UT4: SERVICIOS EN RED

+

Antonio León Almodóvar DAM 2ºB

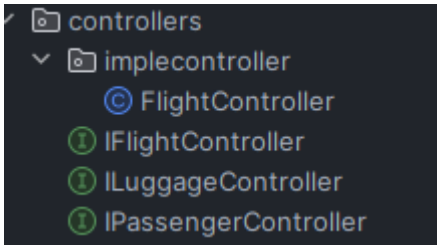
Índice

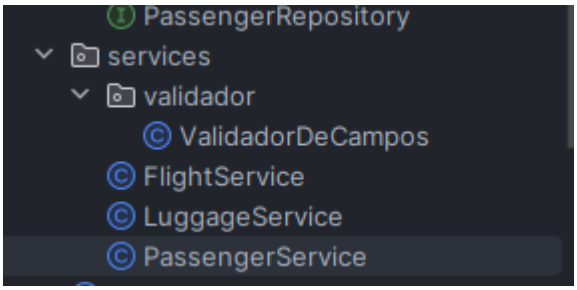
Resumen

Esta práctica se consta de una API REST basada en SpringBoot y un cliente basado en java. Para el servidor he acabado decantándome (Por las charlas que se han tendido en clase) en usar un único controller para así evitar el acoplamiento entre los diferentes servicios.

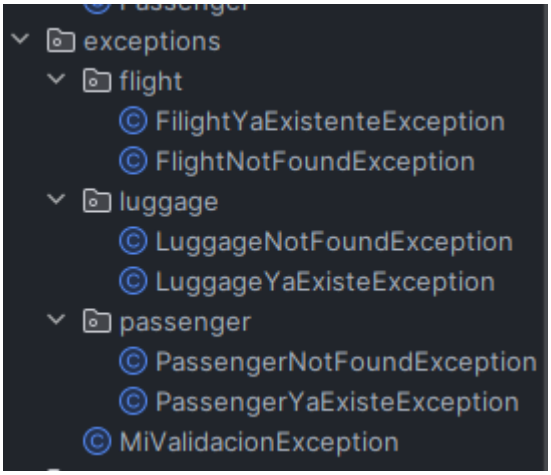
Desarrollo del trabajo

MI primer objetivo al afrontar este proyecto fue sin duda lanzarme a completar el servidor. Primeramente, dividí la sección de controllers en tres clases y tres interfaces y no fue hasta que tuve todos los Endpoints y métodos necesarios en los services (y algún que otro método extra en el repository) que me lance al desarrollo del cliente.





Esta fue una labor tediosa debido a la alta dependencia entre services. Por lo que decidí reducir los tres controllers a uno solo. Aun así, sostener todas las posibles posibilidades a base de booleans me resultó imposible, debido a eso decidí enfocar todos esos errores en el uso de excepciones y de este modo saber que falla, porque, cuándo y dónde.



El cliente fue relativamente simple, se basó en ampliar a tres diferentes acciones (vuelos, pasajeros y equipajes) la actividad 4 cliente que subiste y controlar por medio de excepciones y status codes las diferentes respuestas que el servidor brindada al cliente. 409 cuando hay un POST/PUT duplicado, 400 cuando hay un error de validación…

```
HttpResponse<String> respuesta = client.send(request,
System.out.println(respuesta.body());

if (respuesta.statusCode() == 201) {

}else if (respuesta.statusCode() == 400) {
    throw new BadRequestException();
} else if (respuesta.statusCode() == 404) {
    throw new NotFoundException();
}else if (respuesta.statusCode() == 409) {
    throw new YaExisteException();
} else {
    throw new Exception();
}
```

