

Examen QA Automation

3) Cuáles son los puntos a tener en cuenta cuando probamos una API?

Con respecto a los puntos a tener en cuenta al testear api's, primero tenemos que entender que es una api, según su definición podemos decir

Las API (Application Programming Interfaces) son un conjunto de comandos, funciones y protocolos que permiten la comunicación entre softwares.

Ahora que lo sabemos, cuando creemos nuestro escenario de prueba, debemos conocer plenamente su contrato, para esto podemos guiarnos con Swagger, pues a través de este sabremos los recursos, cuerpo de mensaje, etc.

Debemos conocer las siguientes especificaciones de nuestro services a testear:

RequestData:

- Method
- URI
- Headers
- Query Parameters
- Body

ResponseData:

- Headers
- Body
- Status code

Request behavior:

- Validando los datos de retorno.
- Validar los headers de la respuesta
- Validando si la respuesta está de acuerdo.
- Validar si con el content-type modificado el comportamiento 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 información incompleta, cuál será el comportamiento de la solicitud.

Después de recolectados estos datos ya podemos armar un escenario y sabremos cuáles son los comportamientos esperados en nuestra API para validarlos.

En mi opinión, la mejor práctica que podemos traer para testear microservicios es la de contract testing/Consumer-driven.

¿Qué son los Consumer Driven Contract?

Es un patrón que nos permite testear y comprobar las diferentes interacciones entre los diferentes servicios de nuestro ecosistema y sus clientes.

En estos tests existen un mínimo de dos actores, un *consumidor*, que puede ser el cliente, y un *proveedor*, que será el servicio en sí mismo. El *consumidor* capturará sus expectativas respecto al *proveedor* y lo almacenará en un file que llamará 'contrato', y es lo que usará para realizar los tests. Un 'contrato' sería la definición de la forma en que va a comportarse nuestro servicio en base a una solicitud(me gusta llamarlo swagger ejecutable). Es decir, nosotros vamos a definir que el cliente necesitará por ejemplo, los campos nombre, apellido, y edad, por tanto lo que se va a comprobar es que esto campos le siguen llegando, independientemente de que lleguen otros campos que quizás utilicen o no otros usuarios del endpoint, de esta forma aunque el service se haya actualizado con nuevos campos, sabemos que los clientes que utilicen la versión anterior siguen teniendo soporte y que seguimos ofreciendo compatibilidad.