GiG Code Challenge

For Backend Software Developer in Test

Framework and Tools

.NET Framework 4.6.1: https://dotnet.microsoft.com/download/dotnet-framework/net46

SpecFlow 1.9.0 : https://www.nuget.org/packages/SpecRun.SpecFlow.1-9-0/

kafka-net : https://www.nuget.org/packages/kafka-net/0.9.0.65

MSTest.TestFramework: https://www.nuget.org/packages/MSTest.TestFramework/

Docker : https://www.docker.com/

Kafka : https://kafka.apache.org/

Git project: https://github.com/renaldo122/CodeChallengeTesting

Solution: GiGCodeChallenge

Details regarding directories in the attached solution:

- GiGCodeChallenge.Common
  - Exceptions -> Used this directory for Custom exceptions, to represent errors that occur during application execution
  - Extensions-> Method extensions for Json to object conversion
  - Models-> Models for RESTful API tests and Stream processing
- GiGCodeChallenge.StreamProcessing
  - BaseService –> Base Service to implement base methods for Stream processing
  - Consumer-> Interfaces for implementing methods to Consume messages
  - Producer-> Interfaces for implementing methods to produce messages on Kafka
- GiGCodeChallenge.Api.Tests
  - BaseSteps-> Implements a Base class for API testing steps
  - CommonSteps-> Implements common steps for features
  - Configuration -> Configuration used in testing
  - Features -> Features text describe all steps for methods
  - Helper-> Configuration and Initialize classes
  - Steps-> Implement methods for features scenarios.
  - Transformation-> Custom conversion step for the arguments of the step definitions

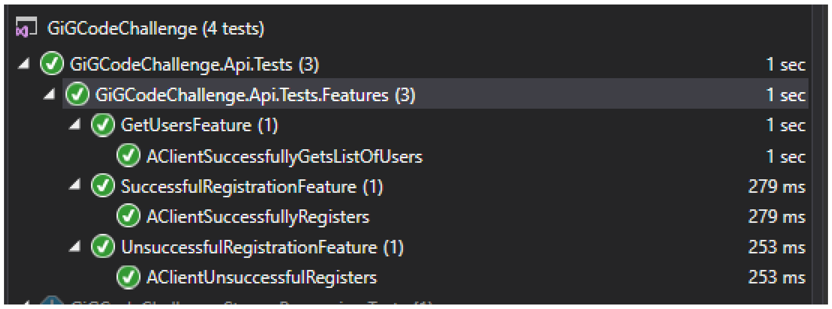
- GiGCodeChallenge.StreamProcessing.Tests
    - Base -> Implements a Base class methods for initialize Interface
    - Extensions -> Assert Extensions for objects
    - StreamProcessing->Implement Test methods for all message producers and consumers

Task 1 – RESTful API test

- I implemented the solution in **GiGCodeChallenge.Api.Tests**
- Created Feature files that describe all the scenarios for implementing tests methods with SpecFlow.
- I have created three tests, one for each scenario.
    - The first one (GetUsers) contains four steps and for each of them there is a method implemented:
        1. Given I have a GET API End Point 'api/users'
        2. When the client makes a get request
        3. *Then I expect response status code '200'*
        4. And Then I verify json response data to have list of users

        The step number 3 is used under **CommonSteps** directory, because it also occurs in other scenarios. In this way I could avoid duplicates or unnecessary code lines.
    - The second one (SucessfulRegistration) contains four steps
        1. Given I have a POST API End Point 'api/register' (is **CommonSteps**)
        2. When the client makes a POST request with the following data email eve.holt@reqres.in and password pistol
        3. Then I expect response status code '200' (is **CommonSteps**)
        4. Then I verify json response body to have a token
    - The third one (UnsuccessfulRegistration) contains four steps
        1. Given I have a POST API End Point 'api/register' (is **CommonSteps**)
        2. When the client makes a POST request with the following data email eve.holt@reqres.in
        3. Then I expect response status code '400' (is **CommonSteps**)
        4. Then I verify json response body to have an error message

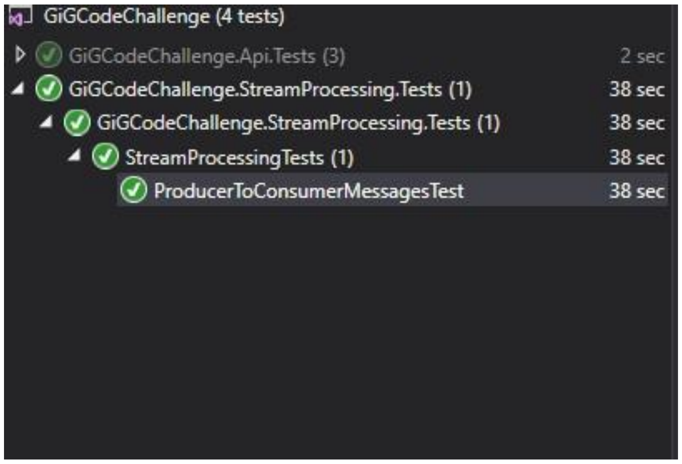
All tests successfully ran:



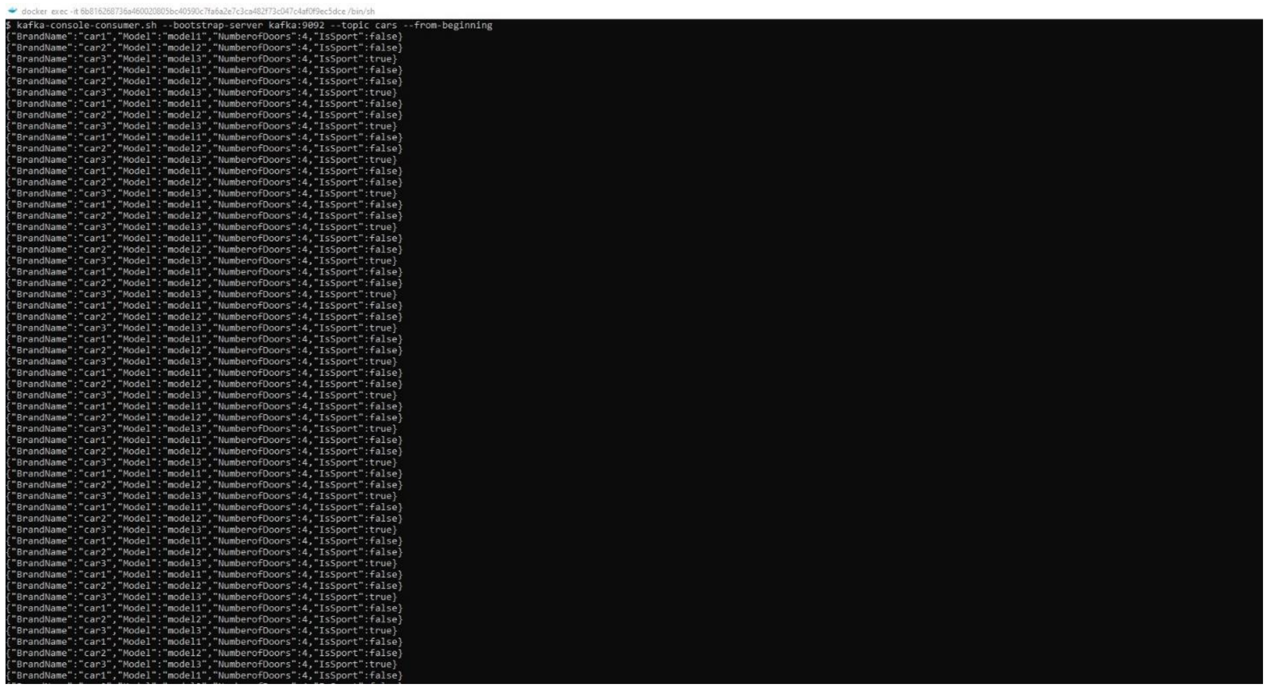
```
GiGCodeChallenge (4 tests)
◢ GiGCodeChallenge.Api.Tests (3)                                   1 sec
  ◢ GiGCodeChallenge.Api.Tests.Features (3)                        1 sec
    ◢ GetUsersFeature (1)                                          1 sec
        AClientSuccessfullyGetsListOfUsers                         1 sec
    ◢ SuccessfulRegistrationFeature (1)                           279 ms
        AClientSuccessfullyRegisters                              279 ms
    ◢ UnsuccessfulRegistrationFeature (1)                         253 ms
        AClientUnsuccessfulRegisters                              253 ms
```

Task 2 – Stream processing tests

- I installed Docker virtualization tool (links that I followed are on top of the page)
- Got an Apache Kafka Image on docker (bitnami/kafka and bitnami/zookeeper)
- On Docker I have 2 containers, kafka and zookeeper
- Kafka ran on 127.0.0.1:9092 and zookeeper run on 127.0.0.1:2181
- Tested all the port connection with telnet
- Used command prompt to create topic, create messages from producer and receive at consumer
- After created the environment, I tried to do the same steps like in command prompt, but this time in code. Created a library using kafka-net to connect to kafka broker. Then I used a method to create a topic and send message with producer. Topic name is Car and message contains car details, like required in the challenge.
- I used another method to get the message with consumer.
- Then I created the method that compares the messages if they are delivered correctly.



```
GiGCodeChallenge (4 tests)
▷ GiGCodeChallenge.Api.Tests (3)                               2 sec
◢ GiGCodeChallenge.StreamProcessing.Tests (1)                 38 sec
  ◢ GiGCodeChallenge.StreamProcessing.Tests (1)               38 sec
    ◢ StreamProcessingTests (1)                               38 sec
        ProducerToConsumerMessagesTest                        38 sec
```



docker exec -it 6b816268736a460020805bc40590c7fa6a2e7c3ca482f73c047c4af0f9ec5dce /bin/sh

```
$ kafka-console-consumer.sh --bootstrap-server kafka:9092 --topic cars --from-beginning
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
{"BrandName":"car2","Model":"model2","NumberofDoors":4,"IsSport":false}
{"BrandName":"car3","Model":"model3","NumberofDoors":4,"IsSport":true}
{"BrandName":"car1","Model":"model1","NumberofDoors":4,"IsSport":false}
```