Программное решение Cloud_Pricing_Calculator_website_UI_testing. Техническое описание

1. Общие сведения

Программное решение "Cloud_Pricing_Calculator_website_UI_testing", далее – "программное решение", "ПР", разработано в учебных целях и не предназначено для какого-либо коммерческого использования.

Настоящее ПР предназначено для частичной автоматизации тестирования пользовательского интерфейса (UI), фронтенд части веб-сайта онлайн оценки стоимости использования оборудования для облачных вычислений сервиса Google Cloud – "Google Cloud Pricing Calculator".

Программное решение выполнено в интегрированной среде разработки IntelliJ IDEA на языке программирования Java с использованием библиотеки автоматизированного управления веб-браузерами "Selenium WebDriver" и фреймворка для автоматизации тестирования программного обеспечения "TestNG"; сборщиком проекта выступает Maven. В качестве рабочего веб-браузера используется "Chrome" и утилита доступа и взаимодействия с ним "ChromeDriver".

2. Структура программного решения

"Cloud_Pricing_Calculator_website_UI_testing" имеет иерархическую структуру в которой модули ПР разделены на общую "main" и тестовую "test' части.

В общем "main" модуле расположена вся логика работы программного решения: классы, описывающие веб-элементы графического интерфейса, представленных в DOM принимаемых веб-страниц с которыми взаимодействует ПР, а также классы и методы для обработки и действий над поступающей информацией.

В тестовом "test" модуле расположены классы, описывающие тестовые сценарии по которым происходит автоматизированная работа веб-браузера. Исходные данные с конфигурацией оборудования для расчёта стоимости, путь расположения к утилите вебдрайвера находятся в классе так же в этом модуле.

Отдельно, в корневом каталоге ПР, распложены xml-файлы, описывающие тестовые последовательности для TestNg и внешние зависимости для конфигурирования и сборки pom проекта для Maven.

Структура каталогов программного решения представлена ниже:

```
Cloud_Pricing_Calculator_website_UI_testing
    src
        main
            java
                pageObjects
                utils
        test
            java
                parentTestClass
                smokeTestScenarios
                testDataAndListeners
```

3. Описание тестовых классов и последовательности работы тестовых сценариев программного решения

В настоящее программное решение включено 2 тестовых сценария.

Тестовые сценарии расположены в каталоге "smokeTestScenarios", описанные в тестовых классах:

- CalculatePriceTest – тестовый сценарий для быстрого смоук-тестирования функциональности подбора конфигурации оборудования, расчёта стоимости и проверки корректности передачи результирующей стоимости на адрес электронной почты стороннего сервиса;

- SimpleFailedTest – автоматизирует вертикальную прокрутку открытой веб-страницы в различные положения, а так же включает в себя метод со специально заданным "упавшим" тестом для проверки работы различных возможностей Selenium WebDriver и TestNG.

Все тестовые сценарии являются логически взаимосвязанными наборами законченных тест-кейсов и могут запускаться как по отдельности из своих тестовых классов, так и в составе – с учётом последовательностей выполнения, описанных в xml-файле тестового фреймворка TestNG.

Применяемые в данном программном решении аннотации @BeforeSuite, @BeforeTest, @BeforeClass, @Test, @AfterClass, @AfterTest, @Listener являются аннотациями фреймворка TestNG если не указано иное.

3.1 Базовый тестовый класс TestBasics

Все вышеперечисленные тестовые сценарии описаны в одноимённых .java классах и являются производными классами от базового класса TestBasics.java, находящегося в каталоге "parentTestClass".

Базовый класс TestBasics.java в своём составе имеет поля и методы, общие для производных от него классов тестовых сценариев и содержит:

- статическое поле *driver* типа WebDriver, предназначенное для передачи вебдрайвера в используемые экземпляры тестовых классов;

- статические поля *browserType, driverPath, searchQuery* типа String предназначены для задания значений типа веб-браузера, пути расположения утилиты вебдрайвера и поискового запроса соответственно;

- статические строковые поля, принимающие значения конфигурации оборудования *provisioningModel, instanceType, region, localSSD, commitmentTerm*.

