

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»  
(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Курсовая работа  
по дисциплине: «Тестирование программных систем»  
тема: «Автоматизация UI-тестирования веб-приложения «Интернет-магазин  
кондитерских изделий»»

Выполнила: ст. группы ПВ-41  
Немшилова А.Г.  
Проверил: Поляков В.К.

Белгород 2020

## Оглавление

Введение .....	3
Тестирование пользовательского интерфейса .....	3
Инструмент для автоматизации действий Selenium .....	4
Установка инструментов тестирования .....	5
Тестирование веб-приложения .....	7
Тесты для главной страницы (HomePageTests) .....	7
Тесты для страниц продуктов (ProductPageTests) .....	11
Тесты для работы с пользователями (UserTests) .....	17
Тестирование корзины (CartTest) .....	23
Тестирование навбара (NavbarTest) .....	26
Тестирование всего сайта (main) .....	27
Заключение .....	29

## **Введение**

В ходе выполнения данной работы будет реализовано интеграционное тестирование веб-приложения. Тесты должны представлять собой эмуляцию пользовательских историй. В ходе тестирования мы должны осуществить проверку экранов с элементами управления, такими как кнопки меню, иконка и все виды баров – панели инструментов, панели меню, диалоговым окном и т.д. и убедиться в правильной работе веб-приложения либо выявить ошибки. Для выполнения данной работы в качестве инструмента тестирования был выбран selenium, а также язык программирования python и модуль unittest. Объектом тестирования будет веб-приложение «Интернет-магазин кондитерских изделий», написанное ранее для дисциплины веб-программирования.

### **Тестирование пользовательского интерфейса**

Графический интерфейс пользователя (Graphical user interface, GUI) – разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений.

Задачей тестирования графического интерфейса пользователя является обнаружение ошибок следующего характера:

- Ошибки в функциональности посредством интерфейса
- Необработанные исключения при взаимодействии с интерфейсом
- Потеря или искажение данных, передаваемых через элементы интерфейса
- Ошибки в интерфейсе (несоответствие проектной документации, отсутствие элементов интерфейса)

Функциональное тестирование пользовательского интерфейса может проводиться различными методами:

- Ручное тестирование (контроль проводится человеком)
- Автоматическое тестирование (используются программные инструменты, эмулирующие поведение тестирующего), которое будет в дальнейшем использовано в данной работе.

Основной задачей автоматизации функционального тестирования является воспроизведение шагов тест-кейса. Автотест должен повторять те же действия и проверки, что делает тестирующий при ручном тестировании. Эти задачи решаются с помощью специализированных инструментов, позволяющих эмулировать пользовательские действия.

## Инструмент для автоматизации действий Selenium

В данной работе для автоматизации тестирования графического интерфейса был выбран инструмент Selenium. Selenium - это в первую очередь набор библиотек для различных языков программирования. Эти библиотеки используются для отправки HTTP запросов драйверу (WebDriver), с помощью протокола JsonWireProtocol, в которых указано действие, которое должен совершить браузер в рамках текущей сессии. Примерами таких команд могут быть команды нахождения элементов по локатору, переход по ссылкам, парсинг текста страницы/элемента, нажатие кнопок или переход по ссылкам на странице веб-сайта. Существуют как официальные привязки библиотеки к популярным языкам программирования, так и любительские.

Проектом Selenium и сообществом поддерживается работа с браузерами Microsoft Internet Explorer, Google Chrome, Mozilla Suite и Mozilla Firefox под управлением операционных систем Microsoft Windows, Linux и Apple Macintosh. В дальнейшем будет установлен

Основными методами, которые позволяет использовать Selenium для автоматического тестирования и которые будут использованы в дальнейшем в данной работе являются следующие:

`driver.get("http://www.google.com")` – переход на страницу по ссылке

`driver.back()` – возврат к предыдущей странице

`driver.close()` – закрытие браузера

Поиск элементов на странице:

`element = driver.find_element_by_id("passwd-id")`

`element = driver.find_element_by_name("passwd")`

`element = driver.find_element_by_xpath("//input[@id='passwd-id']")`

`element.send_keys("some text")` – ввод данных в текстовое поле элемента

`element.clear()` – отчистить содержимое текстового поля

`driver.switch_to_window("windowName")` – переключение к определенному окну

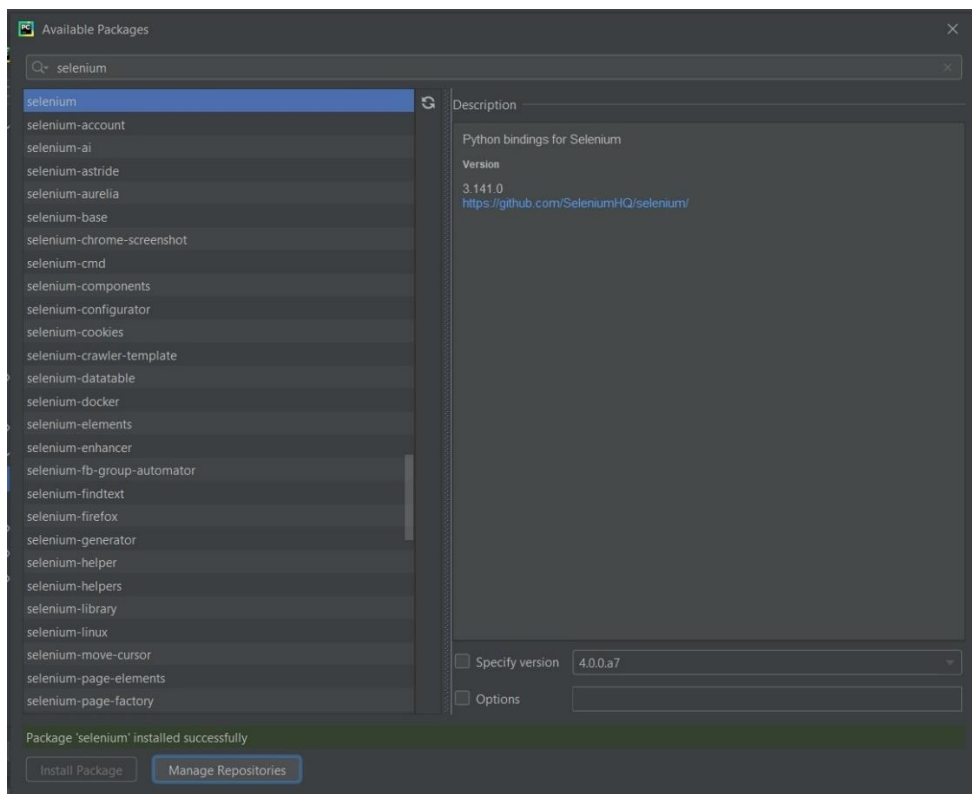
`alert = driver.switch_to_alert()` – переключение на управление всплывающего диалогового окна

