RiverTech Automation Testing Test Guide

Installation of a BDD framework

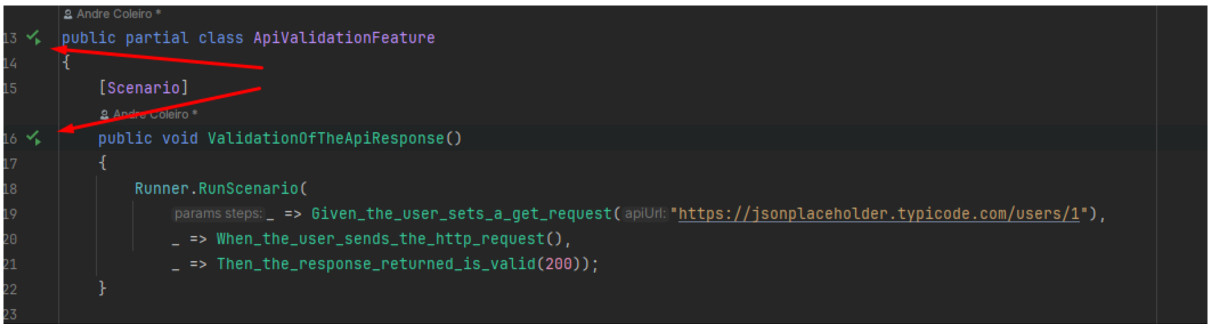
Installed a BDD framework using NuGet packages window. I am using LightBDD since in the test it states that RiverTech uses LightBDD framework.

Creating the API Testing Exercise Features & Steps

For the API testing exercise, two testing scenarios were created. One of the scenario, I am testing that the API returns a valid response code of **200**. In the other scenario, I am asserting that the API returns the expected field and values.

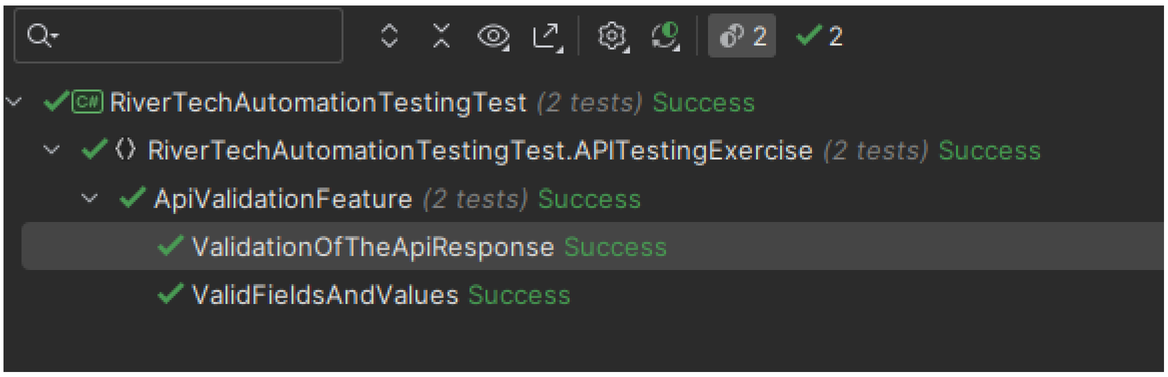
In order to satisfy these tests, I created two partial classes named **ApiValidationFeature.** One class includes the Features of the scenario and the other includes the Steps.

Once the steps are created and implemented for both scenarios, they can be executed by clicking the run button near the scenario method or the class to execute all tests.



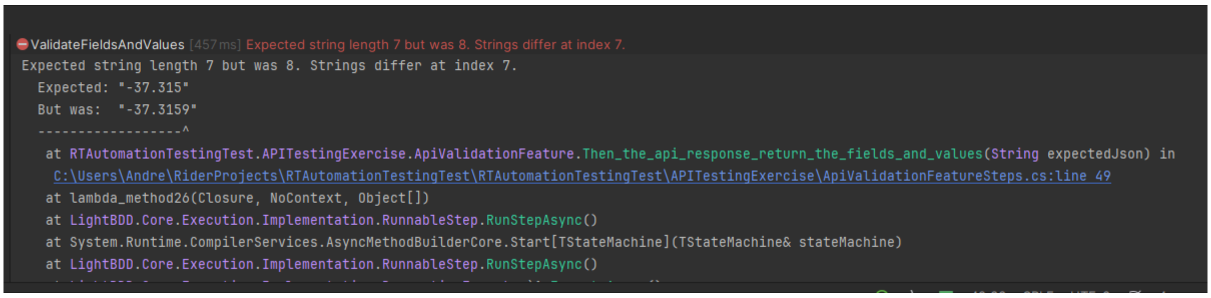
```
Andre Coleiro *
public partial class ApiValidationFeature
{
    [Scenario]
    Andre Coleiro *
    public void ValidationOfTheApiResponse()
    {
        Runner.RunScenario(
            params steps: _ => Given_the_user_sets_a_get_request( apiUrl: "https://jsonplaceholder.typicode.com/users/1"),
            _ => When_the_user_sends_the_http_request(),
            _ => Then_the_response_returned_is_valid(200));
    }
```

Using the Unit Tests Explorer, one could identify if the test succeeded or not. In this case both tests are successful.



```
2  ✓2
RiverTechAutomationTestingTest (2 tests) Success
  RiverTechAutomationTestingTest.APITestingExercise (2 tests) Success
    ApiValidationFeature (2 tests) Success
      ValidationOfTheApiResponse Success
      ValidFieldsAndValues Success
```

If a test fails, and the tester is comparing two values for example, in the unit test console it will illustrate what the value is and what the tester is expecting. This will easily show that the expected value do not match with the value from the GET HTTP request.



