DESAFÍO FALABELLA

Para el ejercicio 1 se utilizó una planilla de Excel, con casos de prueba positivos y negativos, primero se prueba el happy path, es decir, formato de fecha y país correcto, tiene que devolver un status 200 y un body con datos correctos de temperatura en formato Json. Luego se prueban las diferentes combinaciones de datos incorrectos de país, ciudad y formatos incorrectos de fecha, y tiene que devolver errores. También se prueba enviar un método POST y tiene que devolver un error 405 method not allowed. Eventualmente si hubiera un token podría devolver un 401 unauthorized o un 500 server error, si se ingresa un dato en formato no válido y la API no maneja la excepción.

Para el ejercicio 2 se eligieron los endpoints /POSTS /COMMENTS /USERS

En el endpoint /posts la primer petición valida si devuelve un status 200 y si la respuesta es un Json.

La segunda petición valida para el endpoint/comments, si devuelve un status 200 y si la respuesta es un Json.

La tercera petición valida que si se envía un endpoint erróneo devuelve un not found.

La cuarta petición valida para el endopoint posts que cada posteo tenga un userId, eventualmente se podría corroborar todo el formato del objeto, que tenga cada uno de los atributos requeridos.

La petición 5 es para el endpoint posts y busca validar si dos posteos no tienen el mismo título, esto se hizo para mostrar cómo se podría validar cualquier condición que el Product Owner pidiera, ya sea que 2 posteos no tengan el mismo título o que no tengan el mismo id o el mismo userId, etc.

La sexta petición es para validar si al buscar un posteo devuelve el posteo correcto, según el id seleccionado. También esta petición valida si el objeto devuelto contiene los atributos especificados, al compararlos con strings ingresados según los requerimientos del Product Owner, del objeto recibido como Json ( en este caso, un posteo) se extraen las claves, de la estructura clave-valor y se almacenan en un arreglo, que luego sirve para comparar con los atributos ingresados. De esta forma se compara si la estructura del objeto es la especificada.

La séptima petición crea un nuevo posteo y valida que ese posteo haya sido creado, se utilizan variables de entorno y un body variable como pre-request

La octava petición modifica un posteo y verifica que se haya modificado

La novena petición recupera los comments y verifica que devuelva un status 200 y que el tiempo de respuesta sea menor a 300milisegundos, esta validación es interesante para ver el rendimiento del servidor.

La décima petición es recupera los users y valida que haya 10 usuarios.

La undécima petición recupera todos los usuarios también y valida que cada usuario tenga todos los ítems necesarios en su atributo address que a su vez es un objeto dentro de usuario.

Finalmente se podrían correr todos los request de la collection, hacer click en run collection, luego seleccionar la cantidad de iteraciones y el delay y luego click en run.

Se hizo una cantidad variada de scripts para realizar distintas validaciones que se podrían usar en un escenario real.

Para finalizar, creo que los puntos a tener en cuenta para probar una API, son: Probar lo antes posible para lograr solucionar los bugs con anticipación, sobre cada microservicio que se va desarrollando. De esta manera, se va conociendo cada parte del sistema, su funcionamiento y la interacción entre microservicios. Si se ha probado cada parte del sistema por separado, es más fácil encontrar errores y sus soluciones, probando de esta forma modular. Y Además no arrastrar un funcionamiento defectuoso a etapas avanzadas del proyecto donde se pueden producir demoras en las entregas. Al probar una API lo primero que se tendría que revisar es la parte funcional, pedir la documentación y de acuerdo a las especificaciones diseñar casos de prueba para probar cada una de las funciones y comprobar que responde adecuadamente. Luego se probaría la seguridad, comprobar la autenticación, es decir que sólo tenga acceso al servicio el usuario que tenga permiso para ello. También debería probarse que si hay datos sensibles, estén encriptados. Posteriormente se podrían hacer pruebas de rendimiento con JMeter y luego se podrían hacer pruebas de integración con otras apis, hacer funcionalidades completas que involucren varios microservicios.

Esperando sea de su agrado saluda Atte.

Luis Sabina