

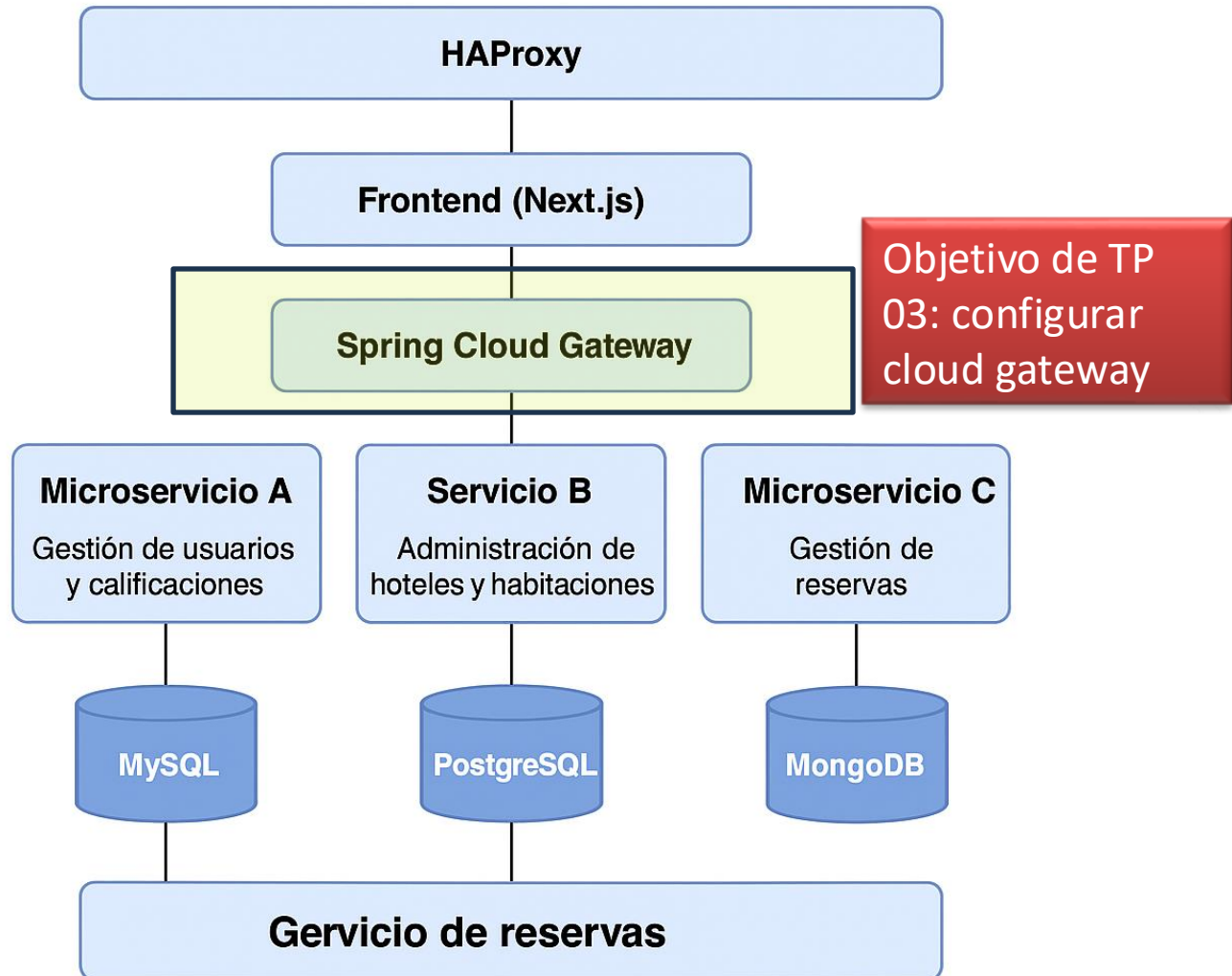
Universidad Tecnológica Nacional  
Facultad Regional Santa Fe

Departamento Ingeniería en sistemas de Información

Arquitectura en la nube

# **TRABAJO PRACTICO DAN - 03**

# Arquitectura



# Configurar Spring Cloud Gateway

- Configuraremos Spring Cloud Gateway Server Web MVC
  - <https://docs.spring.io/spring-cloud-gateway/reference/spring-cloud-gateway-server-webmvc.html>
- Existe otra alternativa que es configurarlo reactive como se indica aquí
  - <https://docs.spring.io/spring-cloud-gateway/reference/spring-cloud-gateway-server-webflux.html>
- El gateway permitirá encapsular las llamadas a nuestro backend en un único punto.

