Desafio 1:

Realizando un estudio sobre el código entregado identifico que hay tres áreas que podrían mejorarse o refactorizarse para aumentar la eficiencia y la facilidad de mantenimiento del framework.

1. Hay algunas piezas de lógica que se repiten en los dos escenarios. Por ejemplo, la generación de UUID, la manipulación de archivos y el manejo de esperas. Sería beneficioso centralizar esta lógica común en métodos reutilizables para evitar la duplicación de código y facilitar futuras modificaciones y mantenimiento. Esto podría lograrse extrayendo métodos de utilidad o utilizando hooks como el Before de Karate para la limpieza del escenario.
2. Algunas secciones del código, especialmente las secciones donde se construyen los objetos JSON, podrían ser difíciles de entender y mantener debido a la mezcla de comillas simples y dobles. Sería útil refactorizar estas secciones para mejorar la legibilidad y evitar errores potenciales. Una opción podría ser utilizar comillas triples para envolver los objetos JSON y evitar la necesidad de escapar comillas simples y dobles.
3. Actualmente, el código no maneja explícitamente situaciones de excepción o error que podrían ocurrir durante la ejecución, como errores de red, errores al cargar archivos, etc. Sería beneficioso implementar un manejo adecuado de excepciones para capturar y manejar estos errores de manera adecuada. Esto ayudaría a mejorar la robustez del framework y proporcionaría una mejor experiencia de usuario al identificar y manejar errores de manera más efectiva.

Desafio 2:

Con el fin de mejorar el secciones del código donde se pueden utilizar métodos para que estos sean fácil de realizarle mantenimiento y que se reutilicen en N escenarios.

1. Se agregan datos al archivo karate-config.js con el fin de tener estos valores por defecto.

function fn() {  
 var config = {  
 bucketName: 'test-automation-qa',  
 folderRecaudoFiles: 'files-to-cash-in',  
 fileExtension: '.csv',  
 baseUrl: 'https://test-container-qa.prueba.co/v1/entity/novelties/'  
  
 };  
  
 return config;  
}

1. Se elimina la linea de código

\* def waitTime = function(seconds) { java.lang.Thread.sleep(seconds \* 1000) }

Del Background con el fin de tener un código mas limpio y se pasa a una clase a parte :

function waitTime(seconds) {

java.lang.Thread.sleep(seconds \* 1000);

}

1. Se crea una clase que cree el numero de UUID

package co.cobre.lib.qa.util;  
  
public class CommonUtils {  
  
 public static String generateUUID() {  
 return java.util.UUID.*randomUUID*().toString();  
 }  
  
}

1. Se cambia el formato como se llaman los valores de la tabla de examples:

**Antes:**

\* def renameFile = FileUtils.renameFile(currentFilePath, currentFileName, newPathNewFile, noveltyUuid, fn.fileExtension)

**Despues:**

\* def renameFile = FileUtils.renameFile('<currentFilePath>', '<currentFileName>', '<newPathNewFile>', noveltyUuid, fileExtension)

1. El valor de URL se toma desde el archivo karate-config.js , este se reutiliza para los dos test en la parte 1 se llama :

*Parte 1: Given url baseUrl + noveltyUuid*

*Parte 2: Given url baseUrl + noveltyUuid + '/' + 'novelty-details'*

1. Para esta linea de código se identifica que la variable **total** no tiene valor asignado por lo tanto puede presentar error ya que la condición no se completa

And retry until responseStatus == 200 && response.cashInNovelty.status == status && response.cashInNoveltyDetailsCounters.total == total

1. La linea con codigo

\* match totalAmount == '<>'

No tiene el valor con el cual se va a validar .

1. El valor de pathJsonDataWithExpectedInformation no tienen utilidad en el escenario.
2. Para el escenario dos se corrige el siguiente código

**Antes:**

\* def renameFile = FileUtils.renameFile(currentFilePath, currentFileName, newPathNewFile, noveltyUuid, fileExtension)

**Despues:**

\* def renameFile = FileUtils.renameFile('<currentFilePath>', '<currentFileName>', '<newPathNewFile>', noveltyUuid, fileExtension)

Se elimina el valor de *pathJsonDataWithExpectedInformation* de la tabla de examples

1. De acuerdo con los métodos creado se reutilizan para este escenario como:

*\* def noveltyUuid = CommonUtils.generateUUID()*

Se utilizan los valores asignados en el archivo karate-config.js