`element.text` – возврат текста, содержащегося внутри тега элемента

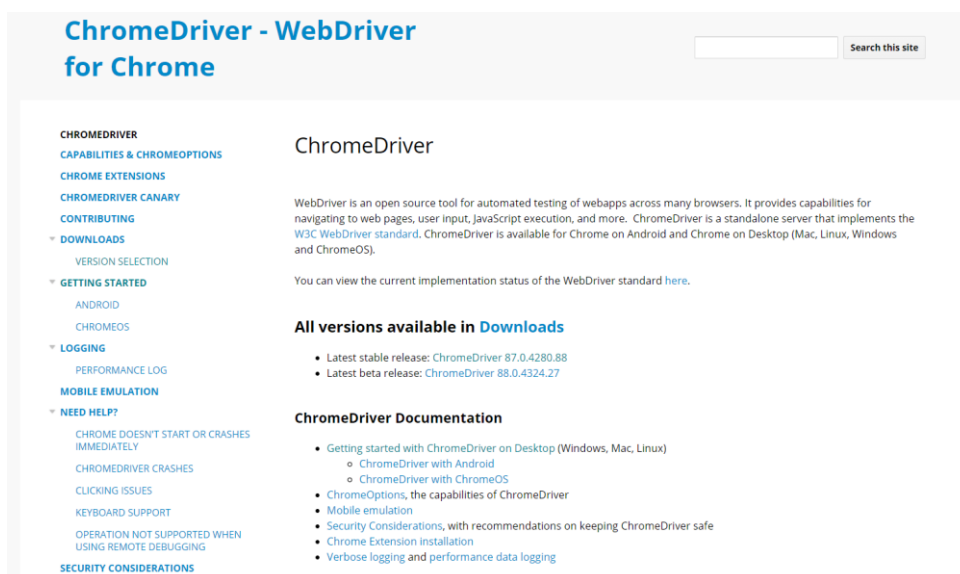
`element.click()` – имитирует нажатие на элемент

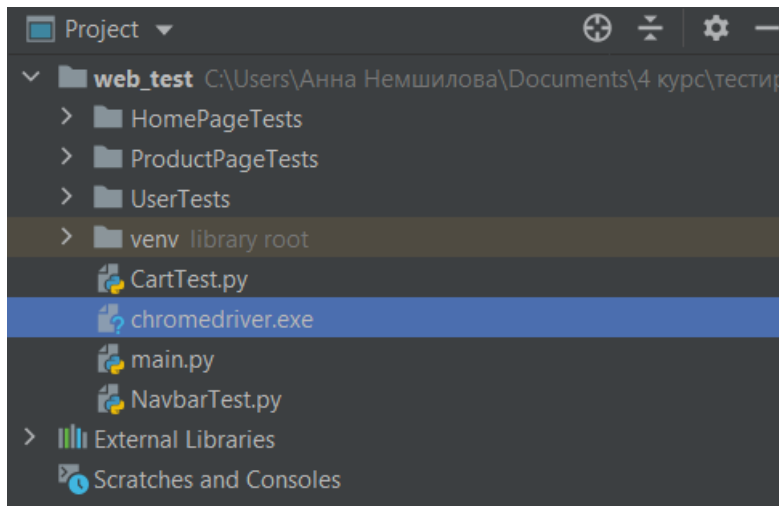
## Установка инструментов тестирования

Так как для реализации тестов был выбран язык программирования python, для использования selenium нужно установить соответствующую библиотеку в настройках проекта:



Также для работы selenium нужно установить соответствующий webdriver, так как для данной работы был выбран браузер Google Chrome, был скачан драйвер для него и помещен в корневую папку проекта:





Также в проекте была импортирована библиотека `unittest` для написания тестов. `Unittest` - стандартный модуль для написания юнит-тестов на Python, который имеет несколько полезных функций:

- можно собирать тесты в группы
- собирать результаты выполнения тестов
- ООП стиль позволяет уменьшить дублирование кода при схожих объектах тестирования

Далее в классах для тестирования будет использована следующая конструкция для применения установленных инструментов (пример одного из далее реализованных классов):

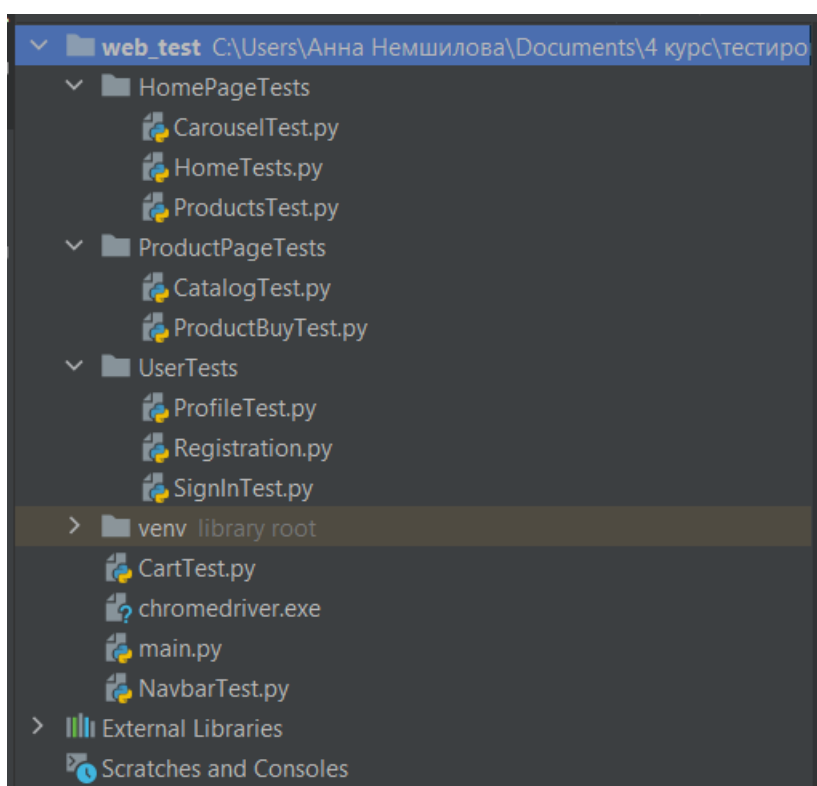
```
class TestRegistration(unittest.TestCase):  
  
    def setUp(self):  
        chromePath = "C:/Users/Анна Немшилова/Documents/4  
курс/тестирование/web_test/chromedriver.exe"  
        self.driver = webdriver.Chrome(executable_path=chromePath)  
        self.driver.get("http://localhost:8080/registration")  
        time.sleep(2)  
  
    def tearDown(self):  
        self.driver.quit()
```

В переменной `chromePath` должен быть указан путь до скачанного раннее драйвера `chromedriver`, который затем передается в метод `webdriver.Chrome` для взаимодействия с браузером. Также в методе `driver.get` указываем ссылку на начальную страницу, которая нужна для тестов данного класса. Методы `setup` и `teardown` реализованы подобным образом в каждом классе тестов.

