

УДК 004

## АВТОМАТИЗАЦИЯ ФУНКЦИОНАЛЬНОГО ТЕСТИРОВАНИЯ ВЕБ-ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ФРЕЙМВОРКА

Расолько Александр Михайлович<sup>а</sup>, Тонкович Ирина Николаевна<sup>б</sup>

<sup>а</sup> *Минский инновационный университет, выпускник, SashaXT@hotmail.com*

<sup>б</sup> *Минский инновационный университет, кандидат химических наук, доцент, доцент кафедры информационных технологий, intonkovich@gmail.com*

### **Аннотация**

Рассмотрена методика разработки универсального фреймворка автоматизации функционального тестирования любых веб-приложений. Представлен обобщенный алгоритм работы автоматизированного тест-кейса.

**Ключевые слова:** автоматизированное тестирование, функциональное тестирование, веб-приложение, фреймворк, разработка фреймворка, Selenium WebDriver, C#.

**Веб:** <http://library.miu.by/journals!/item.science-xxi/issue.7/article.3.html>

**Поступила в редакцию:** 02.08.2018

## AUTOMATION OF WEB-APPLICATIONS FUNCTIONAL TESTING WITH THE USE OF FRAMEWORK

Rasolka Aliaksandr<sup>а</sup>, Tonkavich Iryna<sup>б</sup>

<sup>а</sup> *Minsk Innovation University, graduate, SashaXT@hotmail.com*

<sup>б</sup> *Minsk Innovation University, PhD in chemistry, Associate Professor, associate Professor at the Department of Information Technologies, intonkovich@gmail.com*

### **Abstract**

Methods of development of universal automation framework of functional testing of any web-application are considered. A generalized algorithm for the work of an automated test case is presented.

**Keywords:** automated testing, functional testing, web-application, framework, framework development, Selenium WebDriver, C#.

**Web:** <http://library.miu.by/journals!/item.science-xxi/issue.7/article.3.html>

**Received:** 02.08.2018

## Введение

Одним из факторов целесообразности разработки фреймворка автоматизации функционального тестирования веб-приложений является то, что в условиях современного мира веб-приложения становятся все более популярными и занимают значительную долю рынка программного обеспечения. Сегодня веб-приложения разрабатываются на гетерогенных платформах, с использованием различных языков программирования. При этом время разработки проектов существенно сокращается. Следовательно, увеличиваются и требования, предъявляемые к качеству реализации веб-приложений. По данным отчета World Quality Report 2017-18, доля QA&Testing в разработке программного обеспечения составляет 41 %. Авторы отчета выделяют в качестве одного из ключевых трендов в разработке программных продуктов повышение уровня автоматизации тестирования и широкое использование гибких методологий [1].

Особая роль в обеспечении эффективной поддержки полного жизненного цикла веб-приложений принадлежит автоматизированному функциональному тестированию с использованием фреймворков.

Фреймворки представляют собой решения, объединяющие наборы концепций, библиотек, готовых программных модулей и инструментов, обеспечивающих поддержку для автоматизированного тестирования программного обеспечения.

Каждый фреймворк специализируется на своем виде (модульное тестирование; тестирование мобильных приложений, производительности, веб-приложений и т.д.) и уровне тестирования, характеризуется своим набором инструментов.

При всем многообразии фреймворков автоматизации функционального тестирования веб-приложений (коммерческие, например, Coded UI, LoadRunner, UFT, Soap UI, TestComplete, Ranorex и др., бесплатные с открытым кодом, например, Selenium, Watir, RobotFramework и др.) они имеют общие черты: высокую абстракцию кода, универсальность (в рамках своего набора технологий) и переносимость используемых подходов, высокое качество реализации, возможность многократного тестирования пользователей без дополнительных затрат [2].

Однако обладают и рядом недостатков. Одни фреймворки довольно дорогие, другие – не универсальны вследствие ограниченности использования языков программирования или применимости к отдельным видам приложений. Многие фреймворки довольно сложны или делают поддержание автоматизированных тестов довольно трудоемким и затруднительным в силу того, что генерируемый код сильно нагроможден [3].

Одной из важнейших проблем разработки фреймворков является проблема реализации тестирования функциональности на уровне пользовательского интерфейса приложения. Особенно важен аспект автоматизации поддержки тестовых сценариев в

актуальном состоянии при изменении интерфейса веб-приложения, когда тесты формируются на основе работающей реализации приложения.

