This framework is developed on .net core platform c# and selenium for automate the applications through web browsers.

**How to run:** For executing current tests and writing new tests, clone the repository from <https://github.com/barisgul/SeleniumAutomationFramework.NetCore.git> and restore the applicaiton. The .net core sdk 3.1 should be installed on your computer.

To run API tests, you need to add the following key and token to the RestServices node in [this file](https://github.com/barisgul/SeleniumAutomationFramework.NetCore/blob/master/test/ApiTest/Trello.ApiTests/appsettings.json) (add to the Trello.ApiTests appsettings.json)

**Key:** 4927627eb2ba853e0449c44741ad8ddf

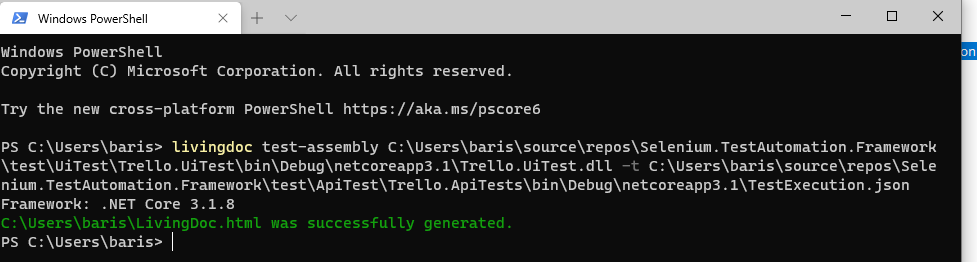
**Token:** 160c7cd24bf087db415a1c73f9b7bea9acc3ee9a8175da5ba37d0bb1f2b99a34

**Tech Stack**

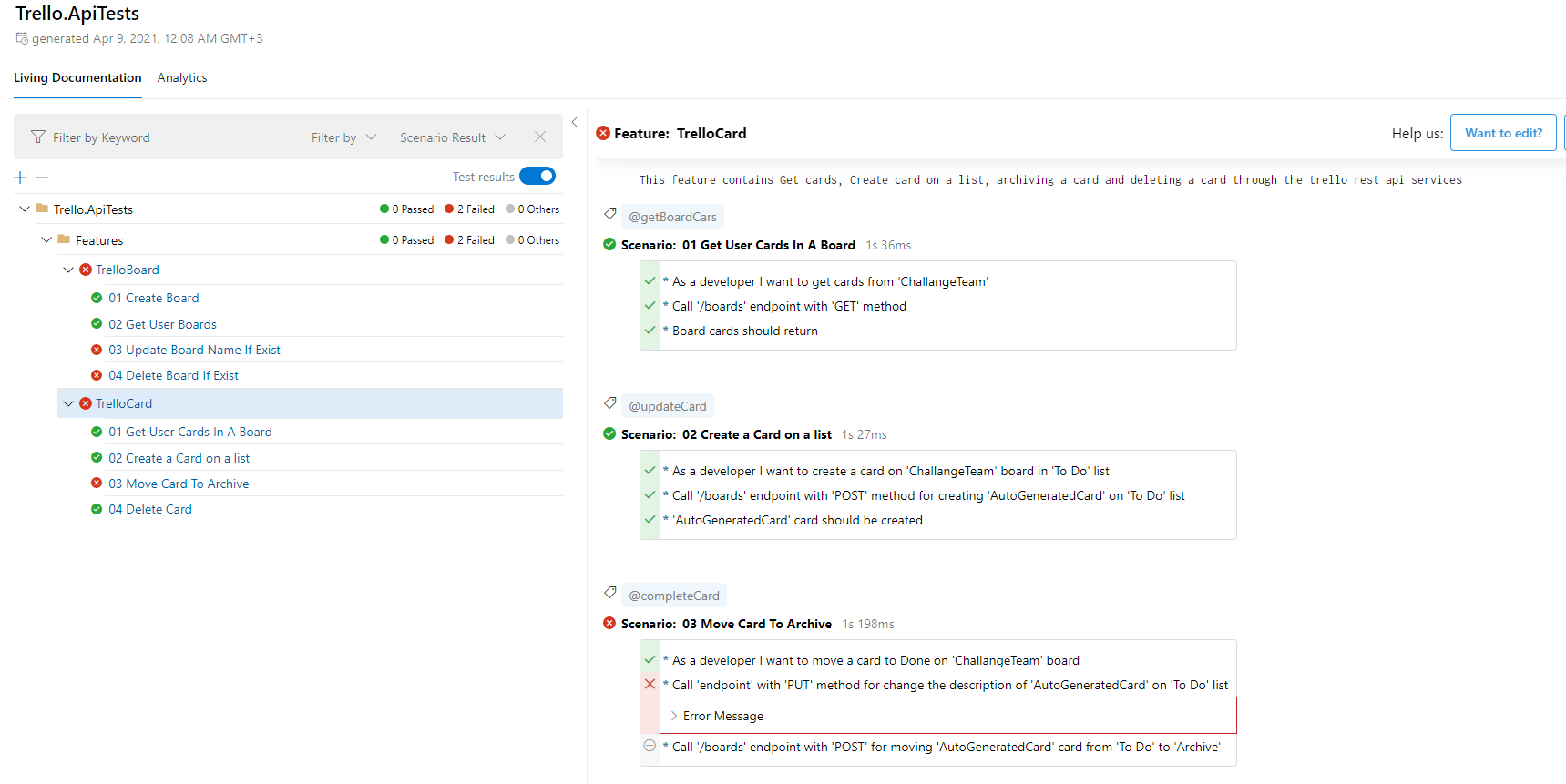
* **Platform:** C#, .Net Core
* **Test Frameworks:** 
  + **xUnit :** For write and execute tests
  + **Specflow:** Write tests in bdd format
* **Assertions**: FluentAssertions
* **Web Automation**
  + Selenium Webdriver 3.14
  + Browser Support **:** Chrome, IE, Firefox
* **Logging:** Serilog (Log to file, console and Elastic common scheme)
* **Configurations:** All configurations are supported by appsettings.json config files. BrowserTye, Url, timeout, execution environment etc.
* **Reporting:**
  + SpecFlow.Plus.LivingDoc.CLI for Specflow BDD based tests.
  + Custom vs xml reporting for unit integration tests

