Lucas Percello – 19/2/2022

Cordoba, Argentina

Resolución de Ejercicios - CleverIT

Ejercicio 1 (QAA-01) API Temperatura

Descripción:
Crear los casos de prueba posibles para testear el siguiente microservicio GET, que recibe:
En el request la fecha del día actual en formato DD-MM-AAAA.
Dos headers, el primero es "Country", cuyo dominio de valores puede ser "Chile" o "Argentina" y el segundo es "City", cuyo dominio de
valores es "Santiago","Arica", "Chiloe" (cuando se trata de Country Chile) y "BuenosAires", "SanJuan" (cuando country es Argentina)
Cuando el resultado es positivo entrega status ok y código http 200 y en caso de error el status es "client error" y en caso de error de
conexion el status es "server error" y en el caso de codigo para cada uno de estos status se entrega los codigos http de error estándar.
Considera que cuando es un País distinto a los señalados o una ciudad distinta el response "País y Ciudad incorrectos", y que todos los
campos a ingresar son Sensitive Case.
El response del microservicio devuelve la temperaturaActual y temperaturaDiaSiguiente en formato json.

Resolución - **Ejercicio 1 (QAA-01)** API Temperatura

Casos de Prueba API Temperatura

CP_001 – Verificar headers “Chile” o “Argentina”

Descripción: Verificar temperatura en países y ciudades esperadas.

Precondiciones: Escritura sensitive case. Fecha formato DD-MM-AAAA.

Encabezados Country = ‘Chile’. City = ‘Santiago’, ‘Arica’, ‘Chiloe’.

Country = ‘Argentina’. City = ‘Buenos Aires’, ‘San Juan’.

Resultado Esperado: Status Ok. Codigo 200.

CP_002 – Verificar headers de país incorrecto y ciudad en minúsculas.

Descripción: Verificar mensajes de error y escritura.

Precondiciones: Ciudades en minúsculas con acentos. Fecha formato DD-MM-AAAA.

Encabezados Country = 'Chile'. City = 'cordoba, 'bogota, 'sucre'.

Encabezados Country = 'Uruguay'. City = 'Buenos Aires', 'San Juan'.

Resultado Esperado: País y ciudad incorrectos. Todos los campos a ingresar son sensitive case.

CP_003 – Verificar headers distintos a los esperados.

Descripción: Verificar mensajes de error y escritura.

Precondiciones: Ciudades en minúsculas con acentos. Fecha formato DD-MM-AAAA.

Encabezados Country = 'Brasil. City = 'Sao Pablo'.

Encabezados Country = 'Italia'. City = 'Roma'.

Resultado Esperado: País y ciudad incorrectos.

CP_004 – Verificar headers sin acceso a recursos

Descripción: Verificar mensajes de error de cliente sin privilegios.

Precondiciones: Usuario sin privilegios suficientes de acceso al environment. Ej. Desarrollo.

Resultado Esperado: Client Error. Codigo 403.

CP_005 – Verificar headers sin conexión

Descripción: Verificar mensajes de error del servidor por problemas de conexión.

Precondiciones: El usuario no está conectado a una red o no esta logueado en una vpn.

Resultado Esperado: Server Error. Codigo 500.

Ejercicio 2 (QAA-02) - Automatización de API
Descripción:
Tenemos la siguiente API https://jsonplaceholder.typicode.com/
Y esta API tiene los siguientes EndPoints de consulta:
posts 100 posts
comments 500 comments

albums 100 albums
photos 5000 photos
todos 200 todos
users 10 users

Considerando esta API y sus recursos, selecciona 3 de estos recursos y diseñar los casos de prueba funcionales que veas adecuados y luego
automatiza al menos 6 casos de prueba distintos (debe ser en Postman)

Resolución - **Ejercicio 2 (QAA-02)** - Automatización de API

Casos de Prueba - Automatización de API

Recursos Seleccionados = [**posts, comments, photos, users**]

CP_001 – Verificar comments con parámetros duplicados

Descripción: Verificar parámetros duplicados.

Precondiciones: Petición GET. Ingresar parámetros duplicados.