В настоящем исследовании предложена методика разработки универсального фреймворка автоматизации функционального тестирования любых веб-приложений. В центре внимания – вопросы тестирования функциональности, реализованной в пользовательском интерфейсе веб-приложения. А именно, решение вопросов поддержания тестовых сценариев в работоспособном состоянии для любого веб-приложения и при изменении его интерфейса.

## 1. Методика разработки фреймворка автоматизации функционального тестирования веб-приложений

В общем виде процесс разработки фреймворка автоматизации функционального тестирования веб-приложений будет включать следующие этапы:

- определение объема тестирования;
- выбор инструментов автоматизированного тестирования;
- планирование автоматизированных тестов;
- разработка дизайна фреймворка;
- разработка тестовых сценариев;
- настройка среды запуска автоматизированных тестов;
- запуск разработанных автоматизированных тестов и анализ результатов;
- поддержка автоматизированных тестов.

Основная идея исследования заключается в следующем. Фреймворк будет взаимодействовать с веб-приложением посредством тестовых сценариев, с учетом того, что основная функциональность приложения реализуется на стороне сервера.

Для разработки фреймворка использованы следующие инструменты автоматизированного тестирования:

*фреймворк Coded UI.* Позволяет автоматизировать веб-приложения в среде *Visual Studio*. К основным достоинствам инструмента можно отнести следующее: полная интеграция с TFS; большое количество возможностей при работе с .NET технологиями; поддержка от Microsoft; реализация AJAX запросов в веб-приложениях; поиск веб-элементов по нескольким критериям поиска (нечеткое совпадение);

*драйвер Selenium WebDriver.* Это инструмент, который используют для автоматизации тестирования веб-приложений (для управления браузером). Примечательно, что при вызове команд браузера Selenium WebDriver использует родной API для каждого конкретного браузера. К преимуществу данного инструмента следует отнести тот факт, что используется способ взаимодействия с браузером, близкий к действиям самого реального пользователя.

*библиотека .Net;*  
*язык программирования C#.*

Фреймворк представлен набором классов и методов на языке программирования C#, а также конфигурационных файлов в формате xml.

Для реализации фреймворка на языке программирования C# был создан проект CodedUI с именем AirBNB\_Tests.sln, который разделен на три пространства «Main», «TestCases» и «Data» (рисунок 1).