Метод *somePreparingActions()*, подписанный аннотацией @BeforeSuite, предназначен для вывода информации о старте работы тестового набора.

Для возможности реагирования на различные события, происходящие во время работы тестовых сценариев, базовый класс подписан аннотацией @Listeners ("слушатель и" тестовый и выполнения). Подробнее о "слушателях" состояний тестов указано в соответствующем разделе

3.2 Описание тестовых сценариев для смоук-тестирования в каталоге "smokeTestScenarios"

3.2.1 CalculatePriceTest

Тестовый сценарий CalculatePriceTest описан в классе CalculatePriceTest.java и представляет собой сценарий поведения пользователя, который хочет рассчитать стоимость интересующей его конфигурации оборудования для облачных вычислений и включает в себя следующие шаги:

- открытие стартовой страницы поиска веб-сайта сервиса Google Cloud;

- написание запроса "Google Cloud Pricing Calculator" и просмотр результатов поиска;

- выбор наиболее релевантного результата и переход на страницу калькулятора;

- задание конфигурации оборудования в подразделе "Compute Engine" и расчёт стоимости;

- проверка выбранной конфигурации и сверка общего результата с суммой цен позиций конфигурации;

- выбор почтового сервиса для отправки электронного письма о рассчитанной стоимости оборудования;

- получение электронного письма от сервиса с рассчитанной стоимостью заданной конфигурации и проверка почтового ящика со сверкой результатов стоимости на сайте сервиса и в письме.

Тестовый сценарий подразумевает работу в сервисах двух различных интернет-сайтов: непосредственно на веб-странице сервиса расчёта стоимости Google Cloud и на веб-странице сервиса временных адресов электронной почты YOPmail. Работа на обоих сайтах происходит в двух различных вкладках интернет-браузера, поэтому в составе класса есть поля типа String для хранения идентификаторов открытых вкладок (*initialBrowserWindow, secondBrowserWindow*).

В класс включены поля типов, описывающих веб-элементы DOM-структуры соответствующих веб-страниц, принимаемых веб-браузером во время работы тестового сценария: *startSearchPage, pricingCalculatorPage, estimateSubPage, yopmailStartPage, incomingEmailsPage*. Классы, описывающие данные поля находятся в каталоге "pageObjects" модуля "main".

Метод *activateTestPages()*, подписанный аннотацией @BeforeTest, предназначен для активации всех веб-элементов тех веб-страниц, на которых проходит тестовый сценарий и получения идентификатора изначально открытой вкладки веб-браузера (*initialBrowserWindow = driver.getWindowHandle()*). Также данный метод выводит информацию о состоянии веб-драйвера перед началом тестирования.

Метод *enterToCloudSearchPage()*, подписанный аннотацией @BeforeClass, предназначен для открытия страницы поиска веб-сайта сервиса Google Cloud и содержит следующую последовательность инструкций:

- *driver.get("адрес_сайта_поиска")* – переход на веб-страницу поиска сервиса Google Cloud для начала тестирования;

- *Assert.assertEquals(driver.getTitle(), "Search | Google Cloud")* – проверка соответствия заголовка открытой веб-страницы ожидаемому названию " *Search | Google Cloud* ".

Тестовый метод *searchingRequiredContents()*, подписанный аннотацией @Test, предназначен для поиска необходимого сервиса через поле поиска и открытия ссылки на веб-страницу с наиболее релевантным результатом. Тест-кейс состоит из следующих шагов, описанных в последовательности инструкций:

Предусловие: открыта веб-страница поиска сервиса Google Cloud.

- *startSearchPage.findNeedInfo(searchQuery)* – в поле для поиска ввести запрос ("Google Cloud Pricing Calculator");

- *startSearchPage.chooseSearchingResutt()* – в блоке полученных результатов выбрать результат, ссылка которого в своём составе имеет путь "calculator-legacy" и перейти по этой ссылке

- *Assert.assertEquals(driver.getCurrentUrl(), "адрес_сайта_калькулятора")* – проверка соответствия открытой после перехода веб-страницы с требуемым веб-адресом калькулятора.

