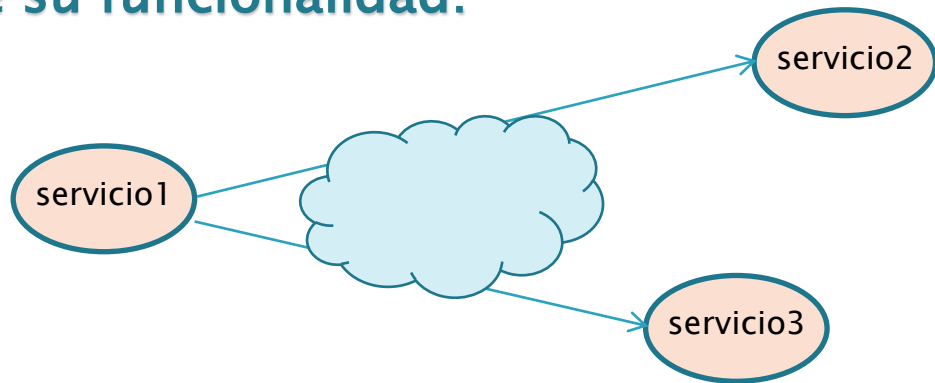


Interacción entre microservicios



Escenario

➤ En muchos escenarios, un microservicio necesita llamar a otro servicio (puede ser o no un microservicio) REST, como parte de su funcionalidad:



➤ Además de las ya conocidas RestTemplate y WebClient, en Spring 6.1 (Spring Boot 3.2), se ha incorporado la interfaz RestClient para llamar a un servicio REST desde otra aplicación Spring

La interfaz RestClient

- Forma parte del módulo Web MVC de Spring, incluido en el starter Web, por lo que no habrá que añadir ningún otro starter adicional.

```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-web</artifactId>  
</dependency>
```

➤ Características:

- Muy similar a WebClient
- Estilo de programación funcional
- Exclusivamente para peticiones síncronas

Creación de un objeto RestClient

➤ Definimos un método en la clase de configuración para su creación mediante el método estático *create()* de la propia interfaz:

```
@Bean  
public RestClient getRestClient() {  
    return RestClient.create();  
}
```

➤ Para inyectarlo en una variable se emplea la anotación *@Autowired* de Spring:

```
@Autowired  
RestClient restClient;
```

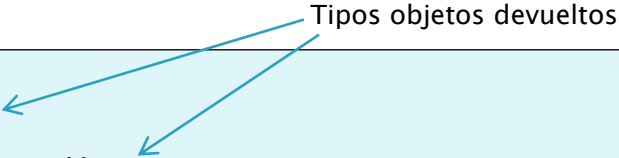
Configuración método de petición

- RestClient proporciona una serie de métodos para establecer el método de petición Http.
- Cada método devuelve un objeto para continuar configurando la petición:
 - RequestHeadersUriSpec *get()*
 - RequestHeadersUriSpec *delete()*
 - RequestBodyUriSpec *post()*
 - RequestBodyUriSpec *put()*

Dirección de la petición

➤ A partir de los objetos devueltos por los métodos anteriores se puede configurar la dirección de la petición mediante el método *uri()*:

restClient
.get() //RequestHeadersUriSpec
.uri("http://localhost:8080/cursos") //RequestHeadersSpec



restClient
.get() //RequestHeadersUriSpec
.uri("http://localhost:8080/curso/{id}",20) //RequestHeadersSpec

restClient
.post() //RequestBodyUriSpec
.uri("http://localhost:8080/curso/alta") //RequestBodySpec

Cuerpo de la petición

- En peticiones POST y PUT se puede enviar un objeto en la petición, habitualmente en formato JSON.
- Se configura a través de los métodos *contentType()* y *body* de *RequestBodySpec*:

```
webClient
    .post() //RequestBodyUriSpec
    .uri("http://localhost:8080/curso/alta") //RequestBodySpec
    .contentType(MediaType.APPLICATION_JSON) //RequestBodySpec
    .body(new Curso(...)) //RequestHeadersSpec
```

- Opcionalmente, se pueden establecer encabezados a través del método *header()* de *RequestHeadersSpec*

Lanzamiento de la petición

- Una vez configurada la uri, encabezados y cuerpo, se procede a lanzar la petición a través del método *retrieve()* de RequestHeadersSpec.
- El método devuelve un ResponseSpec para definir la forma en la que se extraerá la respuesta

```
restClient  
    .get() //RequestHeadersUriSpec  
    .uri("http://localhost:8080/curso/{id}",20) //RequestHeadersSpec  
    .retrieve() //ResponseSpec
```


Extracción de la respuesta

- Mediante *body()* de `ResponseSpec` se extrae directamente el cuerpo de la respuesta, indicando el tipo al que debe convertirse dicho cuerpo.

```
Curso curso=restClient
    .get() //RequestHeadersUriSpec
    .uri("http://localhost:8080/curso/{id}",20) //RequestHeadersSpec
    .retrieve() //ResponseSpec
    .body (Curso.class); //Curso
```

- Otros métodos permiten obtener la respuesta como un `ResponseEntity`:

- `ResponseEntity<Void> toBodilessEntity()`
- `ResponseEntity<T> toEntity(class<T> bodyType)`