INFRASTRUCTURE DE PRODUCTION

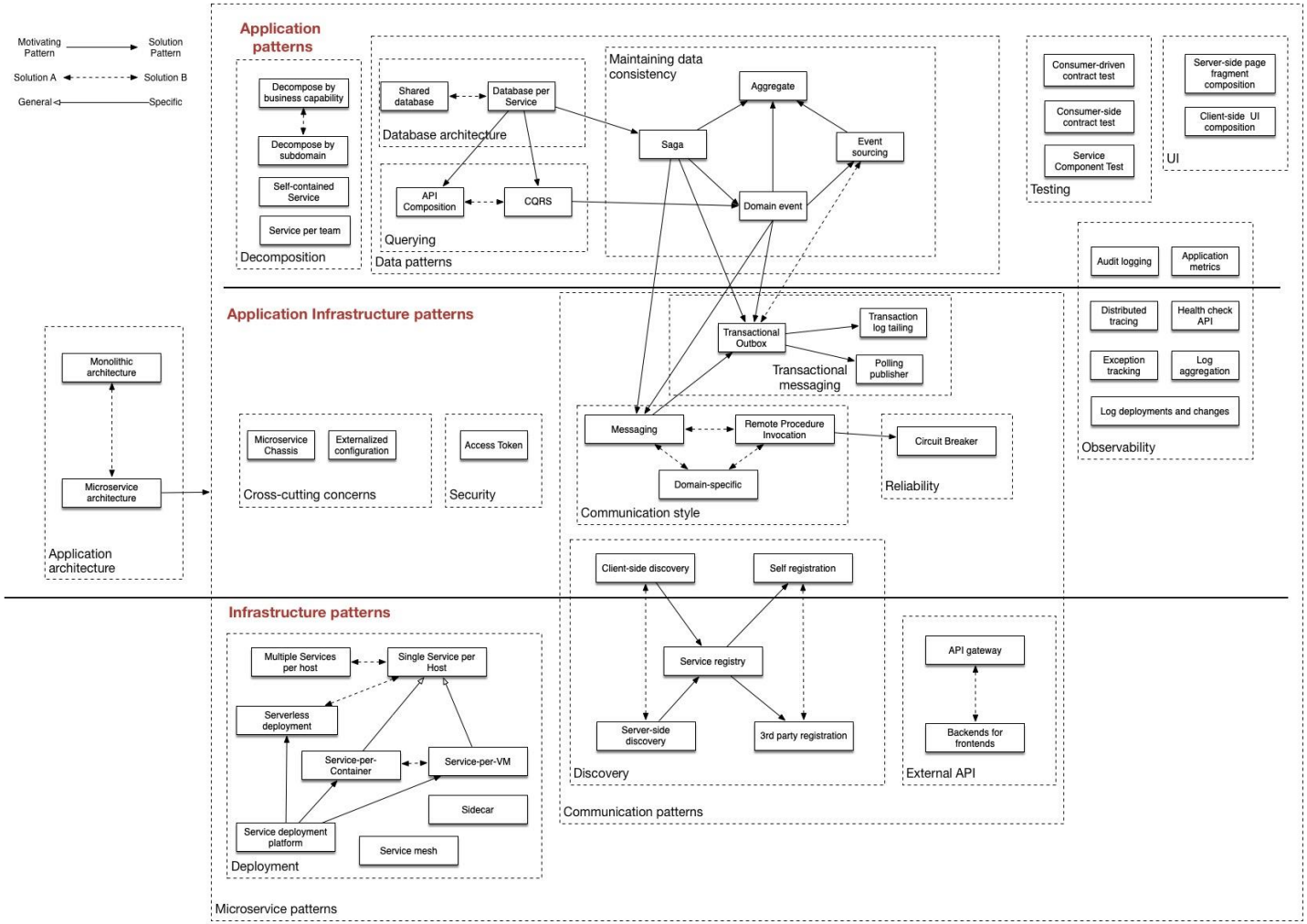
GATEWAY

00.00.2017

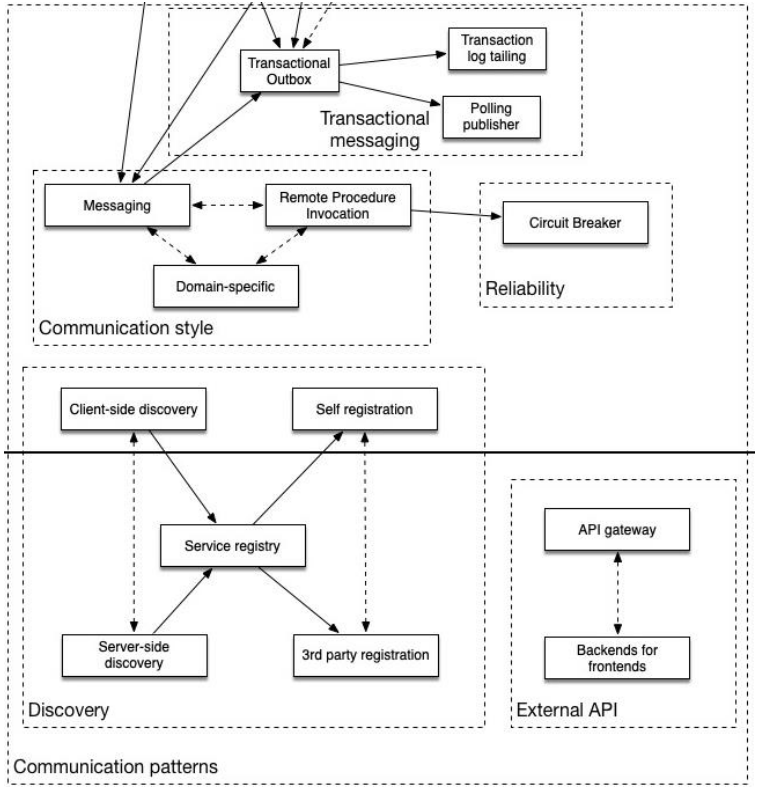
Photo by Diogène Mouton on Unsplash

TABLE DES MATIÈRES

Photo by Martha Dominguez de Gouveia on Unsplash



- Discovery
  - Service Registry
  - Server-side discovery
  - Client-side discovery
  - Self Registration
  - 3rdParty Registration
- External API
  - API Gateway
  - Backend for front end
- Communication Style
- Reliability