Тестовый метод *setupEquipmentConfiguration()*, подписанный аннотацией @Test, предназначен для проверки функциональности работы калькулятора при задании определённой конфигурации интересующего оборудования. Тест-кейс состоит из следующих шагов, описанных в последовательности инструкций:

Предусловие: открыта веб-страница калькулятора расчёта стоимости сервиса Google Cloud.

- *Assert.assertEquals(driver.getTitle(), "Google Cloud Pricing Calculator")* – проверка соответствия заголовка текущей открытой веб-страницы ожидаемому заголовку;

- *pricingCalculatorPage.pressComputeEngineButton()* – выбор конфигуратора оборудования. В данном тестовом сценарии необходимый конфигуратор оборудования находится в подразделе "COMPUTE ENGINE" блока с горизонтальной прокруткой вариантов оборудования для различных целей который находится в отдельном фрейме на открытой веб-странице;

- *pricingCalculatorPage.caclulatePrice()* – после выбора необходимой конфигурации оборудования (поля Number of instances, Operating System, Provisioning model, Series, Machine type, чекбокс Add GPUs, GPU type, Number of GPUs, Local SSD, Datacenter location, Committed usage) нажатие кнопки "ADD TO ESTIMATE".

Тестовый метод *validationOfRequirements(),* подписанный аннотацией @Test, предназначен для проверки фактически рассчитанного оборудования на соответствие требуемой конфигурации. Тест-кейс состоит из следующих шагов, описанных в последовательности инструкций:

Предусловие: открыта веб-страница калькулятора расчёта стоимости сервиса Google Cloud, задана требуемая конфигурация оборудования, расчёт стоимости выполнен.

- *estimateSubPage.getEquipmentResults()* – получение результирующих значений конфигурации и стоимости, вывод на консоль результатов рассчитанной конфигурации. После выбора требуемой конфигурации оборудования и нажатия на кнопку "ADD TO ESTIMATE" в правой части открытой веб-страницы калькулятора появляется боковое подменю "Estimate" с результатами рассчитанной стоимости выбранной конфигурации;

- *Assert.assertTrue(estimateSubPage.provisioningModel.contains(provisioningModel))* – проверка соответствия параметра "Provisioning model" требуемому значению Regular;

- *Assert.assertTrue(estimateSubPage.instanceType.contains(instanceType))* – проверка соответствия параметра "Instance type" требуемому значению n1-standard-8;

- *Assert.assertTrue(estimateSubPage.region.contains(region))* – проверка соответствия параметра "Region" требуемому значению Frankfurt;

- *Assert.assertTrue(estimateSubPage.localSSD.contains(localSSD))* – проверка соответствия параметра "Local SSD" требуемому значению 2x375 GiB;

- *Assert.assertTrue(estimateSubPage.commitmentTerm.contains(commitmentTerm))* – проверка соответствия параметра "Commitment term" требуемому значению 1 Year.

Тестовый метод *estimatePrice(),* подписанный аннотацией @Test, предназначен проверки корректности результата расчёта стоимости требуемой конфигурации оборудования.Тест-кейс состоит из следующих шагов, описанных в последовательности инструкций:

Предусловие: открыта веб-страница калькулятора расчёта стоимости сервиса Google Cloud, задана требуемая конфигурация оборудования, расчёт стоимости выполнен.