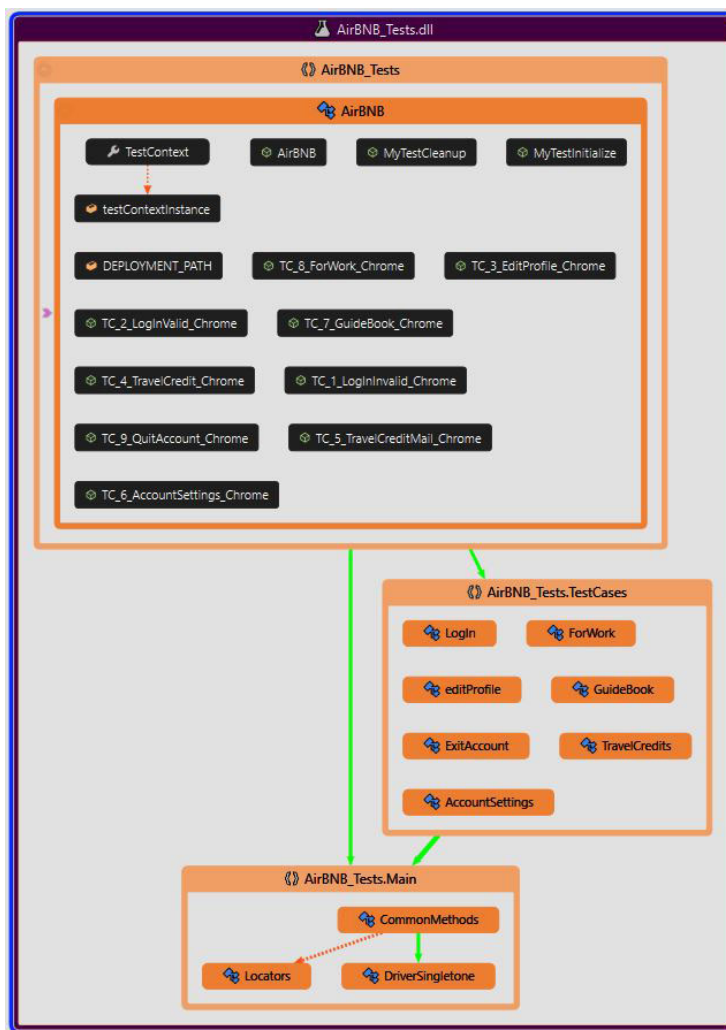


Рисунок 1 – Структура проекта

В Main хранятся основные общие классы фреймворка, а в TestCases – непосредственно сами тестовые сценарии. Пространство имен Data предназначено для хранения xml-файла с данными, которые будут парситься в программный код с помощью метода ParseData. В нашем конкретном случае – это тэг Email с адресом электронной почты для входа в аккаунт пользователя через метод fillInEmail. На рисунке 2 приведен файл parameters.xml.

```

Parameters.xml  AirBNB.cs
1  <?xml version="1.0" encoding="utf-8" ?>
2  <Config>
3  <Email>diplomaProj@hotmail.com</Email>
4  </Config>
5
    
```

Рисунок 2 – Файл parameters.xml

Для работы были установлены необходимые браузеры: Mozilla Firefox и Google Chrome. Это наиболее популярные браузеры последних версий.

В проект добавлены библиотеки Selenium WebDriver, Microsoft.TestApi, Selenium.Support, а также драйверы для браузеров Google Chrome и Mozilla Firefox с помощью менеджера пакетов NuGet для .NET. Установленные пакеты показаны на рисунке 3.

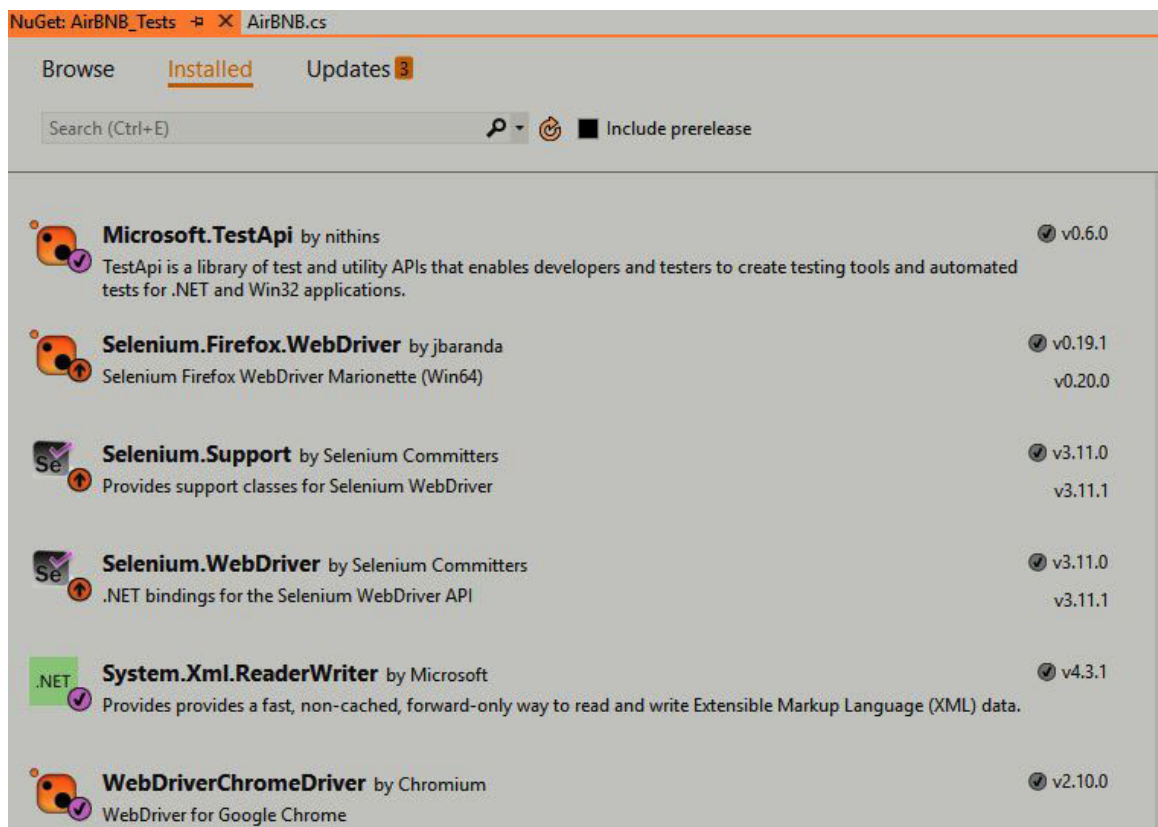


Рисунок 3 – Установленные пакеты из NuGet Package

NuGet – это бесплатный менеджер пакетов с открытым кодом. Он представлен как расширение для Visual Studio. NuGet можно также использовать через командную строку.