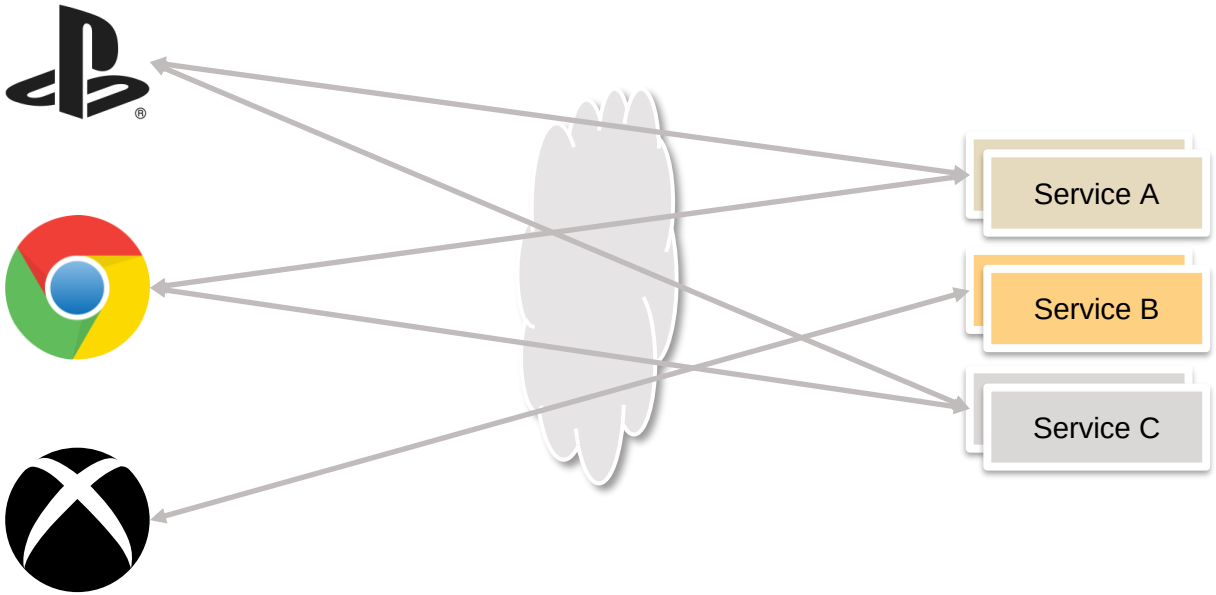


1

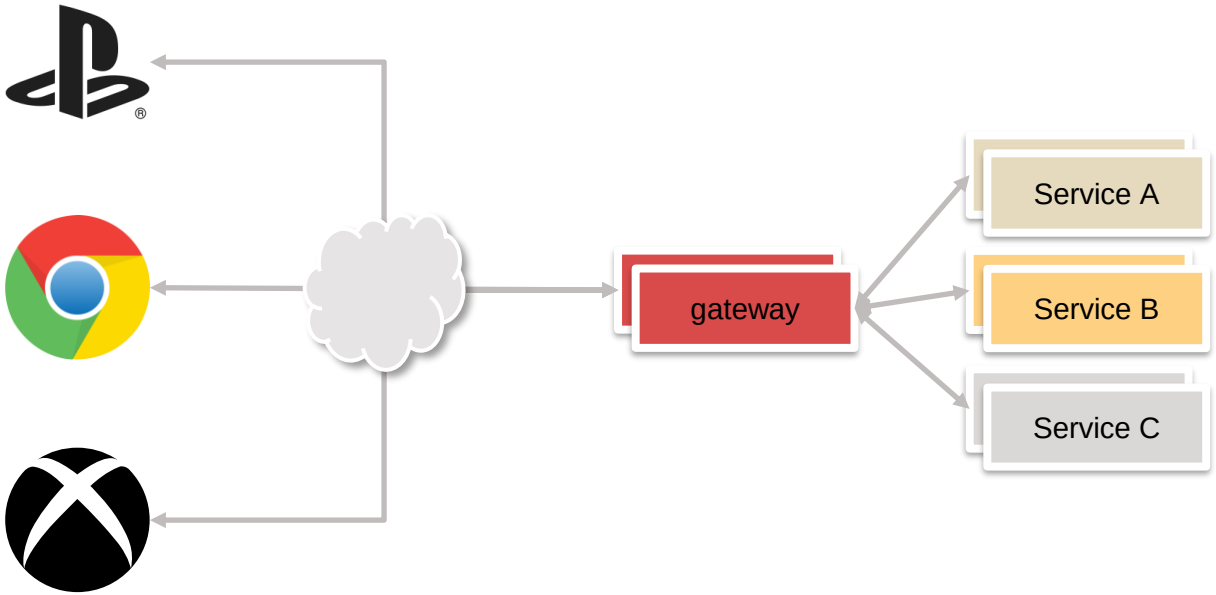
FACILITATEUR

Photo by Luca Bravo on Unsplash

DE INTERNET AU SERVICE DANS LE CLOUD



DE INTERNET AU SERVICE DANS LE CLOUD



RESPONSABILITÉ

- **Routage**
- **Routage précis**
- **Routage canary**
- **Sécurité**
- **Résilience**
- **Monitoring**
- **flexibilité**

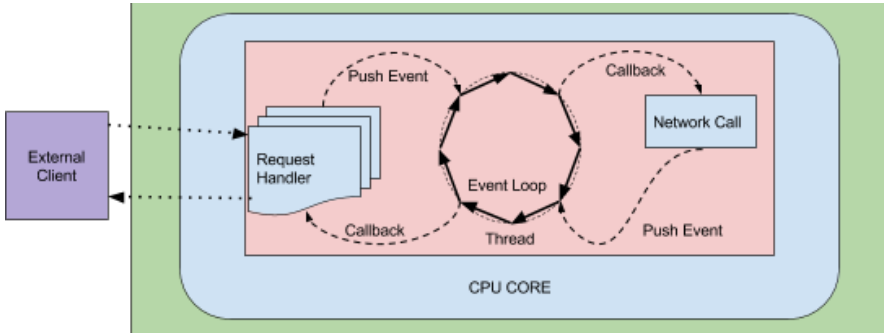
UNE GATEWAY

- Intégré dans les microservices
- Routage logique dynamique
- Loadbalancé
- Centré sur la disponibilité
- Ajout de protection