- *estimateSubPage.getPricesOfComponentsAndTotalCost()* – получение значений стоимости компонентов по отдельности и полной стоимости требуемой конфигурации оборудования из открывшегося правого бокового меню. В поле "Instance type" содержится стоимость выбранного оборудования без учёта стоимости накопителей, в поле "Local SSD" отображена стоимость использования накопителей. Поле

"Total Estimated Cost" содержит общую сумму стоимости использования оборудования в месяц. На консоль выводится сумма первых двух значений;

- *Assert.assertEquals(estimateSubPage.totalCost, estimateSubPage.componentsCostSum)* – проверка равности значений из поля "Total Estimated Cost" с суммой значений полей "Instance type" и "Local SSD";

- *estimateSubPage.highlightPrices()* – выделение цветом значений стоимости полей "Total Estimated Cost", "Instance type" и "Local SSD" в открытом окне веб-браузера;

- *ScreenshotMaker.screenCaptureDuringTest(driver)* – создание скриншота открытой веб-страницы с выделенными цветом значениями вышеназванных полей. Скриншот сохраняется в каталоге размещения данного программного решения по адресу: ".//target/screenshots/DuringTest/каталог_с_текущей_датой/" в растровом графическом формате PNG в файле со структурой "HH-mm-ss.png" в которой паттерн "HH-mm-ss" обозначает текущее время на момент вызова метода формирования скриншота в формате "час-минута-секунда".

Тестовый метод *getEmailAddress(),* подписанный аннотацией @Test, предназначен для перехода на сервис временных адресов электронной почты YOPmail и создания такого адреса. Тест-кейс состоит из следующих шагов, описанных в последовательности инструкций:

Предусловие: открыта веб-страница калькулятора расчёта стоимости сервиса Google Cloud, задана требуемая конфигурация оборудования, расчёт стоимости выполнен.

- *driver.switchTo().newWindow(WindowType.TAB)* – открытие новой вкладки в веб-браузере;

- *driver.get("https://yopmail.com")* – открытие веб-страницы сервиса временных адресов электронной почты YOPmail;

- *Assert.assertTrue(driver.getTiele().contains("YOPmail"))* – проверка заголовка открытой веб-страницы;

- *secondBrowserWindow = driver.getWindowHandle()* – получение идентификатора текущей открытой вкладки браузера;

- *yopmailStartPage.generaneNewEmail()* – создание нового временного адреса электронной почты. Нажатие на кнопку "Случайный адрес электронной почты" для генерации временного адреса электронной почты, закрытие всплывшего окна (попап) с рекламой, нажатие кнопки "Скопировать в буфер обмена".

Тестовый метод SendEmailAndValidatePrice(), подписанный аннотацией @Test, предназначен для отправки электронного письма с рассчитанной ценой конфигурации оборудования на адрес электронной почты и дальнейшей проверки присланного в этом письме значения стоимости. Тест-кейс состоит из следующих шагов, описанных в последовательности инструкций:

Предусловие: открыта веб-страница калькулятора расчёта стоимости сервиса Google Cloud, задана требуемая конфигурация оборудования, расчёт стоимости выполнен, известен адрес электронной почты на который будет отправлено электронное письмо.

- *driver.switchTo().window(initialBrowserWindow)* – переключение на вкладку веб-браузера с открытой страницей сайта Google Cloud Pricing Calculator;

- *estimateSubPage.sendEmailWithPrice()* – отправка электронного письма с результатам расчёта стоимости на электронную почту. Нажатие на кнопку "Email Estimate" в правом боковом меню. Во всплывшем окне "Email Your Estimate" в поле "Email" вставить сгенерированный адрес электронной почты. Нажатие на кнопку "SEND EMAIL";

- *driver.switchTo().window(secondBrowserWindow)* – переключение на вкладку веб-браузера с открытой страницей сервиса YOPmail;

- *driver.navigate().refresh()* – обновление открытой веб-страницы;

- *yopmailStartPage.goToEmailPage()* – переход на веб-страницу адреса электронной почты путём нажатия на кнопку "Проверить почту";

- *ScreenshotMaker.screenCaptureDuringTest(driver)* – создание скриншота открытой веб-страницы. Скриншот сохраняется в каталоге размещения данного программного решения по адресу:

".//target/screenshots/DuringTest/каталог_с_текущей_датой/" в растровом графическом формате PNG в файле со структурой "HH-mm-ss.png" в которой паттерн "HH-mm-ss" обозначает текущее время на момент вызова метода формирования скриншота в формате "час-минута-секунда";

