverse proxy
- Poder real: lógica aplicada todos los servicios
  - Reglas: seguridad, logging, tracking, etc
  - Aplicados a cada servicio , sin modificar cada servicio
- Zuul soporta filters
- Similares servlet filters:
  - Interceptan el request sin afectar código original
- Servlet filters aplican servicio especifico
  - Zuul filters aplican todos los servicios

Tipos de filters

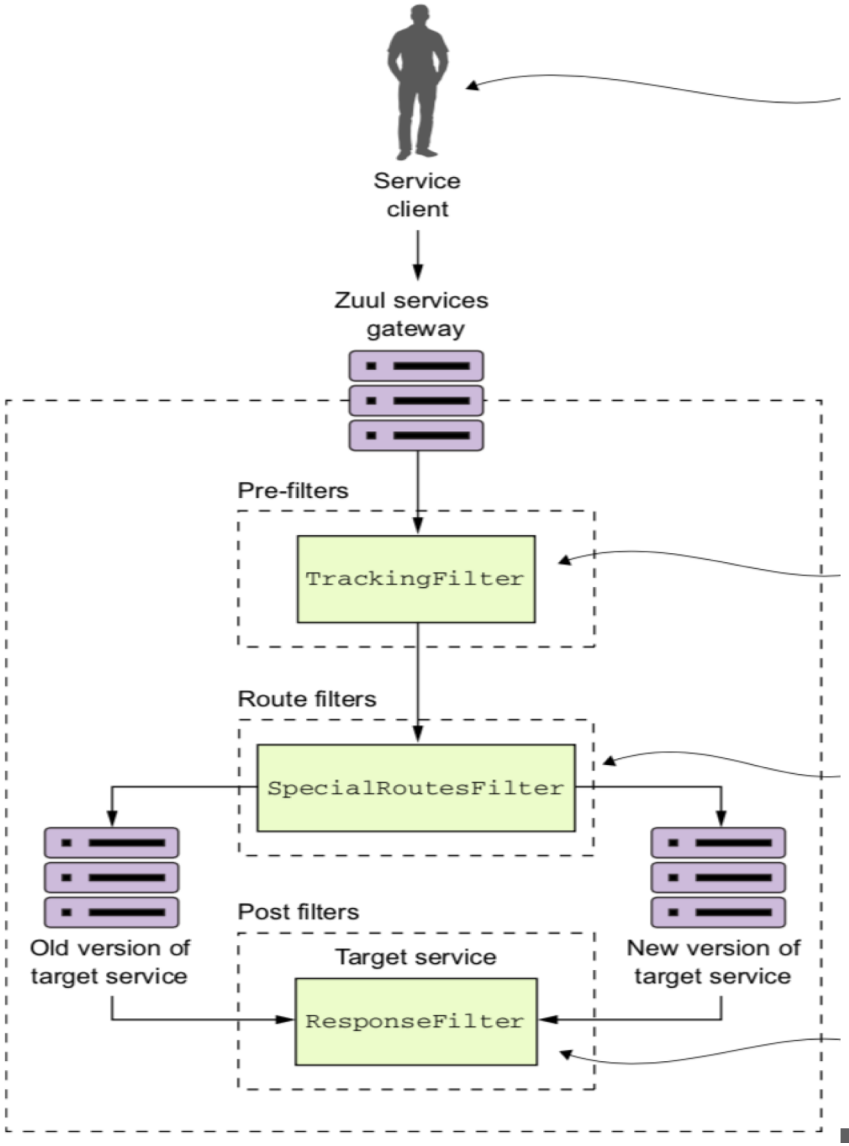
- Pre-filter
  - Antes del recurso destino
  - Formato consistente: incluyan headers,etc
  - Vigilante: cliente autenticado / autorizado
- Post-filters
  - Despues del recurso destino
  - Escribir respuesta log, manejar errores , auditoria, etc
- Route-filters
  - Interceptan llamada antes recurso destino
  - Realiza ruteo dinamico
    - Ejemplo: rutear diferentes versiones del servicio



