Práctica 2: Consumer Driven Contract Testing y Testing de carga con Artillery

Parte A: Consumer Driven Contract Testing (3 ptos)

En la aplicación de la práctica 3 de la asignatura 2.1 Tecnologías de Servicios de Internet, se desarrolló una aplicación con cuatro servicios. Dicha aplicación ha sido refactorizada y se han dejado exclusivamente los servicios *planner* y *toposervice* (se proporciona un zip con ambos servicios: ***cdct_enunciado***).

El servicio planner utiliza el servicio *toposervice* para conocer la orografía de una determinada ciudad. Ambos servicios van a ser desarrollados por equipos diferentes y deben poder ser desplegados en cualquier momento sin depender el uno del otro y asegurando la compatibilidad entre ellos. Para ello, ambos equipos han acordado utilizar **Consumer Driven Contract Testing** usando **Spring Cloud Contract**.

Se pide:

- Definir un contrato para el endpoint *getCity* del servicio *toposervice*. DIcho contrato debe devolver una ciudad y su orografía.
- Configurar el servicio *toposervice* de forma que se pueda validar el contrato contra la implementación de la API proporcionada por *toposervice.*
- Configurar el servicio *planner* e implementar un test que verifique el funcionamiento esperado del endpoint *getCity* del servicio *toposervice*.

Entregables:

- El código de los dos servicios se entregará dentro de una carpeta CDCT en el zip que se entregará con la práctica (ver instrucciones de entrega más abajo)

Nota: Para evaluar la corrección de la solución proporcionada, se utilizarán los siguientes comandos por consola, usando una versión Maven 3.6.x:

- En el servicio *toposervice*: **mvn clean install**
    - debe generar y ejecutar correctamente el test de aceptación (que deben pasar), generar el *stub* e instalarlo en el repositorio maven local
- En el servicio *planner*: **mvn clean test**
    - Debe descargar el *stub* y usarlo en el test como *mock* del servicio *toposervice*. Los tests deben pasar.

Parte B: Artillery (7 ptos)

Se desea definir las pruebas de carga de la API REST de una aplicación que gestiona una librería (proporcionada como ***Artillery_enunciado***). Concretamente, se pide la elaboración de un **Test Script** que contenga la configuración necesaria para poder hacer las pruebas de carga con el toolkit **Artillery** junto a los resultados de su ejecución.

1) **Requisitos/Especificaciones (2pts)**:

- Se establecerá una única fase de 40 segundos en la que habrá una tasa de 5 usuarios/segundo.
- Solo se permitirán 8 conexiones simultáneas a la API.
- Se establecerá como requisito que el 95% de las peticiones no supere los 100ms de latencia
- Se establecerá como requisito un 0% de tasa de error.

NOTA 1: No es necesario que la ejecución cumpla estos requisitos (podemos tener distintas máquinas con recursos más o menos limitados), solo hay que definirlos correctamente.

2) **Escenarios (5pts)**

A continuación se detallan los escenarios que se pretenden probar (se valorará darle nombre al escenario):

- **Escenario 1: Consulta del primer Libro**. Un usuario sin logear recupera todos los libros y pide el primero para ver más información
- **Escenario 2: Creación de un Libro**. Un usuario logueado crea un libro y después lo pide para comprobarlo.
- **Escenario 3: Borrado de un Libro** El administrador crea un libro, pero tras comprobarlo, lo borra y comprueba que ya no existe

NOTA 2: En los casos dónde un escenario realice acciones que requieran de autenticación, se realizará una petición de logIn previa.

NOTA 3: Para simular un comportamiento lo más parecido a la realidad, se fijará la probabilidad de que un usuario realice un escenario:

- Escenario 1 (70%)
- Escenario 2 (20%)
- Escenario 3 (10%)

NOTA 4: Para las peticiones realizadas en cada escenario, se pide verificar los códigos de estado (HTTP) de la respuesta.

3) Entregables

Los ficheros entregables serán los siguientes:

- **solucion.yml** -> Fichero con toda la configuración del Test Script
- **output.json** -> Fichero que contiene los resultados de una ejecución del test de carga (Fichero de salida)
- Ficheros adicionales que contengan datos que utilicen nuestro Test Script (en el caso de utilizarlos).

AYUDA:

- Se proporciona junto al código de la aplicación un fichero CSV:
  - *books.csv*: Contiene una colección de libros utilizables para generar algunos escenarios.
- A la hora de capturar el primer elemento de una lista (ya sea del cuerpo de la petición o de un atributo) puede realizarse usando una sintaxis similar a la de otras librerías de testing:
  - Ejemplo 1: "$[0].atributo"
  - Ejemplo 2: "$.atributoLista[0].atributo

Formato de entrega

La práctica se entregará teniendo en cuenta los siguientes aspectos:

- La práctica se entregará como un fichero .zip que contenga los ficheros establecidos en "*Entregables*". El nombre del fichero .zip será el correo URJC del alumno (sin @alumnos.urjc.es). Este archivo comprimido contendrá 2 carpetas:
  - ***CDCT/*** Carpeta con la solución de la práctica de Contract Testing que debe incluir la implementación completa de ambos servicios (planner y toposervice)
    - Se debe cambiar el groupId al correo del alumno (sin el punto), o si son dos alumnos, los correos (sin puntos) separados por un guión: `<groupId>pperezf2020-zgonzalez2020</groupId>`
  - ***Artillery/*** Carpeta con los archivos entregables definidos
- El proyecto se puede crear con cualquier editor o IDE.

Las prácticas se podrán realizar de forma individual o por parejas. En caso de que la práctica se haga por parejas:

- Sólo será entregada por uno de los alumnos
- El nombre del fichero .zip contendrá el correo de ambos alumnos separado por guión. Por ejemplo `p.perezf2020-z.gonzalez2020.zip`