## Тестирование веб-приложения

Для тестирования веб-приложения интернет-магазина кондитерских изделий были реализованы отдельные тесты для компонентов, связанных с главной страницей сайта, страниц, связанных с продуктами (каталог и отдельные страницы), страниц, связанных с пользователями (профиль, регистрация и авторизация), а также отдельные тесты для корзины и навигационной панели сайта, общей для всех страниц.

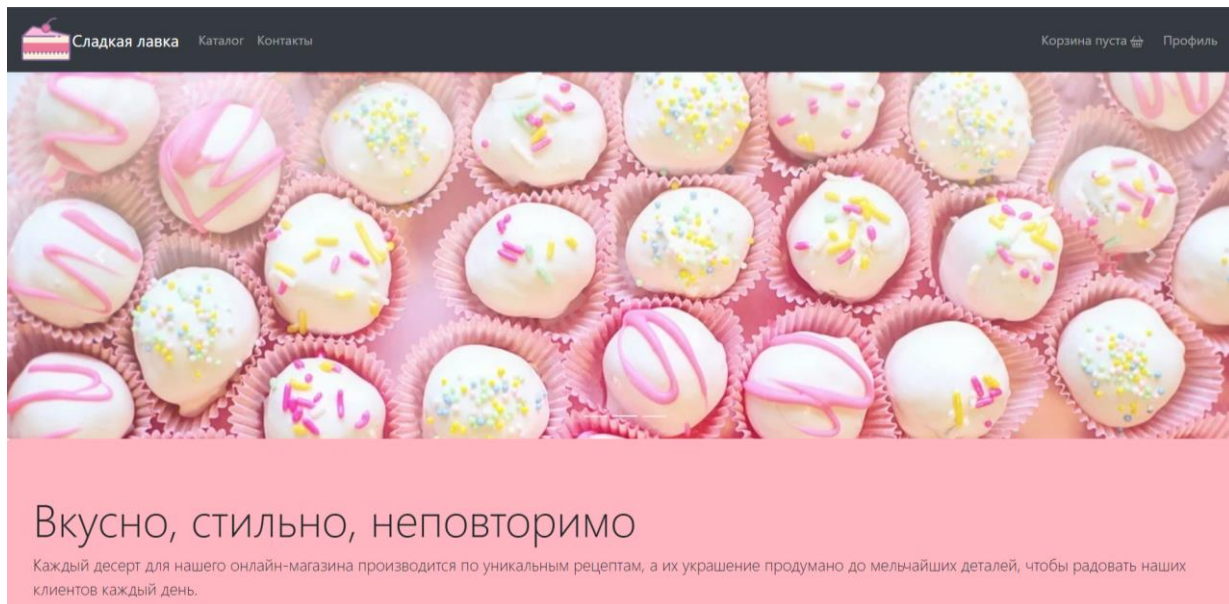
Для запуска всех тестов вместе они также были вынесены в отдельный файл `main.py` проекта.



### Тесты для главной страницы (HomePageTests)

Для главной страницы были написаны тесты для карусели (`CarouselTest`) и продуктов, отображаемых на странице в разделе «Хиты недели» (`ProductsTest`).

## Тестирование карусели главной страницы (CarouselTest)



Класс для тестирования карусели главной страницы включает в себя:

1. Вспомогательная функция для получения статусов изображений, содержащихся в карусели (False – не активно в данный момент на странице, True – активно):

```
#получение активной картинки
def getStatusPictures(self, pictures):
    statusPictures = []
    for picture in pictures:
```



```

        statusPictures.append(
            True if re.search(r'carousel-item active',
picture.get_attribute("outerHTML")) else False)
        return statusPictures

```

## 2. Тестирование переключения изображений карусели (кнопки вперед/назад):

```

#тестирование кнопок карусели переключения слайдов
def testCarouselMoving(self):
    driver = self.driver
    pictures = []
    for i in range(1, 4):
        pictures.append(driver.find_element_by_id("slide" + str(i)))

    #кнопка вперед
    buttonNext = driver.find_element_by_class_name("carousel-control-next")
    buttonNext.click()
    time.sleep(1)
    indNext = self.getStatusPictures(pictures).index(True)
    next = driver.find_element_by_id("slide" + str(indNext + 1))
    self.assertEqual(pictures[indNext], next)

    #кнопка назад
    buttonPrev = driver.find_element_by_class_name("carousel-control-prev")
    buttonPrev.click()
    time.sleep(1)
    indPrev = self.getStatusPictures(pictures).index(True)
    prev = driver.find_element_by_id("slide" + str(indPrev + 1))
    self.assertEqual(pictures[indPrev], prev)

```

Запустив отдельно данные тесты, получили следующий результат:

```
Ran 1 test in 12.229s
```

```
OK
```

## Тестирование продуктов раздела «Хиты недели» (ProductsTest)

## Хиты недели



### Торт 'Секрет небес'

Десерт, выполненный в нежных голубо-розовых тонах со вкусом клубники со сливками, черники и воздушным суфле внутри.

[Перейти к товару](#)



### Капкейки 'Единорог'

Порция ярких воздушных капкейков со вкусом фруктов и нежной сливочной шапочкой.

[Перейти к товару](#)



### Торт 'Космос'

Выполненный в стильной цветовой гамме торт, основные вкусовые составляющие которого - лаванда и ежевика, а также фирменный крем.

[Перейти к товару](#)

## Вкусно, стильно, неповторимо

Каждый десерт для нашего онлайн-магазина производится по уникальным рецептам, а их украшение продумано до мельчайших деталей, чтобы радовать наших клиентов каждый день.

Сделайте свой первый заказ наших фирменных десертов уже сейчас.

[Посмотреть](#)

В классе были проверены переходы на страницы соответствующих продуктов по нажатию на названия либо ссылку в нижней части карточки товара «Перейти к товару».