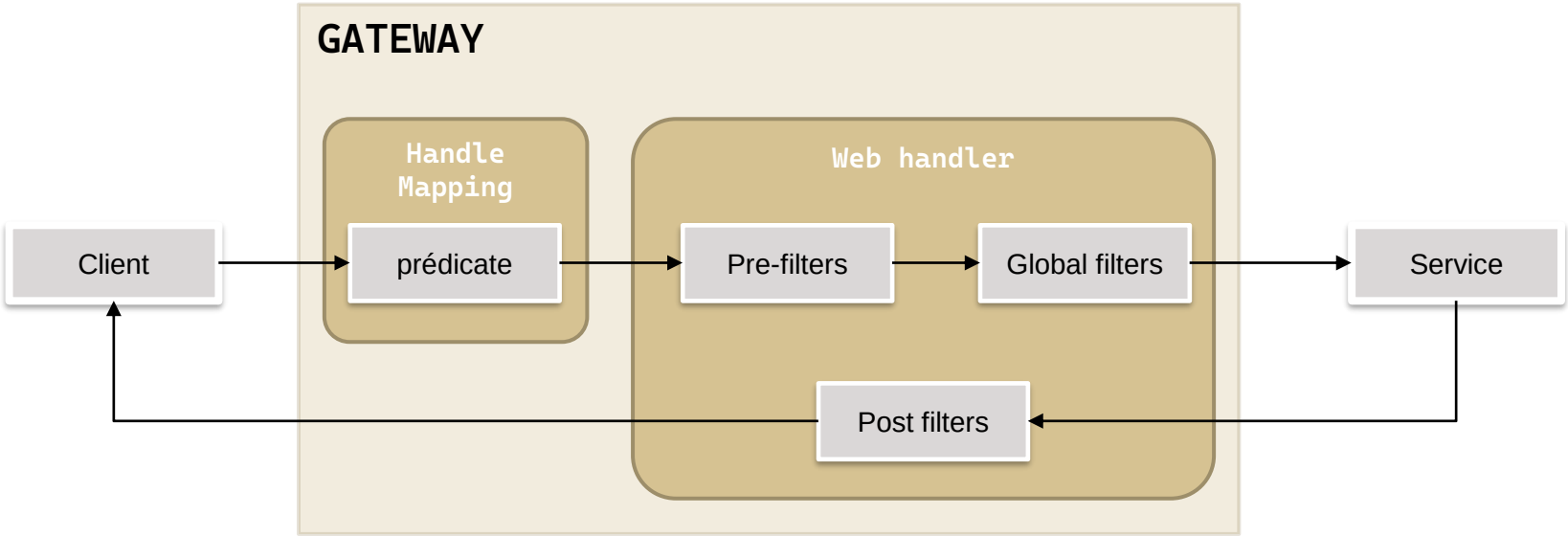
- Basé sur spring 5, reactor et boot 2
- Les routes sont configurables en properties, et en java
- Routable sur les path, host, header, parameters ou toutes autres éléments de la requête
- Filtre
- Réécriture des path
- Rate limiting
- Resilient4j
- Spring cloud loadbalancerClient

- Non blocking IO basé sur RxNetty
- Support du HTTP/2 et des websocket
- Fonctionnalités propriétaires
- Les routes sont configurable en properties
- Filtre
- Resilient4j
- Loadbalancer ribbon