Key = "q", Value="foo". Key = "q", Value="bar".

Resultado Esperado: Status Ok. Codigo 200. Soporta un arreglo vacío.

CP_002 – Verificar comments con valor booleano en false

Descripción: Verificar parámetros duplicados.

Precondiciones: Petición GET. Consulta url "published" = false.

Resultado Esperado: Status Ok. Codigo 200.

CP_003 – Verificar photos con id no encontrado

Descripción: Verificar mensaje de recurso no encontrado.

Precondiciones: Petición GET. Consulta url photos con id 500000.

Resultado Esperado: Client Error. Codigo 404 Not Found.

CP_004 – Verificar posts order by desc

Descripción: Verificar orden descendente.

Precondiciones: Petición POST. Parámetros _sort = id, _order = DESC.

Resultado Esperado: Status Ok. Codigo 200. El Max id se muestra en la parte superior del response.

CP_005 – Buscar posts por número de teléfono

Descripción: Consultar por parámetros un numero de teléfono de usuario.

Precondiciones: Parámetros tel = 123.

Resultado Esperado: Status Ok. Codigo 200. Se muestra una lista con las coincidencias.

CP_006 – Buscar posts por id inexistente y petición incorrecta

Descripción: Consultar un id por url.

Precondiciones: Consulta errónea tipo PUT con id por url 1517.

Resultado Esperado: Server Error. Codigo 500. Internal Server Error.

Ejercicio 3 (QAA-03) Explicación de técnicas
Para ambos ejercicios 1 y 2, explicar las técnicas que usaste para generar las pruebas y donde fueron aplicadas. Además de cuáles son los puntos a tener en cuenta cuando probamos una API.

Resolución - **Ejercicio 3 (QAA-03)** Explicación de técnicas

Para ambos ejercicios 1 y 2, explicar las técnicas que usaste para generar las pruebas y donde fueron aplicadas.

Respuesta Ejercicio 1: Aplico tablas de verdad, dependiendo de las variables que deba tener en cuenta.

Respuesta Ejercicio 2: Utilice Postman y Newman para los tests. Se genera un reporte en la carpeta postman del proyecto. Ese reporte es html, por lo que se puede visualizar en cualquier navegador.

Agrego la funcionalidad de la librería husky. Con cada commit, se publican los resultados de los tests en Newman. (carpeta postman del proyecto)

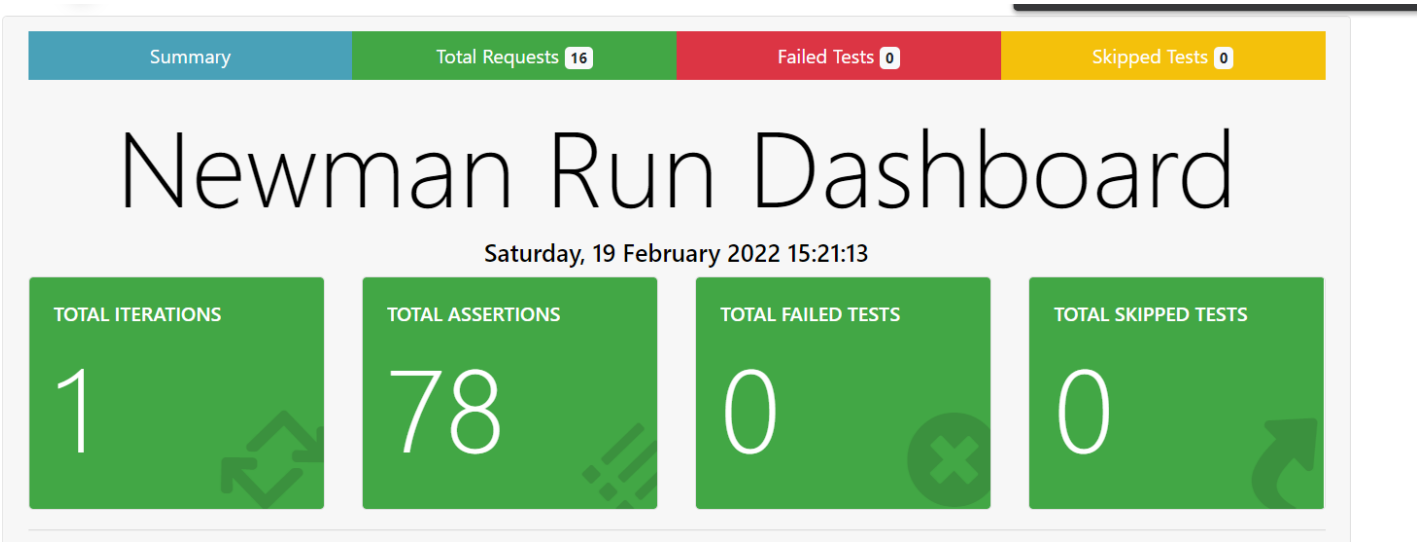
Esto sería útil para verificar el estado actual de los tests, según el último commit que realizo un miembro del equipo en esa rama.