Также написан тест для проверки правильного функционирования при нажатии кнопки «Посмотреть» (переход на страницу каталога товаров продуктов):

```
#тестирование переходов на страницы продуктов с главной страницы
class TestProducts(unittest.TestCase):
    def setUp(self):
        chromePath = "C:/Users/Анна Немшилова/Documents/4
курс/тестирование/web_test/chromedriver.exe"
        self.driver = webdriver.Chrome(executable_path=chromePath)
        self.driver.get("http://localhost:8080/")
        time.sleep(1)

    def tearDown(self):
        self.driver.quit()

#переход на страницу каталога по кнопке подробнее
def testMoreButton(self):
    driver = self.driver
    moreButton =
driver.find_element_by_xpath("//*[@id=\"app\"]/main/div/div[2]/a/p/a")
moreButton.click()
```

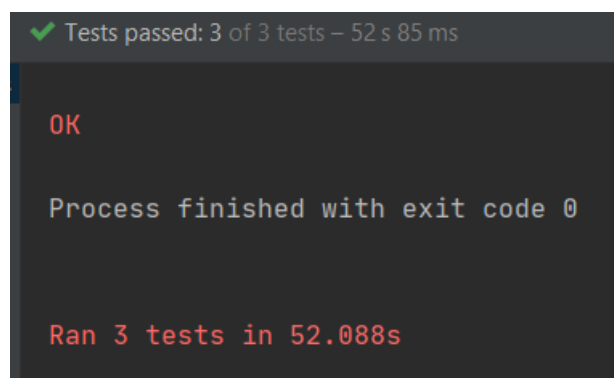
```

time.sleep(1)
title_name = driver.find_element_by_class_name("title-text").text
self.assertEqual(title_name, "Каталог товаров")
driver.back()
time.sleep(1)

#переход к товарам из раздела "хиты" по нажатию на "перейти к товару"
def testHitProductsByGo(self):
    driver = self.driver
    productsNames = ["Торт 'Секрет небес'", "Капкейки 'Единорог'", "Торт 'Космос'"]
    for i in range(1, 4):
        hitProduct = driver.find_element_by_xpath(
            "//*[@id=\"app\"] /main/div/div[3]/div/div[{}]/div/div[2]/small/a/a".format(i)
        )
        hitProduct.click()
        time.sleep(2)
        productName = driver.find_element_by_class_name("product-
name").text
        self.assertEqual(productName, productsNames[i-1])
        driver.back()
        time.sleep(2)

# переход к товарам из раздела "хиты" по нажатию на название товара
def testHitProductsByName(self):
    driver = self.driver
    productsNames = ["Торт 'Секрет небес'", "Капкейки 'Единорог'", "Торт 'Космос'"]
    for i in range(1, 4):
        hitProduct = driver.find_element_by_xpath(
            "//*[@id=\"app\"] /main/div/div[3]/div/div[{}]/div/div[1]/h5/a".format(i)
        )
        hitProduct.click()
        time.sleep(2)
        productName = driver.find_element_by_class_name("product-
name").text
        self.assertEqual(productName, productsNames[i - 1])
        driver.back()
        time.sleep(2)

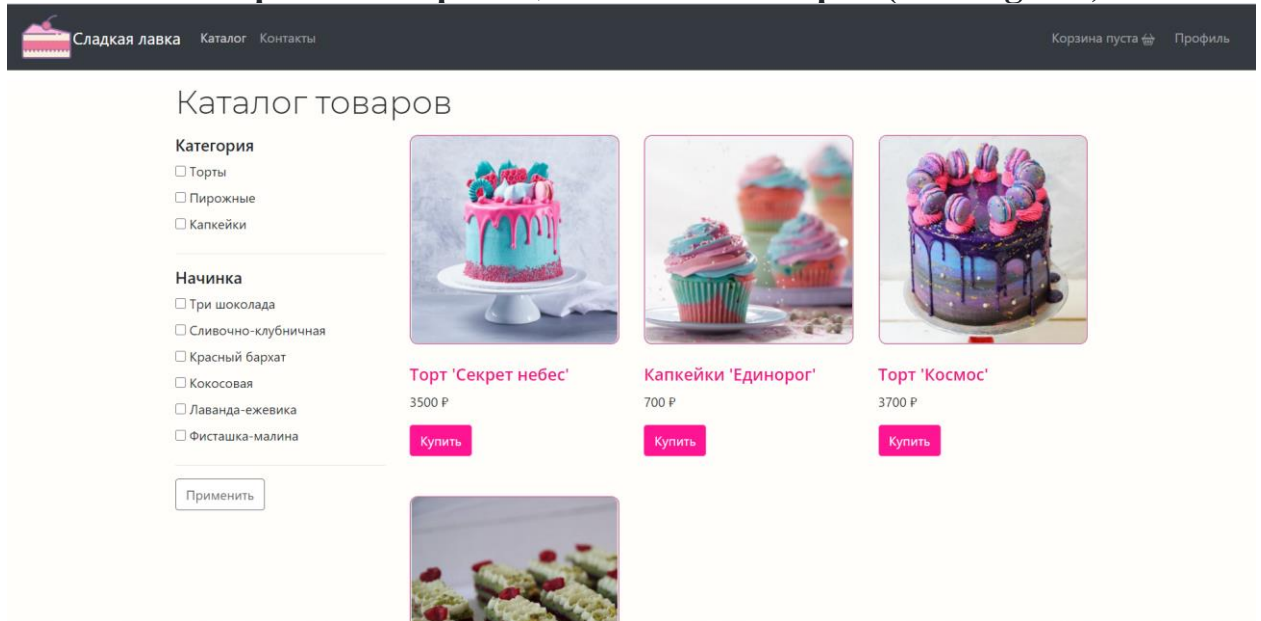
```



## Тесты для страниц продуктов (ProductPageTests)

Тесты страниц продуктов включают себя тесты для страницы каталога товаров(CatalogTest) и страницы одного товара(ProductBuyTest).

## Тестирование страницы каталога товаров (CatalogTest)



Для страницы каталога были реализованы следующие тесты:

- переход на страницу отдельного товара по нажатию на его название
- добавление в корзину товара по нажатию на кнопку «Купить» (для получения содержимого корзины использовался метод `self.driver.execute_script("return localStorage.getItem('cart')")`, так как содержимое корзины хранится в переменной `cart` локальной памяти)
- проверка правильной работы чекбоксов для фильтрации товаров каталога (категории товара, начинки товара, а также совместное использование двух параметров для фильтрации):

## Каталог товаров

### Категория

- ☒ Торты
- ☐ Пирожные
- ☐ Капкейки

### Начинка

- ☐ Три шоколада
- ☐ Сливочно-клубничная
- ☐ Красный бархат
- ☐ Кокосовая
- ☐ Лаванда-ежевика
- ☐ Фисташка-малина

Применить



Торт 'Секрет небес'

3500 Р

Купить



Торт 'Космос'

3700 Р

Купить

## Каталог товаров

### Категория

- ☐ Торты
- ☐ Пирожные
- ☐ Капкейки

### Начинка

- ☐ Три шоколада
- ☐ Сливочно-клубничная
- ☐ Красный бархат
- ☒ Кокосовая
- ☐ Лаванда-ежевика
- ☐ Фисташка-малина

Применить



Капкейки 'Единорог'

700 Р

Купить

## Каталог товаров

### Категория

- ☐ Торты
- ☒ Пирожные
- ☐ Капкейки

### Начинка

- ☐ Три шоколада
- ☐ Сливочно-клубничная
- ☐ Красный бархат
- ☐ Кокосовая
- ☐ Лаванда-ежевика
- ☒ Фисташка-малина

Применить



Пирожные 'Фисташка-малина'