Следует отметить, что библиотека `Microsoft.TestApi` хороша тем, что содержит ряд полезных методов для улучшения работы фреймворка, а библиотека `Selenium.Support` – ряд важных классов поддержания работы Selenium WebDriver для .NET.

Основополагающими классами проекта являются: `Locators`, `DriverSingleton`; `CommonMethods`; `AirBNB`. Схематично взаимодействия данных классов представлены на рисунке 4.

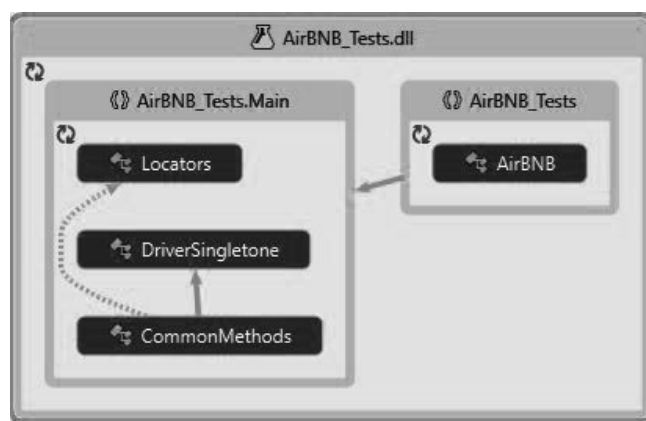


Рисунок 4 – Схема основных классов

Класс `Locators` содержит параметры поиска искомых объектов веб-приложения.

Класс `DriverSingleton` отвечает за вызов сессии браузеров и их закрытие. Здесь обозначены все основные параметры браузеров, а также определен ряд настроек этих браузеров. Для данного класса определены следующие методы: `getDriver_Chrome()`; `getDriver_Mozilla()`; `closeAllDrivers()`.

Методы `getDriver_Chrome()` и `getDriver_Mozilla()` вызывают и запускают сессии необходимых нам браузеров – Google Chrome и Mozilla Firefox соответственно. Метод `closeAllDrivers()` при его вызове закрывает все запущенные браузеры, если они открыты и очищает дерево процесса драйвера браузеров в операционной системе. На рисунке 5 приведен фрагмент класса `DriverSingleton`.

```

8      99+ references
9      public class DriverSingleton
10     {
11         #region Chrome
12         private static string CHROME_DRIVER = "webdriver.chrome.driver";
13         private static string CHROME_DRIVER_PATH = "E:\\MIU\\2018\\Spring\\Diploma\\Project\\AirBNB_Tests\\" +
14             "packages\\WebDriverChromeDriver.2.10\\tools";
15
16         #endregion
17
18         #region Mozilla
19         private static FirefoxDriverService MOZILA_DRIVER_PATH =
20             FirefoxDriverService.CreateDefaultService("E:\\MIU\\2018\\Spring\\Diploma\\Project\\AirBNB_Tests\\" +
21                 "packages\\Selenium.Firefox.WebDriver.0.19.1\\driver");
22
23         #endregion
24
25         #region WebDriver variables declaration
26         public static IWebDriver ChromeDriver;
27         public static IWebDriver MozillaDriver;
28         #endregion
29
30         //Call Chrome driver session
31         1 reference
32         public static IWebDriver getDriver_Chrome()
33         {
34             //Set as environment variable
35             System.Environment.SetEnvironmentVariable(CHROME_DRIVER, CHROME_DRIVER_PATH);
36
37             if (ChromeDriver == null)
38             {
39                 ChromeDriver = new ChromeDriver(CHROME_DRIVER_PATH);
40                 ChromeDriver.Manage().Timeouts().PageLoad = TimeSpan.FromSeconds(30);
41                 ChromeDriver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(10);
42                 ChromeDriver.Manage().Window.Maximize();
43             }
44             return ChromeDriver;
45         }
46     }

```

Рисунок 5 – Фрагмент класса DriverSingleton

Относительно данного класса следует сделать несколько пояснений. В методах вызова браузеров нам необходимо дополнительно указывать пути исполняемого файла браузера Firefox, а также пути нахождения самого WebDriver и драйверов для браузеров. Отдельно следует добавить системную переменную для драйвера Chrome. Для драйвера Firefox это делается вручную.