- *incomingEmailsPage.getPriceFromEmail()* – просмотр входящего письма от сервиса Google Cloud Pricing Calculator. Вывод на консоль строки "Estimated Monthly Cost" со значением рассчитанной стоимости, выделение того значения из полученной строки;

- *Assert.assertEquals(incomingEmailsPage.receivedPrice, estimateSupPage.totalCost)* – оценка равенства полученного в письме значения стоимости с рассчитанной в калькуляторе сервиса Google Cloud.

Метод *closeBrowserTab()*, подписанный аннотацией @AfterClass, предназначен закрытия второй вкладки веб-браузера и перехода на первую.

Метод *endOfSession()*, подписанный аннотацией @AfterTest, предназначен для выгрузки экземпляра утилиты вебдрайвера. После данного метода происходит завершение работы тестового класса CalculatePriceTest.

3.2.2 SimpleFiledTest

Тестовый сценарий SimpleFiledTest описан в тестовом классе SimpleFiledTest.java и представляет собой сценарий вертикальной прокрутки открытой веб-страницы сервиса Google Cloud Pricing Calculator в различных положениях с одним реализованным тест-кейсом с заранее неудавшимся результатом для проверки работы "слушателя" (листенера) при "упавшем" тестовом методе. Тестовый сценарий включает в себя следующие шаги:

- открытие веб-страницы сервиса Google Cloud Pricing Calculator;
- проверка нахождения на открытой странице веб-элемента;
- выделение цветом этого веб-элемента;
- вертикальная прокрутка открытой веб-страницы в различные положения:
  - на 1500 пикселей вниз относительно верха страницы;
  - ещё на 100 пикселей вниз относительно текущего положения;
  - на 150 пикселей относительно верха страницы;
  - вниз страницы;
  - вверх страницы;
- вертикальная прокрутка вниз до искомого веб-элемента;
- попытка подсчитать стоимость без ввода конфигурации – "упавший" тестовый метод.

В класс включено поле типа *PricingGalculatorPage*, описывающие веб-элементы DOM-структуры соответствующей веб-страницы, принимаемой веб-браузером во время работы тестового сценария. Класс, описывающий данное поле, находится в каталоге "pageObjects" модуля "main". Подробное описание классов данных полей приведено далее в соответствующем разделе.

Метод *activateTestPage()*, подписанный аннотацией @BeforeTest, предназначен для активации всех веб-элементов веб-страницы, участвующей в тестовом сценарии. Также данный метод выводит информацию о состоянии веб-драйвера перед началом тестирования.

Метод *enterToPage()*, подписанный аннотацией @BeforeClass, предназначен для открытия страницы калькулятора для подсчёта стоимости сервиса Google Cloud и содержит следующую последовательность инструкций:

- *driver.get("адрес_сайта_сервиса")* – переход на веб-страницу поиска сервиса Google Cloud для начала тестирования;
- *Assert.assertEquals(driver.getTitle(), "Google Cloud Pricing Calculator")* – проверка соответствия заголовка открытой веб-страницы ожидаемому названию "Google Cloud Pricing Calculator ".

Тестовый метод *pageScrolling()*, подписанный аннотацией @Test, предназначен для проверки возможности вертикальной прокрутки открытой страницы по нескольким различным ориентирам. Тест-кейс состоит из следующих шагов, описанных в последовательности инструкций:

Предусловие: открыта веб-страница калькулятора расчёта стоимости сервиса Google Cloud Pricing Calculator.