500 Р

Купить

```
#тестирование страницы каталога товаров
class TestCatalog(unittest.TestCase):
    def setUp(self):
        chromePath = "C:/Users/Анна Немшилова/Documents/4
курс/тестирование/web_test/chromedriver.exe"
        self.driver = webdriver.Chrome(executable_path=chromePath)
        self.driver.get("http://localhost:8080/catalog")
        time.sleep(1)

    def tearDown(self):
        self.driver.quit()

    #переход на страницу товара
    def testProducts(self):
        driver = self.driver
        productsNames = ["Торт 'Секрет небес'", "Капкейки 'Единорог'", "Торт
'Космос'", "Пирожные 'Фисташка-малина'"]
```

```

        for i in range(1, 5):
            product = driver.find_element_by_xpath(
                "//*[@id=\"app\"] / main / div / div / div / div[2] / div / div[{}] / div / h3 / a".format(i))
            product.click()
            time.sleep(2)
            productName = driver.find_element_by_class_name("product-
name").text
            self.assertEqual(productName, productsNames[i - 1])
            driver.back()
            time.sleep(2)

#добавление в корзину товара по кнопке "Купить"
def testButtonBuy(self):
    driver = self.driver
    for i in range(1, 5):
        productName = driver.find_element_by_class_name("product-
name").text
        productBuy = driver.find_element_by_xpath(
            "//*[@id = \"app\"] / main / div / div / div / div[2] /
div / div[{}] / div / p[2] / button".format(i))
        productBuy.click()
        time.sleep(1)
        cart = self.driver.execute_script("return
localStorage.getItem('cart')")
        self.assertTrue(productName in cart)

#проверка фильтра по категории при выборе чекбокса на примере категории
"Торты"
def testCategory(self):
    driver = self.driver
    productsNames = []
    checkbox = driver.find_element_by_id('Торты')
    checkbox.click()
    time.sleep(2)
    buttonApply =
driver.find_element_by_xpath("//*[@id=\"app\"] / main / div / div / div / div[1] / button
")
    buttonApply.click()
    time.sleep(2)
    for element in driver.find_elements_by_class_name("catalog-title"):
        productsNames.append(element.text)
    self.assertTrue("Торт 'Секрет небес'" in productsNames)
    self.assertTrue("Торт 'Космос'" in productsNames)
    self.assertFalse("Капкейки 'Единорог'" in productsNames)
    self.assertFalse("Пирожные 'Фисташка-малина'" in productsNames)

#проверка фильтра по начинке при выборе чекбокса на примере начинки
"Кокосовая"
def testFilling(self):
    driver = self.driver
    productsNames = []
    checkbox = driver.find_element_by_id('Кокосовая')
    checkbox.click()
    time.sleep(2)
    buttonApply =
driver.find_element_by_xpath("//*[@id=\"app\"] / main / div / div / div / div[1] / button
")
    buttonApply.click()
    time.sleep(2)
    for element in driver.find_elements_by_class_name("catalog-title"):
        productsNames.append(element.text)
    self.assertTrue("Капкейки 'Единорог'" in productsNames and

```

```

len(productsNames) == 1)

#проверка совместного использования фильтров при выборе двух чекбоксов
def testCategoryFilling(self):
    driver = self.driver
    productsNames = []
    checkbox = driver.find_element_by_id('Пирожные')
    checkbox.click()
    time.sleep(1)
    checkbox = driver.find_element_by_id('Фисташка-малина')
    checkbox.click()
    time.sleep(1)
    buttonApply =
driver.find_element_by_xpath("//*[@id=\"app\"] /main/div/div/div/div/div[1]/button
")
    buttonApply.click()
    time.sleep(2)
    for element in driver.find_elements_by_class_name("catalog-title"):
        productsNames.append(element.text)
    self.assertTrue("Пирожные 'Фисташка-малина'" in productsNames and
len(productsNames) == 1)

```

```

> ✓ Tests passed: 5 of 5 tests – 1 m 39 s 766 ms

Process finished with exit code 0

Ran 5 tests in 99.774s


OK

```

## Тестирование отдельной страницы товара (ProductBuyTest)

Страница каждого отдельного товара из активных элементов содержит только кнопку «Добавить в корзину», поэтому был написан тест для проверки добавления данного товара по нажатию на нее в корзину:





### Торт 'Секрет небес'

Десерт, выполненный в нежных голубо-розовых тонах со вкусом клубники со сливками, черники и воздушным суфле внутри.

Состав: суфле, кокос, сливки, клубничный джем, черничное суфле

Вес: 3000г

Срок хранения: 5 суток

---

Цена: 3500Р

[Добавить в корзину](#)

Средний срок приготовления - 1 день

Подробнее

Десерт, выполненный в нежных голубо-розовых тонах со вкусом клубники со сливками, черники и воздушным суфле внутри.

Пищевая и энергетическая ценность на 100 г продукта (средние значения):  
Жиры- 12,9 г, Белки- 3,5 г, Углеводы- 29,5 г,  
Энергетическая ценность (калорийность)- 1079 кДж (258 Ккал).

Условия хранения: от +2 до +6 °С

```
#тестирование добавления продукта в корзину со страницы продукта по нажатию
кнопки "Купить"
class TestProductBuy(unittest.TestCase):

    def setUp(self):
        chromePath = "C:/Users/Анна Немшилова/Documents/4
курс/тестирование/web_test/chromedriver.exe"
        self.driver = webdriver.Chrome(executable_path=chromePath)
        self.driver.get("http://localhost:8080/catalog/СекретНебес")
        time.sleep(1)

    def tearDown(self):
        self.driver.quit()

    def buyProduct(self, urlProduct):
        driver = self.driver
        driver.get("http://localhost:8080/catalog/" + urlProduct)

        productName = driver.find_element_by_xpath("//*[@id=\"v-pills-
tabContent\"]/div[1]/div/div[2]/div/h4").text
        buyButton = driver.find_element_by_xpath("//*[@id=\"v-pills-
tabContent\"]/div[1]/div/div[2]/div/button")
        buyButton.click()
        time.sleep(2)
        cart = self.driver.execute_script("return
localStorage.getItem('cart')")
        self.assertTrue(productName in cart)

    def testBuyProducts(self):
        self.buyProduct("СекретНебес")
        self.buyProduct("КапкейкиЕдинорог")
        self.buyProduct("ТортКосмос")
        self.buyProduct("ПирожныеФисташкаМалина")
```