Класс *CommonMethods* предназначен для хранения основных методов, которые будут вызываться на протяжении всего проекта, то есть общих методов для работы с элементами страницы веб-приложения. Следует учесть, что все методы и переменные в этом классе будут static, для того чтобы иметь возможность напрямую обратиться к ним, не прибегая к ссылкам на экземпляр класса. На наш взгляд, это довольно удобно, с учетом того, что быстродействие кода от этого практически не изменится. На рисунке 6 приведен фрагмент данного класса.

Следует отметить, что здесь уже приведены некоторые элементы персонализации теста, с учетом особенностей тестируемого веб-приложения. В целом же, персонализировать вышеупомянутый класс нашего фреймворка под любое конкретное веб-приложение не должно составить особого труда, достаточно лишь переинициализировать должным образом требуемые переменные соответствующих методов, с учетом пользовательского интерфейса логики работы конкретного веб-приложения.

Класс *AirBNB* координирует совместные действия всех классов и методов фреймворка. *AirBNB* – класс с атрибутом `[CodedUITest]` (имя класса не имеет принципиального значения, как правило, это имя проекта, для которого пишутся автоматизированные тесты).

Вышеназванный атрибут `[CodedUITest]` используется для того, чтобы четко указать, что мы будем использовать инструмент автоматизированного тестирования Coded UI Test. По ходу выполнения проекта указываем, где это необходимо, тайм-аут, для того чтобы приложение выполнилось до наступления определенного условия.

Остановимся более подробно на некоторых методах класса *CommonMethods*).

Метод `LoadApp(IWebDriver currentDriver)` используется для загрузки веб-приложения, при условии передачи ему в качестве параметра драйвера (браузера), который мы хотим использовать.

Метод `CheckoutExistence(int Expected, IList ActualSize, string message)` использован для проверки наличия на странице приложения какого-то конкретного объекта, причем абсолютно не важно, какого типа этот объект – чекбокс, выпадающий список, кнопка или просто текст. Но выполняется этот метод при условии передачи ему параметра размера коллекции, количества объектов самой коллекции, а также текстового сообщения, которое

будет выведено в отчет в случае, если ожидаемый результат не соответствует действительному.

Все остальные методы более индивидуальны. Они могут быть применены для веб-приложений, в которых присутствуют страницы аккаунтов (кабинетов) пользователей, таких как настройки аккаунта, редактирование аккаунта и прочего.

Каждый отдельный тест определяется атрибутом [TestMethod] для того, чтобы указать и четко

разграничить определенный тестовый сценарий и, впоследствии, сгенерировать отчет по результатам выполненного теста.

Все отчеты о результатах тестового запуска собираются в папку TestResults в виде .html страницы, а также с выполнением скриншота экрана монитора на момент завершения тестового метода.

```
public class CommonMethods
{
    #region Variable declaration
    static IWebElement logInBtn;
    static IWebElement MyAccount;
    static IWebElement EditBtn;
    static IWebElement WorkTravelBtn;
    static IWebElement AccSettingsBtn;
    static IWebElement GuidebookBtn;
    static IWebElement ForWork;
    #endregion

    //Parse xml
    #region TAGS
    private const string EMAIL_TAG = "Email";
    #endregion

    1 reference
    public static string ParseData()
    {
        XmlDocument docParams = new XmlDocument();
        XmlNodeList eMail;
        docParams.Load("Parameters.xml");
        XmlElement rootElement = docParams.DocumentElement;
        eMail = rootElement.GetElementsByTagName(EMAIL_TAG);
        string email = eMail[0].InnerText;

        return email;
    }

    //Go to https://www.airbnb.com/
    2 references | 0/2 passing
    public static void LoadApp(IWebDriver currentDriver)
    {
        if (currentDriver == DriverSingleton.ChromeDriver)
        {
            DriverSingleton.getDriver_Chrome();
            DriverSingleton.ChromeDriver.Navigate().GoToUrl("https://www.airbnb.com/");
        }

        else //(currentDriver == DriverSingleton.MozillaDriver)
        {
            DriverSingleton.getDriver_Mozilla();
            DriverSingleton.MozillaDriver.Navigate().GoToUrl("https://www.airbnb.com/");
        }
    }
}
```