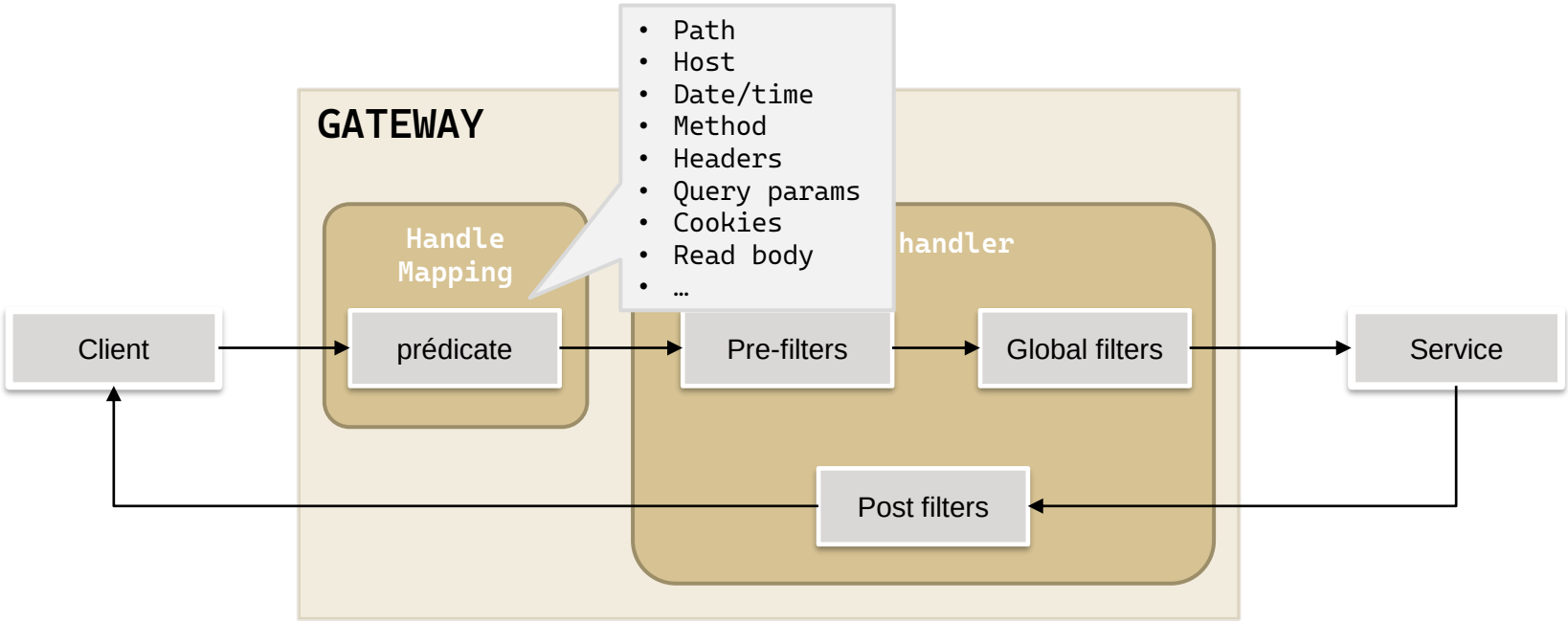
SPRING CLOUD GATEWAY NON BLOCKING IO



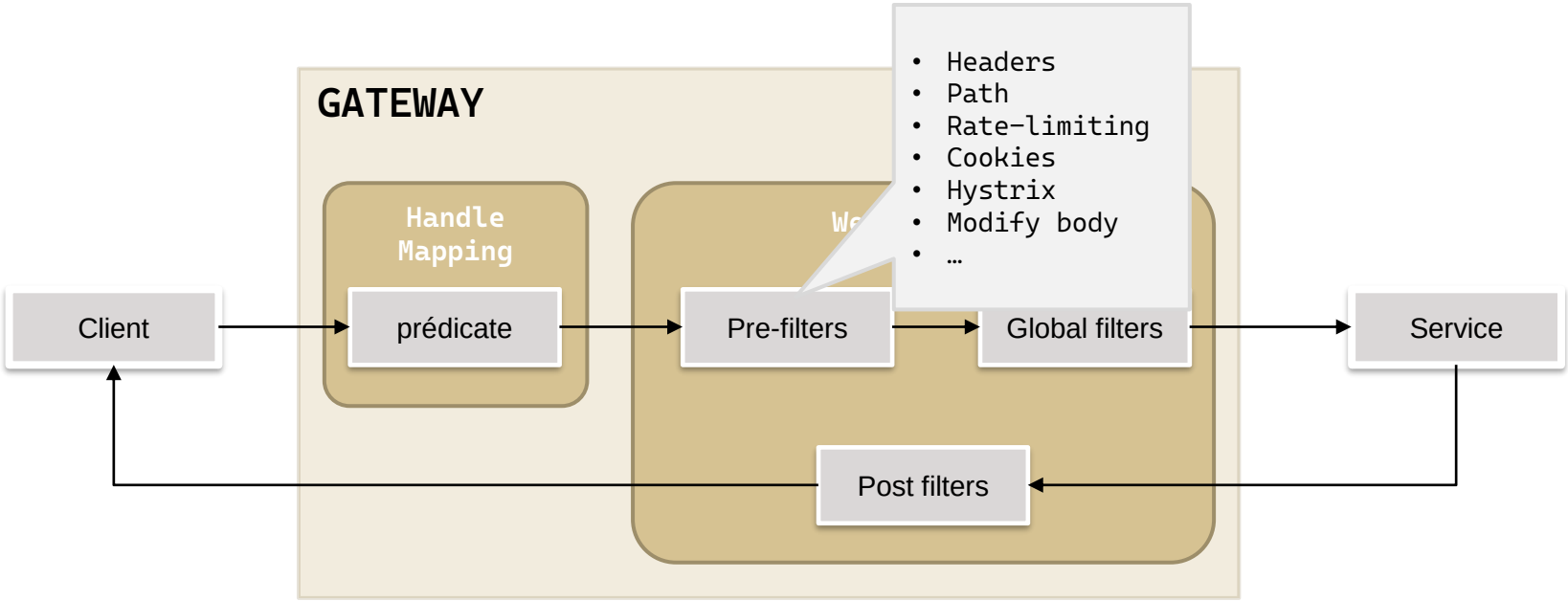
PARCOURS DU FLUX DE LA GATEWAY



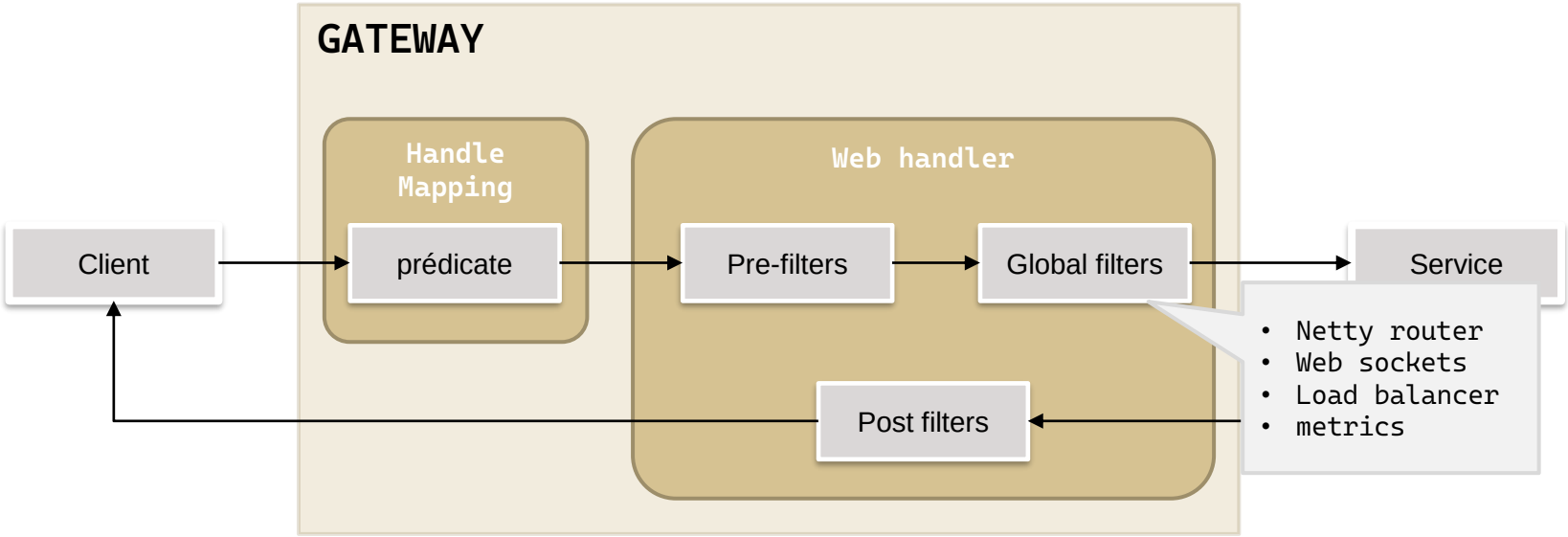