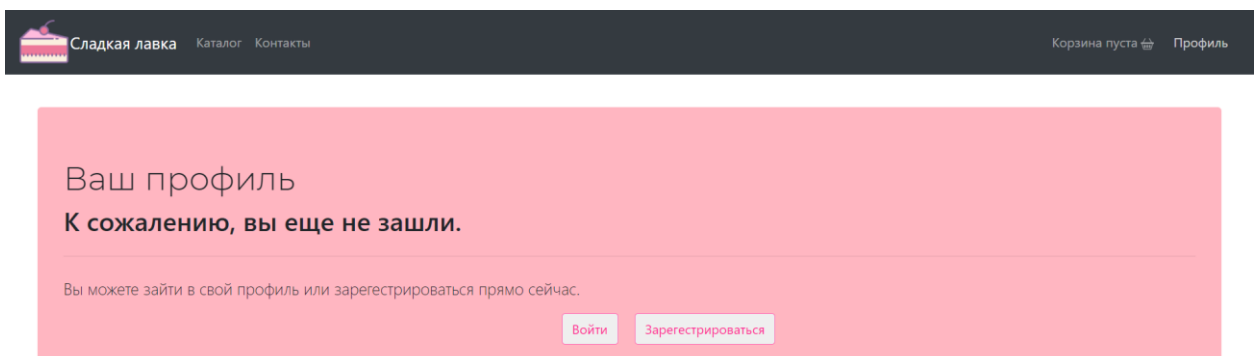


```
✓ Tests passed: 1 of 1 test – 16 s 965 ms
OK
Process finished with exit code 0
Ran 1 test in 16.966s
```

## Тесты для работы с пользователями (UserTests)

Тесты для работы с пользователями включают в себя тесты для страницы профиля пользователя (ProfileTest), страницы регистрации (Registration) и авторизации пользователя (SignInTest).

### Тестирование страницы профиля пользователя (ProfileTest)



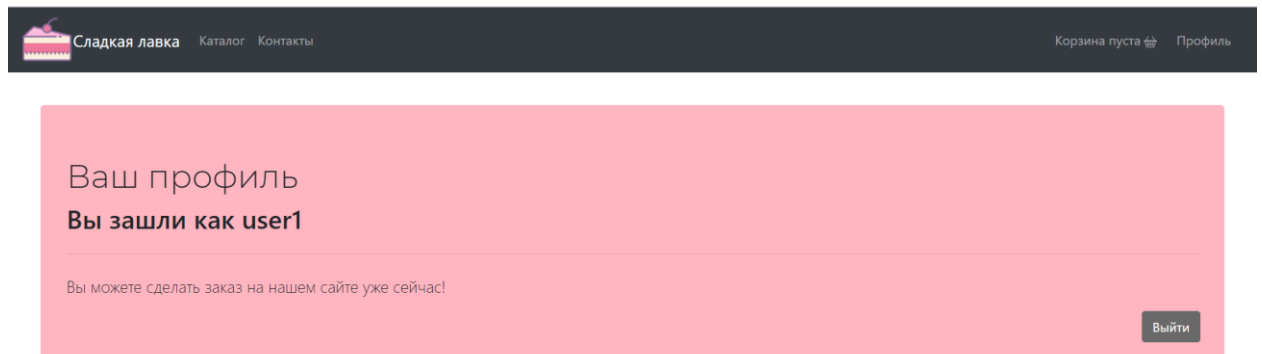
Когда пользователь еще не зашел в профиль, на странице проверяется переход на соответствующие страницы по кнопкам «Войти» и «Зарегистрироваться»:

```
#проверка перехода на страницы входа и регистрации по кнопкам, если
пользователь еще не зашел в свой профиль
def testProfilePage(self):
    driver = self.driver

    buttonSignIn =
driver.find_element_by_xpath("//*[@id=\"app\"] /main/div/p[2]/button[1]")
    buttonSignIn.click()
    time.sleep(1)
    self.assertEqual(driver.current_url, "http://localhost:8080/sign-in")
    time.sleep(2)
    driver.back()
    time.sleep(2)

    buttonRegistration =
driver.find_element_by_xpath("//*[@id=\"app\"] /main/div/p[2]/button[2]")
    buttonRegistration.click()
    time.sleep(1)
    self.assertEqual(driver.current_url,
```

```
"http://localhost:8080/registration")
time.sleep(2)
driver.back()
```



Когда пользователь зашел в свой профиль, проверяется выход из профиля по кнопке «Выйти» (после этого должен пройти переход на страницу профиля, показанную в первом случае):

```
#выход из профиля пользователя по кнопке "Выход"
def testExit(self):
    driver = self.driver
    buttonSignIn =
driver.find_element_by_xpath("//*[@id=\"app\"]/main/div/p[2]/button[1]")
    buttonSignIn.click()
    time.sleep(1)

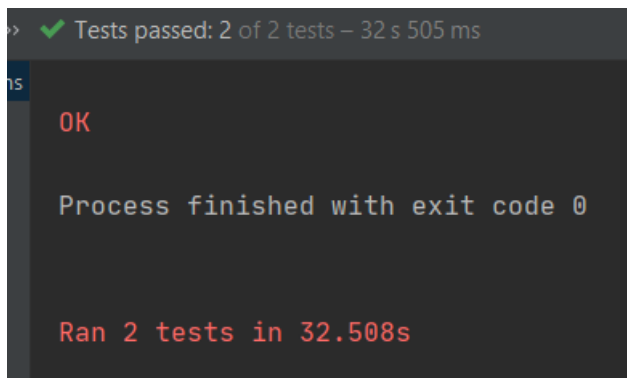
    #вход пользователя

driver.find_element_by_xpath("//*[@id=\"inputUsername\"]").send_keys("user1")

driver.find_element_by_xpath("//*[@id=\"inputPassword\"]").send_keys("pass1")
    time.sleep(1)
    signInButton =
driver.find_element_by_xpath("//*[@id=\"app\"]/main/div/div/div/div[2]/form/b
utton")
    signInButton.click()
    time.sleep(2)

    #выход пользователя
    buttonExit =
driver.find_element_by_xpath("//*[@id=\"app\"]/main/div/p[2]/button")
    buttonExit.click()
    time.sleep(2)

self.assertEqual(driver.find_element_by_xpath("//*[@id=\"app\"]/main/div/h2")
.text, "К сожалению, вы еще не зашли.")
```



## Тестирование страницы регистрации пользователя (Registration)

В тестах для данной страницы сначала была реализована вспомогательная функция регистрации пользователя, которая с помощью метода `send_keys` заполняет текстовые поля формы регистрации соответствующими значениями, переданными как аргументы:

```
#регистрация пользователя
def registration(self, username, password, password2):
    driver = self.driver

    driver.find_element_by_xpath("//*[@id=\"inputUsername\"]").send_keys(username)

    driver.find_element_by_xpath("//*[@id=\"inputPassword\"]").send_keys(password)

    driver.find_element_by_xpath("//*[@id=\"repeatPassword\"]").send_keys(password2)
    time.sleep(1)
    regButton =
driver.find_element_by_xpath("//*[@id=\"app\"]/main/div/div[2]/form/button")
    regButton.click()
    time.sleep(2)
```

Затем проверяются два случая регистрации пользователя. В случае успешной регистрации (пароль и повторный пароль совпадают) должна появиться страница профиля данного пользователя, уже осуществившего вход в систему:

Регистрация

Добро пожаловать

Пожалуйста, заполните поля ниже.