OS ENTERPRISE CONSULTING



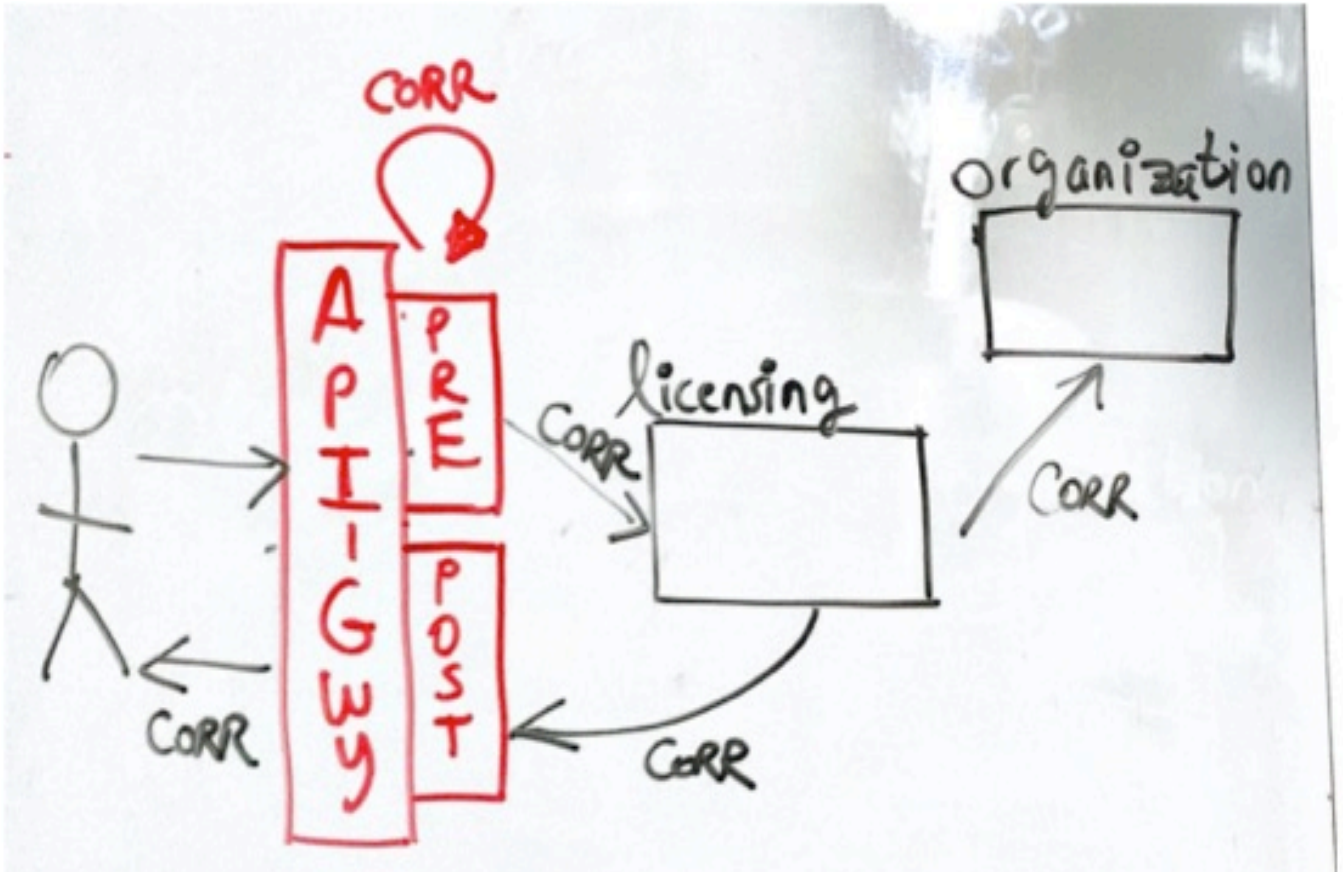
1. Pre-filters
    1. No redirige a otro destino
2. Route-filters
    1. Redirigen a otro destino
    2. Envia request destino
3. Post-filters
    1. Modificar respuesta



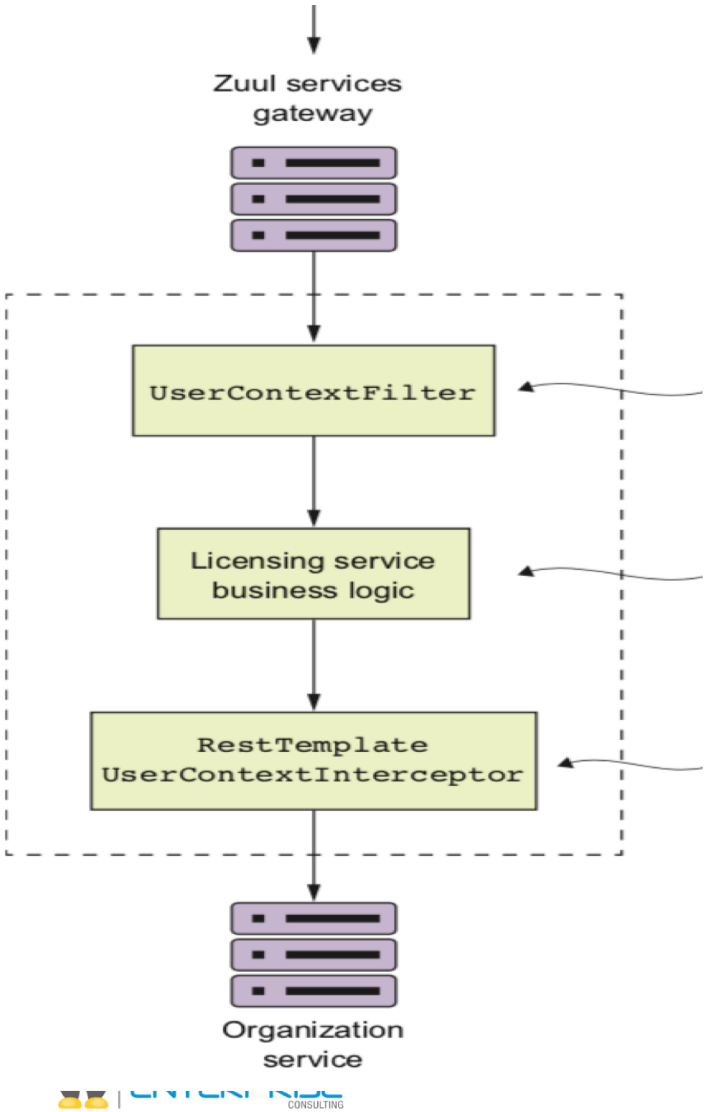
Filters de ejemplo

- Tracking filter
  - Asegura correlation-id
  - Identificador propagado por servicios
  - Tracear eventos entre llamadas
- SpecialRoutesFilter
- Prueba A/B
  - Aleatoriamente 2 versiones servicio.
  - Testeadas antes despliegue
  - Pocos ven nueva version
- ResponseFilter
  - Inyecta correlation-id en respuesta
  - Cliente accede a correlation-id

Propagar correlation-id
Pre y post Filters



Laboratorio Filters



Propagar correlation

Propagacion es comun a todos los microservicios