Problemas principales

RequestParam

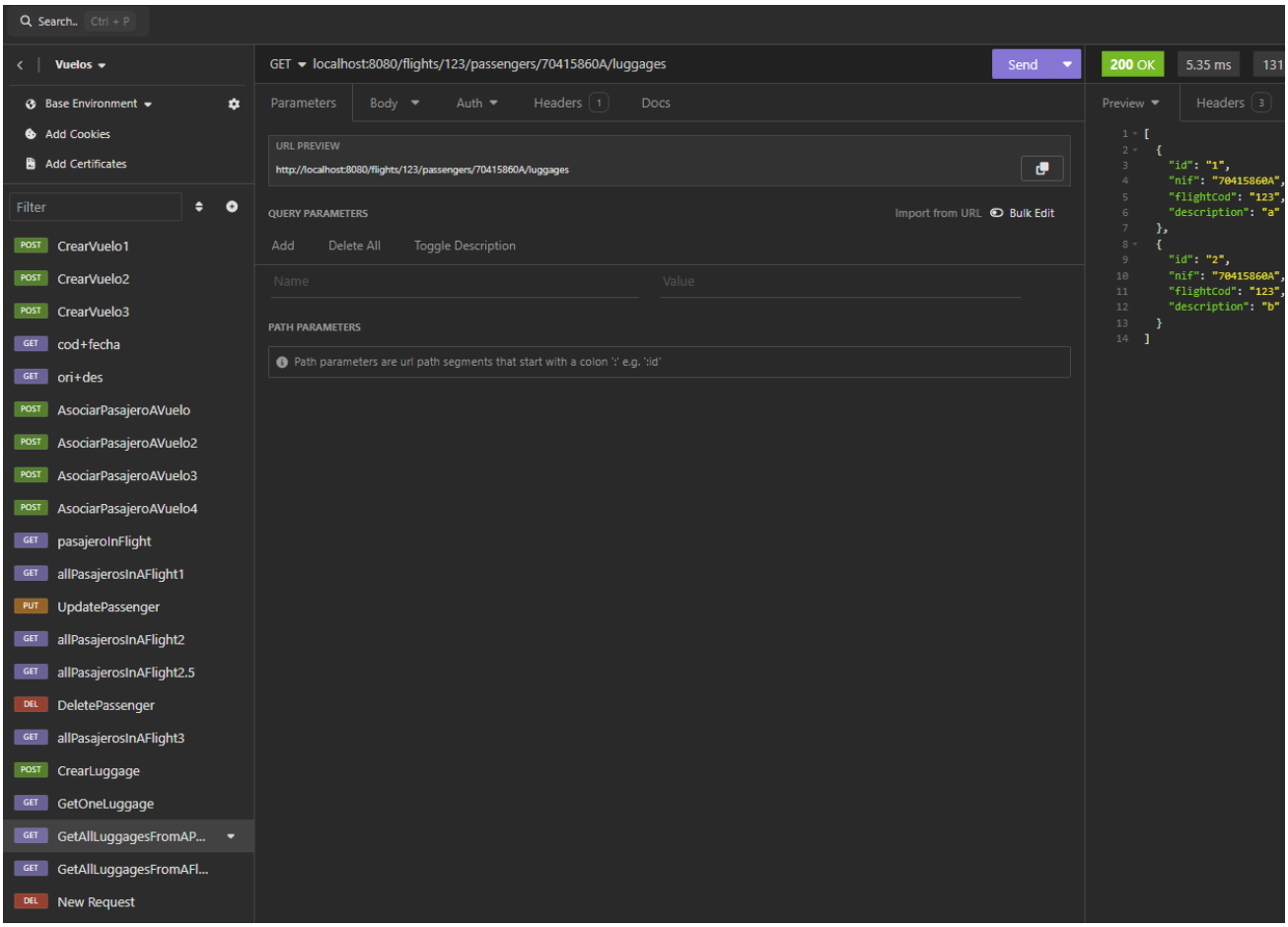
Al principio descubrir cómo afrontar los “date=DATE” resultó un problema, un @PathVariable no servía y tras varios minutos de investigación descubrí que los datos alojados de esta forma se consiguen por medio de @RequestParam.

Dependencia

Este problema me ha atormentado durante días, en definitiva, me ha hecho hacer este trabajo más difícil de lo que en verdad era. He cambiado la arquitectura del servidor tres veces, desde una clase abstracta con los tres repositorios que es el padre de los tres services hasta el “infame circular import”. Al final “he cortado por lo sano” y lo he reducido todo a un controller con los services.

Depuración

No mentiría si dijera que la inmensa mayoría de testeos lo he hecho en insomnia.



Eso no quita que haya tenido que debuguear en el cliente para saber porque las cosas no iban como deberían…

Además de los siempre confiables “debug prints”. En definitiva, el proceso de testeo se ha mostrado una tediosa labor de repetir los mismos errores una y otra vez tratando de que, dejaran de ser errores y horas de forzar excepciones para verificar que efectivamente printeaban lo que yo quería que printeasen.