PARCOURS DU FLUX DE LA GATEWAY



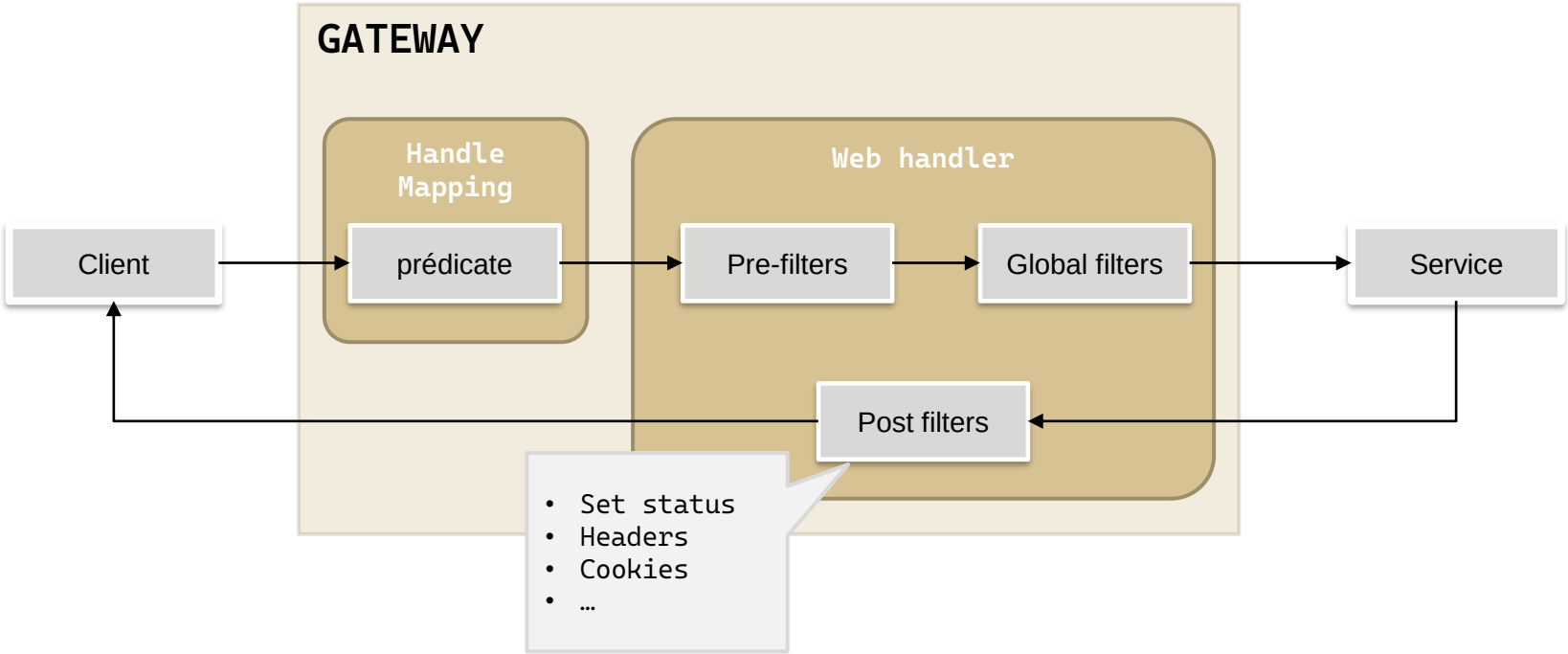
PARCOURS DU FLUX DE LA GATEWAY



PARCOURS DU FLUX DE LA GATEWAY



PARCOURS DU FLUX DE LA GATEWAY