En la carpeta postman también se encuentran las colecciones con su respectivo environmet.

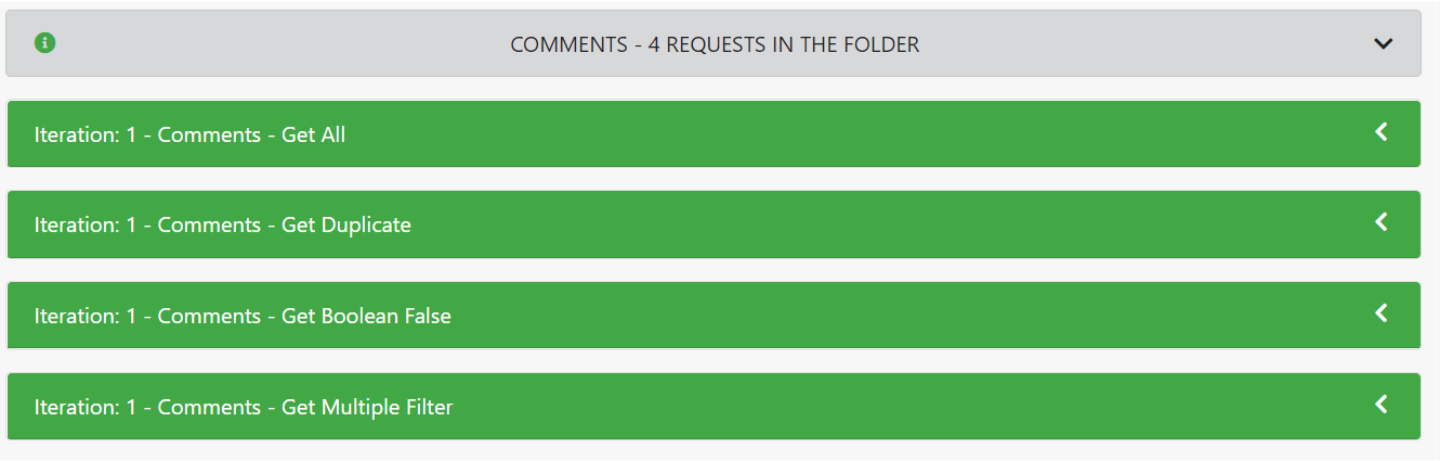
Se puede correr la aplicación con npm run test, o bien realizar un commit para ver la compilación antes del push.

Codigo de repositorio en: https://github.com/lucjs/**automation_postman_newman**

./postman/report.html



Ej: Comments



Además de cuáles son los puntos a tener en cuenta cuando probamos una API.

Respuesta Ejercicio 3

- Validar datos para la solicitud
- Validar datos de comprobación de una respuesta.
- Validar los datos de retorno.
- Validar los headers de la respuesta.
- Validar si la respuesta está de acuerdo.
- Validar si con el content-type modifica el comportamiento o continúa igual.
- Validar si la estructura del JSON o XML está correcta.
- Validar si cuando da error el status está de acuerdo con los códigos de error.
- Validar si hay una solicitud con informacion incompleta.