Рисунок 6 – Фрагмент класса CommonMethods

## 2. Обобщенный алгоритм работы автоматизированного тест-кейса

Схематично обобщенный алгоритм работы автоматизированного тест-кейса представлен на рисунке 7.

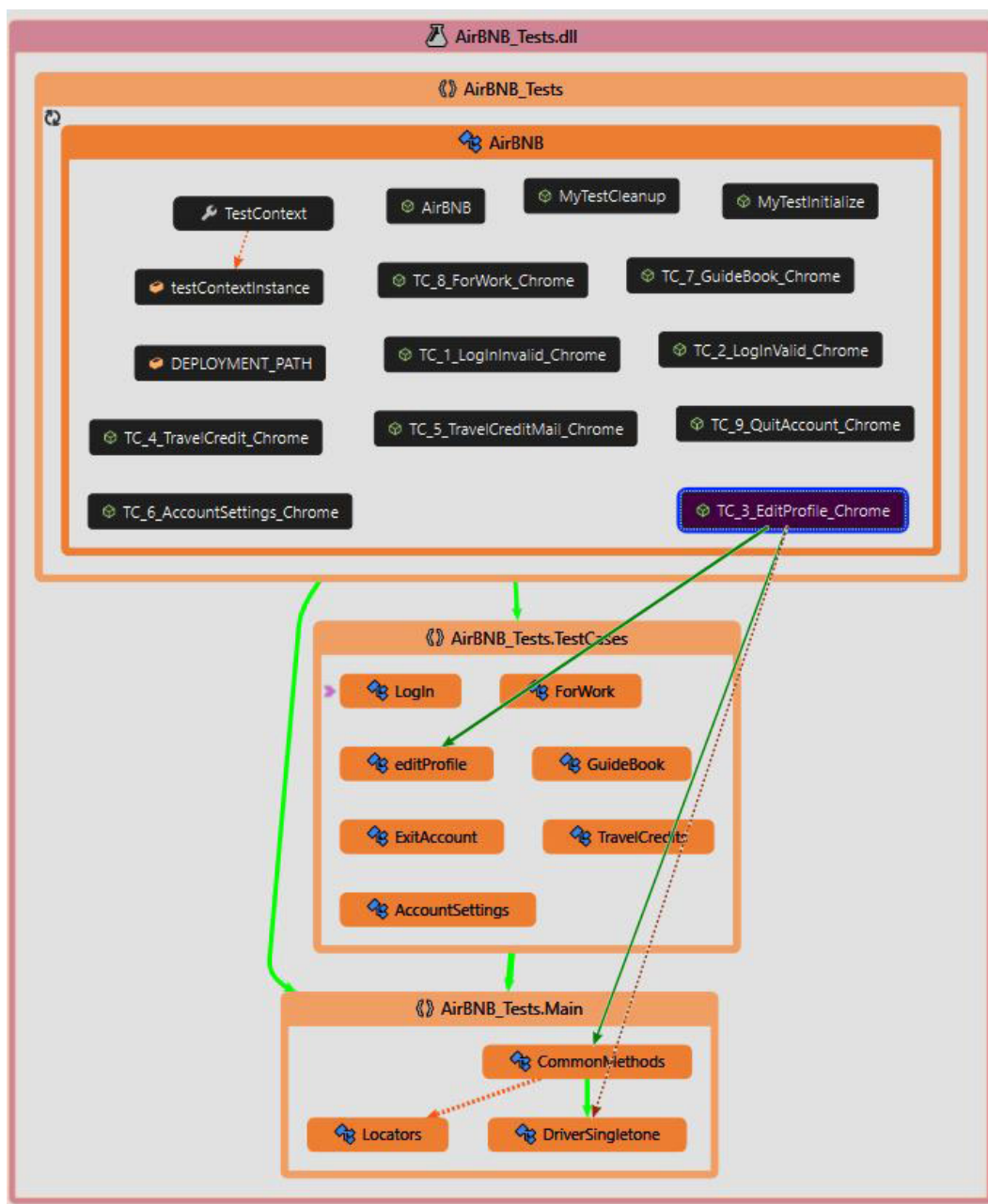


Рисунок 7 – Обобщенный алгоритм работы автоматизированного тест-кейса

Данный алгоритм представляет собой взаимодействие следующих компонентов системы:

- *реального браузера*, работу которого мы автоматизируем;
- *драйвера браузера*, который используется для управления браузером. Этот драйвер представляет собой веб-сервер, в функции которого входит запуск браузера, отправка команд браузеру и закрытие его. Для каждого браузера предназначен свой драйвер, поскольку каждый браузер характеризуется своим набором команд и своей реализацией;
- *самого теста*, который содержит набор команд на языке программирования C# для драйвера браузера.

В классе DriverSingleton.cs с помощью методов

getDriver мы вызываем необходимый браузер, разворачиваем его и загружаем индексную страницу веб-приложения. После чего даем команду подождать, до полной загрузки веб-приложения. Далее обращаемся к классу AirBNB.cs, а в нем – к необходимому нам методу, например, к методу TC\_3\_EditProfile\_Chrome. Из этого метода обращаемся к соответствующему классу editProfile и методу конкретного тестового сценария, где определены параметры поиска объектов приложения, а также вызываются необходимые методы из ранее упомянутого класса CommonMethods.cs.

По завершению тестового сценария автоматически генерируется html-отчет. Отчет настраивается в файле app.config нашего проекта.

Файл app.config представлен на рисунке 8.

```

app.config -# X AirBNB.cs
1 <?xml version="1.0" encoding="utf-8" ?>
2 <configuration>
3   <system.diagnostics>
4     <switches>
5       <add name="EqTraceLevel" value="4" />
6     </switches>
7   </system.diagnostics>
8   <appSettings>
9     <add key="StopTestRunCallTimeoutInSeconds" value="5"/>
10    <add key="LogSizeLimitInMegs" value="20"/>
11    <add key="CreateTraceListener" value="yes"/>
12    <add key="GetCollectorDataTimeout" value="300"/>
13    <add key="EnableSnapshotInfo" value="no"/>
14    <add key="EnableHttpLogger" value="no"/>
15  </appSettings>
16 </configuration>

```

Рисунок 8 – Файл app.config

Пример отчета о результатах тестирования приведен на рисунке 9.

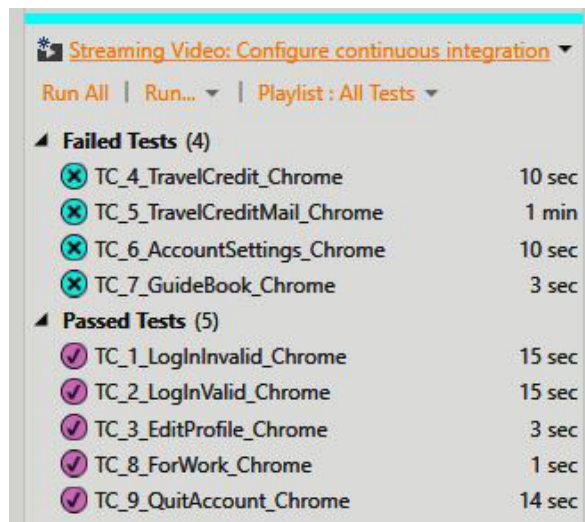


Рисунок 9 – Отчет о результатах тестирования

Пример проваленного отчета в формате html-документа представлен на рисунке 10.