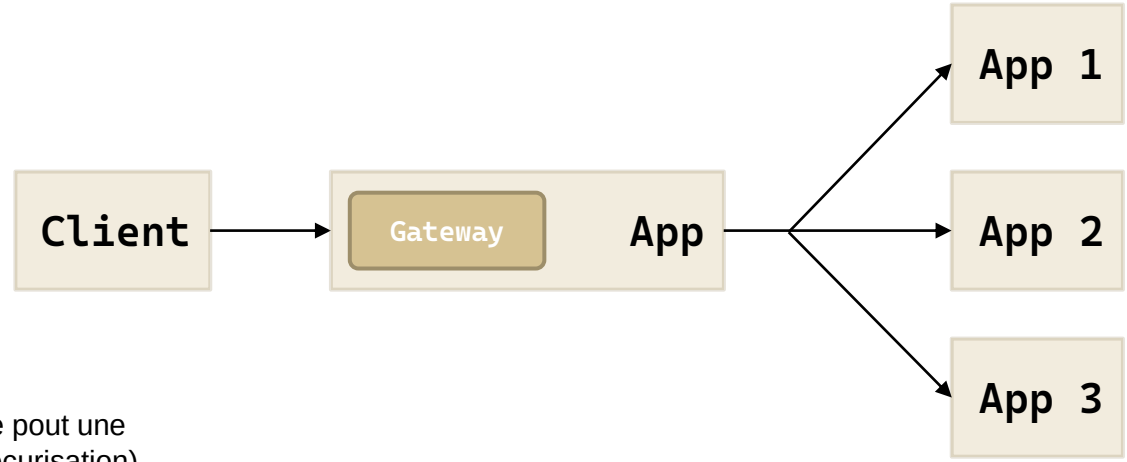
CONSIDÉRATION DE CONCEPTION

- **Embedded**

- **Cas d'utilisation**
  - Quand la latence est essentiel (pas d'ajout de temps réseau)