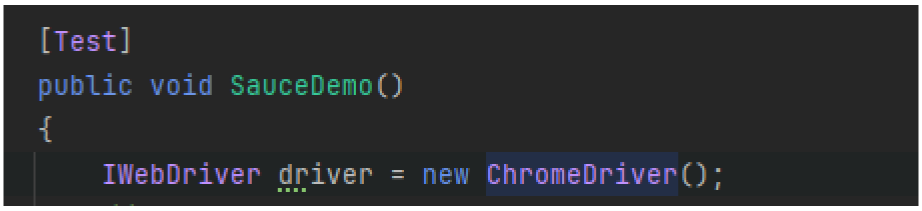
```
ValidateFieldsAndValues [457 ms] Expected string length 7 but was 8. Strings differ at index 7.
Expected string length 7 but was 8. Strings differ at index 7.
  Expected: "-37.315"
  But was:  "-37.3159"
  ------------------^
   at RTAutomationTestingTest.APITestingExercise.ApiValidationFeature.Then_the_api_response_return_the_fields_and_values(String expectedJson) in
    C:\Users\Andre\RiderProjects\RTAutomationTestingTest\RTAutomationTestingTest\APITestingExercise\ApiValidationFeatureSteps.cs:line 49
   at lambda_method26(Closure, NoContext, Object[])
   at LightBDD.Core.Execution.Implementation.RunnableStep.RunStepAsync()
   at System.Runtime.CompilerServices.AsyncMethodBuilderCore.Start[TStateMachine](TStateMachine& stateMachine)
   at LightBDD.Core.Execution.Implementation.RunnableStep.RunStepAsync()
```

UI Automation Testing Exercise

To implement the UI automation test, the selenium webdriver needs to be installed from the NuGet packages. Once installed UI Automation tests can be executed.

To perform a UI Automation test, I started by creating a class and inside of it I declared a method with a **[Test]** tag. This marks the method as a test and can be executed from the unit test explorer.

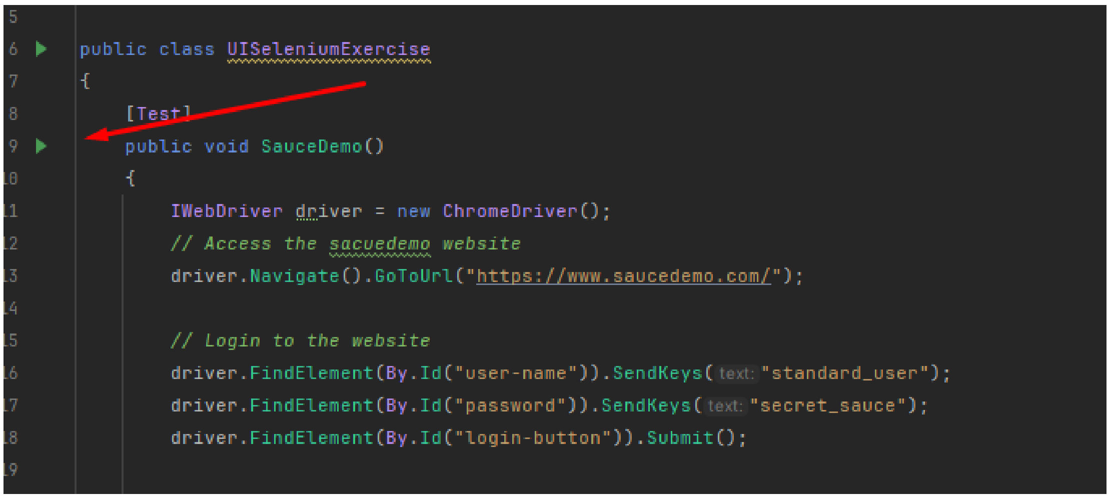
First, I created a driver as shown below so that I can write test against the Chrome browser.



```
[Test]
public void SauceDemo()
{
    IWebDriver driver = new ChromeDriver();
```

Then I went through the scenario manually and took notes of the elements **ids** and **classes** that are going to be used in the tests. These ids and classes were used in various situations such as to write a string in a field using the **SendKeys(),** to submit a form **Submit()**, to click a button **Click()** etc etc…

Once the implementation was completed, I executed the test using the play button shown in the screen shot. This launched the Chrome browser and automate all the steps



```
public class UISeleniumExercise
{
    [Test]
    public void SauceDemo()
    {
        IWebDriver driver = new ChromeDriver();
        // Access the sacuedemo website
        driver.Navigate().GoToUrl("https://www.saucedemo.com/");

        // Login to the website
        driver.FindElement(By.Id("user-name")).SendKeys(text: "standard_user");
        driver.FindElement(By.Id("password")).SendKeys(text: "secret_sauce");
        driver.FindElement(By.Id("login-button")).Submit();

```