- *WebElement footerSolutions = driver.findElement (By.xpath "строка_xpath_искомого_веб-элемента")* – поиск в DOM открытой страницы веб-элемента с заголовком "Solutions", находящегося в футере, не описанного в классе PricingCalculatorPage;
- *boolean assertion = footerSolutions.isEnabled();*
- *Assert.assertTrue(footerSolutions.isEnabled)* – проверка наличия искомого веб-элемента на открытой веб-странице; дальнейший вывод на консоль сообщения о наличии данного веб-элемента;
- *JavaScriptMethods.elementsHighlighter(footerSolutions)* – выделение цветом искомого элемента;
- *JavaScriptMethods.pageScroller()* – вертикальная прокрутка открытой веб-страницы на 1500 пикселей вниз относительно верха, дополнительная прокрутка на 100 пикселей вниз, прокрутка на 150 пикселей вниз относительно верха страницы, прокрутка в низ страницы, прокрутка страницы в изначальное положение на верх.

Тестовый метод failedTest(), подписанный аннотацией @Test, предназначен для представления работы "слушателя" состояния путём "падения" теста. Тест-кейс состоит из следующих шагов, описанных в последовательности инструкций:

Предусловие: открыта веб-страница калькулятора расчёта стоимости сервиса Google Cloud Pricing Calculator.

- *pricingCalculatorPage.pressComputeEngineButton()* – нажатие на кнопку "COMPUTE ENGINE " подраздела выбора конфигуратора оборудования в горизонтальном блоке для перехода во фрейм DOM для выбора конфигураций;
- *WebElement addToEstimate = driver.findElement (By.xpath "строка_xpath_искомого_веб-элемента")* – поиск в DOM веб-страницы веб-элемента кнопки кнопку "ADD TO ESTIMATE";
- Вывод на экран консоли сообщения о наличии искомого веб-элемента на странице путём вызова методов "isDiplayed()" и "isEnabled()";
- *addToEstimate.click()* – клик по неактивному искомому веб-элементу;
- *JavaScriptMethods.elementsHighlighter(addToEstimate)* – выделение цветом искомого элемента;
- *new WebDriverWait(driver, Duration.ofSeconds(5)).until (ExpectedConditions.elementToBeClickable(addToEstimete)).click()* – вызов явного ожидания для искомого веб-элемента на предмет его интерактивности. Без выбора конфигурации оборудования кнопка "ADD TO ESTIMATE" остаётся не активна, через 5 секунд тест "падает".

В момент "падения" тестового метода вступает в работу предусмотренный программным решением "слушатель" (@Listener) класса TestListener который делает скриншот состояния открытой веб-страницы. Скриншот сохраняется в каталоге размещения данного программного решения по адресу:

".//target/screenshots/FailedTest/каталог_с_текущей_датой/" в растровом графическом формате PNG в файле со структурой "HH-mm-ss.png" в которой паттерн "HH-mm-ss" обозначает текущее время на момент вызова метода формирования скриншота в формате "час-минута-секунда".

Метод *quitWebDriver()*, подписанный аннотацией @AfterTest, предназначен для выгрузки экземпляра утилиты вебдрайвера. После данного метода происходит завершение работы тестового класса SimpleFailedTest.

3.3 Содержимое каталога "testDataAndListeners"

Каталог "testDataAndListeners" содержит в себе классы ExecutionListener и TestListener, описывающие работу "слушателей" состояния работы тестовых сценариев, а также класс InputTestData, хранящий в себе исходные данные для тестирования.

3.3.1 InputTestData

Класс InputTestData состоит из констант строкового типа, содержащих исходные данные для работы с тестовыми сценариями такие как: задаваемый тип веб-браузера и путь к расположению утилиты вебдрайвера в памяти используемого компьютера; содержание запроса поиска сервиса калькулятора; конфигурация оборудования для расчёта стоимости.

3.3.2 ExecutionListener

Класс "слушателя" ExecutionListener, реализующий интерфейс "IExecutionListener" фреймворка TestNG предназначен для передачи оповещений о начале и завершении работы тестовых сценариев.

Переопределяет методы *onExecutionStart()* и *onExecutionFinish()* для вывода на консоль сообщений с текущей датой и временем наступления соответствующего события.

3.3.3 TestListener

Класс "слушателя" TestListener, реализующий интерфейс "ITestListener" фреймворка TestNG предназначен для передачи оповещения об "упавшем" (неуспешном либо сломанном) тест-кейсе.