CONSIDÉRATION DE CONCEPTION

- **Embedded Facade**

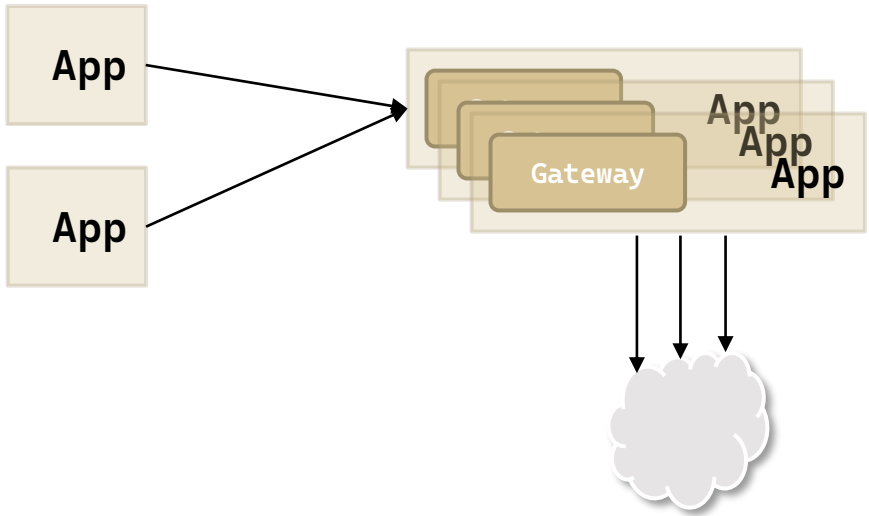


- **Cas d'utilisation**
  - Pour exposer des services via un point unique pout une application web (réduit la complexité des la sécurisation)

CONSIDÉRATION DE CONCEPTION

- **Embedded**

- **Cas d'utilisation**
  - Gestion fine des accès pour un paiement à l'usage ou pour un backend fragile



CONSIDÉRATION DE CONCEPTION

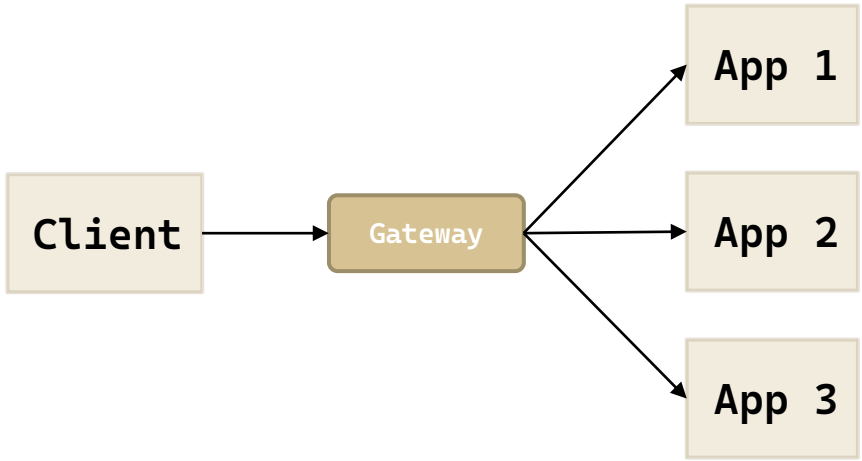
Facade

- **Cas d'utilisation**
  - Pour ne pas exposer les accès en direct sur une App
  - Pour gérer les ruptures protocolaires et de langage

CONSIDÉRATION DE CONCEPTION

- **Facade**

- **Cas d'utilisation**
  - Présenter une api unique pour le client



```java
@Slf4j
public class SimpleFilter implements GatewayFilter {

    @Override
    public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {
        log.info("BEFORE");
        return chain.filter(exchange)
                    .then(Mono.fromRunnable(() -> {
                                log.info("AFTER");
                    }));
    }

}
```

```yaml
spring:
  cloud:
    gateway:
    - id: foo_route
    uri: lb://foo
    predicates:
    - Host=**.foo.org
    - Path=/headers
    - Method=GET
    - Header=X-Request-Id, \d+
    - Query=foo, ba.
    - Query=baz
    - Cookie=chocolate, ch.p
    - After=1900-01-20T17:42:47.789-
07:00[America/Denver]
    filters:
    - AddRequestHeader=X-Request-Foo, Bar
    - AddResponseHeader=X-Response-Foo, Bar
    - Hystrix=foo
    - SecureHeaders
    - RewritePath=/foo/(?<segment>.*), /$\{segment}
```

```java
@Bean
public RouteLocator customRouteLocator(RouteLocatorBuilder builder, ThrottleGatewayFilterFactory throttle) {
    return builder.routes()
                    .route(r -> r.host("**.abc.org").and().path("/image/png")
                        .filters(f -> f.addResponseHeader("X-TestHeader", "foobar"))
                        .uri("http://httpbin.org:80")
                    )
                    .route(r -> r.path("/image/webp")
                        .filters(f -> f.addResponseHeader("X-AnotherHeader", "baz"))
                        .uri("http://httpbin.org:80")
                    )
                    .route(r -> r.order(-1)
                        .host("**.throttle.org").and().path("/get")
                        .filters(f -> f.filter(throttle.apply(1, 1, 10, TimeUnit.SECONDS)))
                        .uri("http://httpbin.org:80")
                    )
                    .build();
}
```

TP

Photo by Annie Spratt on Unsplash