## Agregar el gateway

- El gateway se agrego en “common/dan-spring-gateway”.
- Para la creación desde spring inicializr agregaremos la dependencia

### Gateway **SPRING CLOUD ROUTING**

Provides a simple, yet effective way to route to APIs in Servlet-based applications. Provides cross-cutting concerns to those APIs such as security, monitoring/metrics, and resiliency.

### Reactive Gateway **SPRING CLOUD ROUTING**

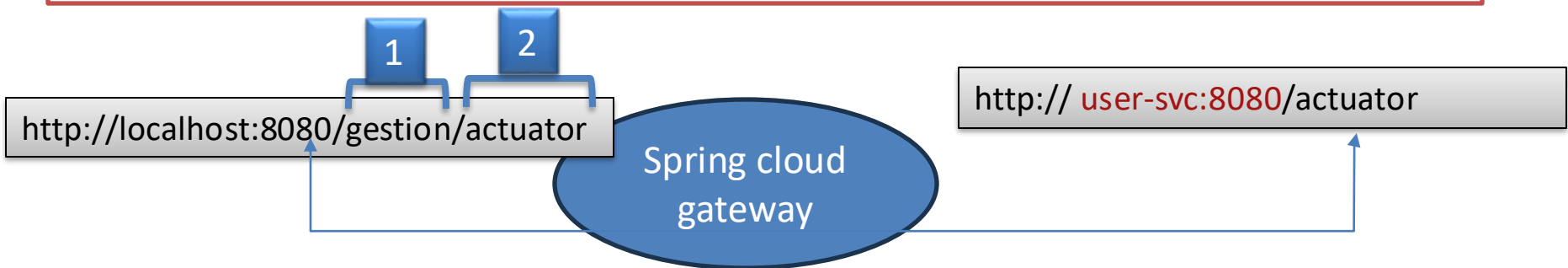
Provides a simple, yet effective way to route to APIs in reactive applications. Provides cross-cutting concerns to those APIs such as security, monitoring/metrics, and resiliency.



# Configurar el gateway

- Para cada ruta configuraremos 1
  - Id: un string arbitrario para identificar la ruta
  - Uri: la url base a donde redigir las peticiones
  - Predicates: analiza las URL recibidas y las que coinciden con la expresión regular entonces son reescritas y reenviadas
  - Filter: es una transformación que se le realiza a la URL recibida. En este caso le eliminaremos solo el primer segmento (<https://docs.spring.io/spring-cloud-gateway/reference/spring-cloud-gateway-server-webmvc/filters/stripprefix.html>)

```
spring.cloud.gateway.server.webmvc.routes[0].id=user-svc  
spring.cloud.gateway.server.webmvc.routes[0].uri=http://user-svc:8080  
spring.cloud.gateway.server.webmvc.routes[0].predicates[0]=Path=/users/**  
spring.cloud.gateway.server.webmvc.routes[0].filters[0]=StripPrefix=1
```



De las dos partes de la URL se elimina la primera (`gestion`) y se pasa la segunda (`actuator`)

# Configurar el gateway en docker-compose

- Agregar la configuracion a docker-compose.yml

```
dan-spring-gateway:  
  build:  
    context: ../common/dan-spring-gateway  
  container_name: dan-spring-gateway  
  restart: unless-stopped  
  ports:  
    - "8080:8080"  
  depends_on:  
    - user-svc  
    - reservas-svc  
    - gestion-svc  
  environment:  
    SPRING_PROFILES_ACTIVE: default
```

# Probar el funcionamiento

- Recompilar con maven todos los proyectos y levantar los contenedores docker.
- Actualizar la URL de postman para pasar a través del gateway

The screenshot shows the Postman interface for a POST request to the endpoint `http://localhost:8080/gestion/hoteles`. The request body is a JSON object with the following data:

```
{
  "nombre": "Hotel Test 3",
  "cuit": "20123456789",
  "domicilio": "Calle 123",
  "latitud": -31.4167,
  "longitud": -64.1833,
  "telefono": "3511234567",
  "correoContacto": "contacto@hotel.com",
  "categoria": 3
}
```

The response is a 200 OK status with a response time of 440 ms and a body size of 373 B. The response body is a JSON object:

```
{
  "id": 3,
  "nombre": "Hotel Test 3",
  "cuit": "20123456789",
  "domicilio": "Calle 123",
  "latitud": -31.4167,
  "longitud": -64.1833,
  "telefono": "3511234567",
  "correoContacto": "contacto@hotel.com",
}
```