For creating a report SpecFlow.Plus.LivingDoc.CLI for Specflow should be installed and after installation this command should be executed in powershell.

*livingdoc test-assembly pathToTestExecution.json*



Then report will be generated as below

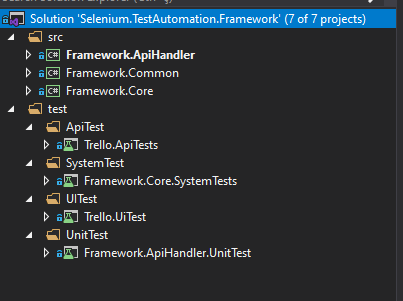


**Capabilities**

With this framework you can write and execute BDD based UI, Integration, System and API tests.

The **Framework.Core** project supports UI tests through the Selenium and **Framework.ApiHandler** project supports Api Testing over the RestSharp.

**Included Test Types and Projects**



**Projects**

* **Framework.ApiHandler**

ApiHandler is a wrapper of Restsharp and includes some generic methods for making rest requests.

* **Framework.Common**

Includes common usages like configuration management and common entities

* **Framework.Core**

Core Automation framework applicaiton. Some design patterns and solid principles used in this project.

**Test Types**

* **Api Tests**

Include some rest api tests works on backend. Specflow and Restsharp used. Framework.ApiHandler is a wrapper of Restsharp application.

* **System Tests**

Include some UI based browser automation tests based on Selenium. These tests allow us to make sure that the version we are using is working correctly. They works against to deployed selenium webdriver version that i used in the Framework.Core.

* **UI Tests**

Include some UI based browser automation tests based on Selenium. Works with different browsers. Default browser is chrome

* **Unit Tests**

Include some basic unit tests for ApiHandler project. I just added it to show how to create mocks and test the static methods.