#### Coded UI Test Log

```

- [X] TC_5_TravelCreditMail_Chrome
  - [X] Assert.AreEqual
    - [X] Test method TC_5_TravelCreditMail_Chrome threw exception:
      Assert.AreEqual failed. Expected:<1>. Actual:<0>. No "Share your love..." text exists
      Microsoft.VisualStudio.TestTools.UnitTesting.AssertFailedException
        at Microsoft.VisualStudio.TestTools.UnitTesting.Assert.HandleFailure(String assertionName, String message)
        at Microsoft.VisualStudio.TestTools.UnitTesting.Assert.AreEqual(T expected, T actual, String message, Object[] parameters)
        at Microsoft.VisualStudio.TestTools.UnitTesting.Assert.AreEqual(T expected, T actual, String message)
        at AirBNB_Tests.Main.CommonMethods.CheckOutExistence(Int32 Expected, IList ActualSize, String message)
          in e:\MIU\2018\Spring\Diploma\Project\AirBNB_Tests\AirBNB_Tests\Main\CommonMethods.cs:line 68
        at AirBNB_Tests.TestCases.TravelCredits.travelCreditsPage()
          in e:\MIU\2018\Spring\Diploma\Project\AirBNB_Tests\AirBNB_Tests\TestCases\TravelCredit.cs:line 34
        at AirBNB_Tests.AirBNB.TC_5_TravelCreditMail_Chrome()
          in e:\MIU\2018\Spring\Diploma\Project\AirBNB_Tests\AirBNB_Tests\Main\AirBNB.cs:line 71

```

Рисунок 10 – Проваленный отчет о результатах тестирования



Работа фреймворка тестировалась на примере веб-приложения `airbnb.com` – в его части, связанной с аккаунтом пользователя. Был составлен ряд тестовых сценариев поведения пользователей, действия которых симулировали автоматизированные тесты.

После прохождения тестов был сгенерирован отчет в формате `html`-страницы. Тестирование проведено успешно.

### Заключение

В данной работе представлена методика разработки фреймворка автоматизации функционального тестирования веб-приложений. Преимуществом предложенной методики является разграничение общих классов и методов от классов и методов конкретных исполняемых тест-кейсов, а также конфигурационных файлов. Это позволило реализовать универсальный фреймворк автоматизации функционального тестирования практически любых веб-приложений, с помощью которого можно: быстро и легко начать автоматизацию тестовых сценариев; реализовать возможность поддержания автоматизированных тестов

в будущем в актуальном состоянии; поддерживать тестовые сценарии в работоспособном состоянии с учетом изменения интерфейса веб-приложения; добавлять любые другие инструменты и фреймворки в виде библиотек классов; выполнять многократные тестирования без дополнительных затрат.

Внедрение разработанного фреймворка автоматизации функционального тестирования веб-приложений позволит решить следующие задачи: исполнение мануальных тест-кейсов в автоматическом режиме; увеличение скорости процесса тестирования; уменьшение количества трудовых ресурсов в ходе разработки веб-приложения; исключение человеческого фактора; возможность переключения ресурсов команды тестирования на тестирование новой функциональности или более глубокого тестирования существующей.

Все вышеперечисленное приводит к повышению уровня доверия к веб-приложению и его успешному развертыванию.

## ЛИТЕРАТУРА / REFERENCES

1. World Quality Report 2017-18 – 9TH EDITION [Electronic resource]. – Mode of access: [https://www.sogeti.com/globalassets/global/downloads/testing/wqr-2017-2018/wqr\\_2017\\_v9\\_secure.pdf](https://www.sogeti.com/globalassets/global/downloads/testing/wqr-2017-2018/wqr_2017_v9_secure.pdf). – Date of access: 15.09.2018.
2. Куликов, С.С. Тестирование программного обеспечения. Базовый курс / С.С. Куликов. – Минск: Четыре четверти, 2017. – 312 с.  
Kulikov, S.S. Testirovanie programmnoho obespecheniya. Bazovyy kurs / S.S. Kulikov. – Minsk: CHetyre chetverti, 2017. – 312 p.
3. Расолько, А.М. Фреймворк автоматизации функционального тестирования веб-приложений / А.М. Расолько // Человек, психология, экономика, право, управление: проблемы и перспективы : материалы XXI Междунар. науч. конф. аспирантов, магистрантов и студентов, г. Минск, 18 мая 2018 г. / Минский инновационный университет; под ред. канд. пед. наук В.В. Гедранович. – Минск: Минский инновационный университет, 2018. – С. 51–52.  
Rasol'ko, A.M. Frejmvork avtomatizacii funkcional'nogo testirovaniya веб-prilozhenij / A.M. Rasol'ko // CHelovek, psihologiya, ekonomika, pravo, upravlenie: problemy i perspektivy : materialy XXI Mezhdunar. nauch. konf. aspirantov, magistrantov i studentov, g. Minsk, 18 maya 2018 g. / Minskij innovacionnyj universitet; pod red. kand. ped. nauk V.V. Gedranovich. – Minsk: Minskij innovacionnyj universitet, 2018. – P. 51–52.