Переопределяет метод *onTestFailure()* для создания и сохранения на используемом компьютере скриншота состояния веб-браузера в месте и в момент "падения" теста.

4. Модуль "main". Краткое описание составляющих модуля, используемых для работы тестовых сценариев

Рабочий модуль "main" включает в себя .java классы, обеспечивающие логику программного решения и взаимосвязь выполнения тестов.

В состав модуля "main" входят каталоги `pageObjects` и `utils`.

В каталоге "pageObjects" находятся классы, содержащие локаторы для веб-элементов DOM, принимаемых браузером веб-страниц с которыми происходит взаимодействие во время автоматизированного тестирования веб-сайта и методы с алгоритмами для работы с этими веб-элементами.

Каталог "utils" содержит классы для установки вебдрайвера, создания скриншотов состояния веб-браузера и обработки скриптов на открытых веб-страницах.

4.1 Каталог "pageObjects"

Каталог "pageObjects" содержит в себе файл класса PageObjectBasics и пакеты "emailServicePages", "pricingCalcPages".

Для определения и инициализации веб-элементов подключен PageFactory, входящий в состав библиотеки Selenium. Все используемые в работе веб-элементы (за исключением двух веб-элементов в тестовом сценарии SimpleFailedTest) помечены аннотацией @FindBy. В качестве локаторов для веб-элементов выбрано определение элементов по Xpath как наиболее стабильный и неизменчивый со временем.

4.1.1 PageObjectBasics

Класс PageObjectBasics является суперклассом для всех файлов каталога "pageObjects" и предназначен для задания основных членов, общих для всех дочерних классов в пакетах "emailServicePages" и "pricingCalcPages".

В своём составе суперкласс имеет поля типов WebDriver и Actions из библиотеки Selenium для задания которых составлен конструктор, принимающий на вход экземпляр вебдрайвера. Задание PageFactory для отложенной инициализации вебэлементов также предусмотрено в конструкторе. Методы для работы с явными ожиданиями веб-элементов заданы в этом классе.

4.1.2 emailServicePages

Пакет "emailServicePages" содержит в себе классы, описывающие взаимодействие с веб-страницами сервиса одноразовых адресов электронной почты "YOPMail".

Все классы пакета являются дочерними по отношению к суперклассу PageObjectBasics и наследуют от него все перечисленные в разделе 4.1.1 элементы.

emailServicePages.YopmailStartPage

Класс YpomailStartPage содержит в себе логику для работы на стартовой странице веб-сайта сервиса "YOPMail". Экземпляр, созданный на основе этого класса входит в состав тестового сценария CalculatePriceTest.

В классе предусмотрен метод *closeAdPopup()* для закрытия всплывающего рекламного окна путём имитации нажатия мышки по закрытой попапом области.

emailServicePages.IncomingEmailsPage

Класс IncomingEmailsPage содержит в себе логику для работы на странице входящих писем веб-сайта сервиса "YPOMail". Экземпляр, созданный на основе этого класса входит в состав тестового сценария CalculatePriceTest.

В классе предусмотрен метод *getPriceFromEmail()*, вычленяющий из пришедшего письма строку "Estimated Monthly Cost" рассчитанную стоимость.

4.1.3 pricingCalcPages

Пакет "pricingCalcPages" содержит в себе классы, описывающие взаимодействие со страницами веб-сайта сервиса "Google Cloud".

Все классы пакета являются дочерними по отношению к суперклассу PageObjectBasics и наследуют от него все перечисленные в разделе 4.1.1 элементы.

pricingCalcPages.StartSearchPage

Класс StartSearchPage содержит в себе логику для работы на странице поиска веб-сайта сервиса "Google Cloud". Экземпляр, созданный на основе этого класса входит в состав тестового сценария CalculatePriceTest.