Логин

newUser

Пароль

.....

Повторите пароль

.....

Зарегистрироваться

Ваш профиль

Вы зашли как newUser

Вы можете сделать заказ на нашем сайте уже сейчас!

Выйти

```
#тест успешной регистрации пользователя
def testRegistration(self):
    driver = self.driver
    self.registration("newUser", "12345", "12345")

self.assertEqual(driver.find_element_by_xpath("//*[@id=\"app\"]/main/div/h2")
    .text, "Вы зашли как userNew1")
```

В случае неправильной регистрации (пароль и повторный пароль при регистрации не совпадают) должно появиться всплывающее диалоговое окно с сообщением об ошибке:

Регистрация

Добро пожаловать

Пожалуйста, заполните поля ниже.

Логин

newUser

Пароль

.....

Повторите пароль

.....

Зарегистрироваться

Подтвердите действие на странице localhost:8080

Пароли не совпадают

OK

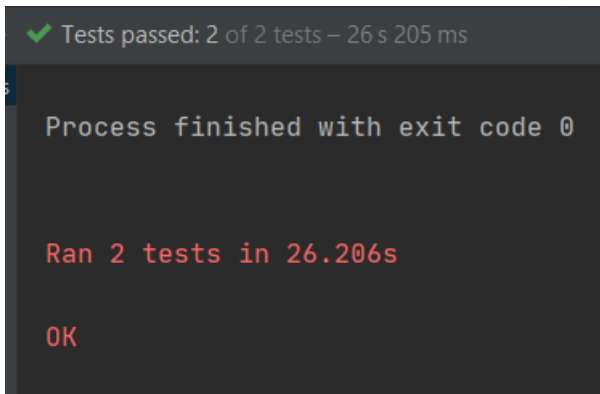
```
#тест неправильной регистрации пользователя
def testRegistrationError(self):
```

```

driver = self.driver
self.registration("newUser", "12345", "12344")

#проверка появления диалогового окна с предупреждением
wait = WebDriverWait(driver, 10)
wait.until(EC.alert_is_present())
alert = driver.switch_to.alert
self.assertEqual(alert.text, "Пароли не совпадают")

```



## Тестирование страницы входа пользователя (SignInTest)

В тестах для данной страницы сначала была реализована вспомогательная функция входа пользователя, которая с помощью метода `send_keys` заполняет текстовые поля формы регистрации соответствующими значениями, переданными как аргументы:

```

#вход пользователя на сайт
def signIn(self, username, password):
    driver = self.driver

    driver.find_element_by_xpath("//*[@id=\"inputUsername\"]").send_keys(username)

    driver.find_element_by_xpath("//*[@id=\"inputPassword\"]").send_keys(password)

    time.sleep(1)
    signInButton =
driver.find_element_by_xpath("//*[@id=\"app\"] /main/div/div/div/div[2] /form/b
utton")
    signInButton.click()
    time.sleep(2)

```

Затем проверяются два случая входа пользователя. В случае успешного входа пользователя на сайт должна появиться страница профиля данного пользователя, осуществившего вход в систему:

Авторизация

Добро пожаловать  
Введите ваш логин и пароль.

Логин  
user1

Пароль  
.....

Войти

Зарегистрироваться

Ваш профиль

Вы зашли как user1

Вы можете сделать заказ на нашем сайте уже сейчас!

Выйти

```
#тестирование успешного входа на сайт
def testSignIn(self):
    driver = self.driver
    self.signIn("user1", "pass1")

self.assertEqual(driver.find_element_by_xpath("//*[@id=\"app\"] /main/div/h2")
.text, "Вы зашли как user1")
```

В случае ввода неправильного логина или пароля должно появиться всплывающее диалоговое окно с сообщением об ошибке:

Авторизация

Добро пожаловать  
Введите ваш логин и пароль.

Логин  
user1

Пароль  
.....

Войти

Зарегистрироваться

Подтвердите действие на странице localhost:8080  
Неправильно введен логин или пароль  
ОК

```
#тестирование неправильного входа на сайт
def testSignInError(self):
    driver = self.driver
    self.signIn("user576", "pass576")

    wait = WebDriverWait(driver, 10)
    wait.until(EC.alert_is_present())
    alert = driver.switch_to.alert
    self.assertEqual(alert.text, "Неправильно введен логин или пароль")
```

Также тестируется переход на страницу регистрации по нажатию на соответствующую ссылку внизу формы входа:

```
#переход на страницу регистрации по ссылке "Зарегистрироваться"
def testGoReg(self):
    driver = self.driver
    buttonReg = driver.find_element_by_class_name("reg-link")
    buttonReg.click()
    time.sleep(1)
    self.assertEqual(driver.current_url,
"http://localhost:8080/registration")
```

```
> ✓ Tests passed: 3 of 3 tests – 35 s 103 ms
s Testing started at 20:54 ...
  "C:\Users\Анна Немшилова\Documents\
  Launching unittests with arguments

  Ran 3 tests in 35.108s

  OK
```

## Тестирование корзины (CartTest)



### Корзина

К сожалению, в вашей корзине пока что нет товаров.

# Корзина



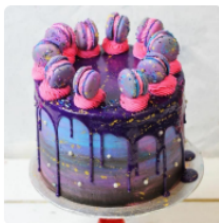
Торт 'Секрет небес'  
3500 Р

Удалить



Капкейки 'Единорог'  
700 Р

Удалить



Торт 'Космос'  
3700 Р

Удалить

Итого: 7900 Р

Тестирование корзины включает в себя следующие тесты:

- отображение соответствующего текста для пустой корзины
- правильное отображение товаров при добавлении их в корзину
- удаление товара из корзины

```
#тестирование корзины
class TestCart(unittest.TestCase):
    def setUp(self):
        chromePath = "C:/Users/Анна Немшилова/Documents/4
курс/тестирование/web_test/chromedriver.exe"
        self.driver = webdriver.Chrome(executable_path=chromePath)
        self.driver.get("http://localhost:8080/cart")
        time.sleep(1)

    def tearDown(self):
        self.driver.quit()

    #если корзина пустая, то выводится соответствующее сообщение
    def testEmptyCart(self):
        driver = self.driver
        if self.driver.execute_script("return localStorage.getItem('cart')")
== None:
            self.assertIsNotNone(driver.find_elements_by_class_name("cart-
empty"))
```