В классе предусмотрен метод *chooseSearchingResult()* для выбора релевантного варианта из результатов поиска с дальнейшим переходом по выбранной ссылке.

pricingCalcPages.PricingCalculatorPage

Класс PricingCalculaqtorPage содержит в себе логику работы на странице калькулятора конфигураций оборудования веб-сайта сервиса "Google Cloud".

В классе предусмотрен метод *pressComputeEngineButton()* для выбора раздела предлагаемого оборудования из горизонтального меню прокрутки для дальнейшего перехода к составлению конфигурации и расчёта стоимости.

pricingCalcPages.EstimateSubPage

Класс EstimateSubPage содержит в себе логику работы в блоке правого бокового меню с результатами расчёта оборудования, появляющегося после нажатия кнопки "ADD TO ESTIMATE".

В классе предусмотрен метод *getPricesOfComponentsAndTotalCost(),* предназначенный для вычленения результатов расчёта общей стоимости использования оборудования и цен по позициям применённой скидки на компоненты конфигурации (Committed Use Discount applied)

4.2 Каталог "utils"

Каталог "utils" содержит в себе файлы классов JavaScriptMethods, ScreenshotMaker и WebDriverSingleton, которые предназначены для вспомогательных действий во время работы программного решения.

4.2.1 JavaScriptMethods

Класс JavaScriptMethods содержит в себе различные методы для работы со скриптами на языке JavaScript, необходимые для выполнения некоторых действий на тестируемых веб-страницах. В частности присутствуют методы для прокрутки открытой веб-страницы и выделения цветом веб-элементов.

4.2.2 ScreenshotMaker

Класс ScreenshotMaker предназначен для создания скриншотов состояния тестируемых веб-страниц в растровом формате .png. В классе предусмотрены методы для создания скриншотов во время проведения тестирования и во время "падения" тесткейса. Создаваемые скриншоты сохраняются в памяти компьютера в каталоге с настоящим программным решением по адресу: ".//target/screenshots/" в каталогах "DuringTest" и "FailedTest" соответственно. Предусмотрено создание подкаталогов, которым присваиваются имена по текущей дате в формате "число-месяц-год". Имена конечных файлов сохраняемых скриншотов имеют формат "час-минута-секунда".

4.2.3 WebDriverSingleton

Класс WebDriverSingleton разработан для настройки единственного экземпляра вебдрайвера используемого в работе веб-браузера и передачи этого экземпляра всем тестовым классам.

В составе класса есть метод который в качестве аргументов принимает значение названия рабочего веб-браузера и значение адреса нахождения утилиты вебдрайвера в постоянной памяти рабочего компьютера. Если значение переменной веб-драйвера равно null, создаётся его экземпляр по входным параметрам, настраиваются таймауты неявных ожиданий, загрузки страницы и скриптов; тут же задаётся размер рабочего окна открываемого веб-браузера.

5 Файлы конфигурации и управления тестами

Программным решением предусмотрен файл управления тестовыми сценариями фреймворка TestNg в формате .xml:

- fullTestSuite.xml задаёт исполнение теста на проверку работы веб-сайта калькулятора сервиса "Google Cloud", имитируя действия пользователя по выбору конфигурации необходимого оборудования, расчёту его стоимости и отправке электронного письма с рассчитанной стоимостью себе на электронную почту с дальнейшим сравнением полученной цены. Содержит два тестовых сценария, описанных в тестовых классах:
    - CalculatePriceTest;
    - SimpleFiledTest.

6 Использование программного решения Cloud_Pricing_Calculator_website_UI_testing

Работа со всеми тестовыми сценариями, разработанными настоящим программным решением не предусматривает наличия у пользователя учётной записи либо регистрировать новый аккаунт в тестируемом сервисе.

Для использования необходимо в константу *DRIVER_PATH* класса InputTestData прописать путь доступа к утилите веб-дайвера в памяти на компьютере пользователя.

При использовании в работе настоящего программного решения веб-браузера, отличного от Chrome, в поле BROWSER_TYPE класса InputTestData указать название рабочего веб-браузера (например – Firefox).