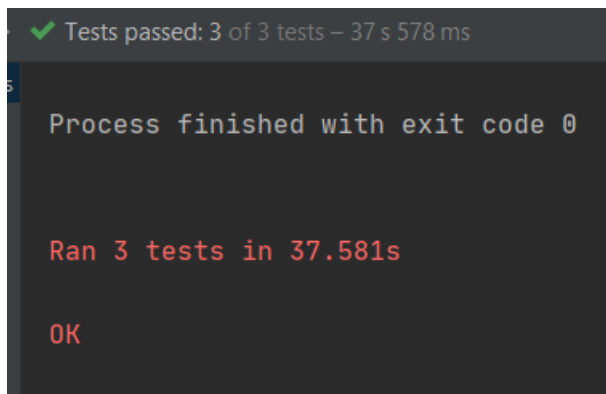
```

#тестирование отображения добавленных товаров в корзине
def testProductsCart(self):
    #переход на страницу каталога и добавление товаров в корзину по
    #кнопке "Купить" и возврат на страницу корзины
    driver = self.driver
    driver.get("http://localhost:8080/catalog")
    time.sleep(1)
    for i in range(1, 5):
        productBuy = driver.find_element_by_xpath(
            "// *[@ id = \"app\"] / main / div / div / div / div[2] /
div / div[{}] / div / p[2] / button".format(i))
        productBuy.click()
        time.sleep(1)
        pageCartButton =
driver.find_element_by_xpath("//*[@id=\"navbarSupportedContent\"]/div/ul/li[1
]")
        pageCartButton.click()
        time.sleep(2)

        #проверка отображения добавленных товаров на странице корзины
        cart = self.driver.execute_script("return
localStorage.getItem('cart')")
        if cart != None:
            for i in range(1, 5):
                productCart = driver.find_element_by_xpath(
                    "//*[@id=\"app\"]/main/div/div/div[{}]/div/div/h5/a".format(i)).text
                self.assertTrue(productCart in cart)

#удаление продукта из корзины (на примере торта "Космос")
def testDelProduct(self):
    driver = self.driver
    driver.get("http://localhost:8080/catalog")
    time.sleep(1)
    productBuy = driver.find_element_by_xpath(
        "// *[@ id = \"app\"] / main / div / div / div / div[2] /
div / div[3] / div / p[2] / button")
    productBuy.click()
    time.sleep(1)
    pageCartButton =
driver.find_element_by_xpath("//*[@id=\"navbarSupportedContent\"]/div/ul/li[1
]")
    pageCartButton.click()
    time.sleep(2)
    delButton =
driver.find_element_by_xpath("//*[@id=\"app\"]/main/div/div/div/div/button")
    delButton.click()
    time.sleep(2)
    cart = self.driver.execute_script("return
localStorage.getItem('cart')")
    self.assertFalse("Торт 'Космос'" in cart)

```



## Тестирование навбара (NavbarTest)

Также была протестирована навигационная панель, общая для всех страниц. Тесты включают себя переходы на правильные страницы по нажатию на соответствующие названия и правильное отображение статуса корзины (пустая или нет).



```
#тестирование навбара для всех страниц
class TestNavbar(unittest.TestCase):
    def setUp(self):
        chromePath = "C:/Users/Анна Немшилова/Documents/4
курс/тестирование/web_test/chromedriver.exe"
        self.driver = webdriver.Chrome(executable_path=chromePath)
        self.driver.get("http://localhost:8080/")
        time.sleep(1)

    def tearDown(self):
        self.driver.quit()

    #переход по страницам меню
    def testNavbarPages(self):
        driver = self.driver
        pagesNameArray = ["Каталог товаров", "Контакты"]
        for i in range(1, 3):
            pageButton =
driver.find_element_by_xpath("//*[@id=\"navbarSupportedContent\"]/ul/li[{}]".
format(i))
            pageButton.click()
            time.sleep(1)
            title_name = driver.find_element_by_class_name("title-text").text
            self.assertEqual(title_name, pagesNameArray[i-1])

        pagesNameArray = ["Корзина", "Ваш профиль"]
        for i in range(1, 3):
            pageButton =
driver.find_element_by_xpath("//*[@id=\"navbarSupportedContent\"]/div/ul/li[{}]".
format(i))
            pageButton.click()
            time.sleep(1)
            title_name = driver.find_element_by_class_name("title-text").text
            self.assertEqual(title_name, pagesNameArray[i-1])
```

```

#проверка отображения состояния корзины
def testCart(self):
    cartButtonName =
self.driver.find_element_by_xpath("//*[@id=\"navbarSupportedContent\"]/div/ul
/li[1]").text
    cart = self.driver.execute_script("return
localStorage.getItem('cart')")
    if cart != None:
        self.assertNotEqual(cartButtonName, "Корзина пуста")
    else:
        self.assertEqual(cartButtonName, "Корзина пуста")

```

```

✓ Tests passed: 2 of 2 tests – 30 s 548 ms
Testing started at 21:07 ...
"C:\Users\Анна Немшилова\Documents\
Launching unittests with arguments
Ran 2 tests in 30.550s
OK

```

## Тестирование всего сайта (main)

Для запуска всех тестов сразу они были объединены в файле main.py.

```

import unittest
from HomePageTests.CarouselTest import TestCarousel
from HomePageTests.ProductsTest import TestProducts
from ProductPageTests.CatalogTest import TestCatalog
from ProductPageTests.ProductBuyTest import TestProductBuy
from UserTests.ProfileTest import TestProfile
from UserTests.Registration import TestRegistration
from UserTests.SignInTest import TestSignIn
from CartTest import TestCart
from NavbarTest import TestNavbar

def runTests(testClasses):
    loader = unittest.TestLoader()
    suitesList = []
    for testClass in testClasses:
        suite = loader.loadTestsFromTestCase(testClass)
        suitesList.append(suite)

    runner = unittest.TextTestRunner(verbosity=2)
    runner.run(unittest.TestSuite(suitesList))

def main():
    runTests([TestCarousel, TestProducts])
    runTests([TestNavbar])
    runTests([TestCatalog, TestProductBuy])
    runTests([TestCart])
    runTests([TestProfile, TestRegistration, TestSignIn])

```

```
if __name__ == '__main__':  
    main()
```

Результат работы:

```
testCarouselMoving (HomePageTests.CarouselTest.TestCarousel) ... ok  
testHitProductsByGo (HomePageTests.ProductsTest.TestProducts) ... ok  
testHitProductsByName (HomePageTests.ProductsTest.TestProducts) ... ok  
testMoreButton (HomePageTests.ProductsTest.TestProducts) ... ok  
  
-----  
Ran 4 tests in 64.503s  
  
OK  
testCart (NavbarTest.TestNavbar) ... ok  
testNavbarPages (NavbarTest.TestNavbar) ... ok  
  
-----  
Ran 2 tests in 24.030s  
  
OK  
testButtonBuy (ProductPageTests.CatalogTest.TestCatalog) ... ok  
testCategory (ProductPageTests.CatalogTest.TestCatalog) ... ok  
testCategoryFilling (ProductPageTests.CatalogTest.TestCatalog) ... ok  
testFilling (ProductPageTests.CatalogTest.TestCatalog) ... ok  
testProducts (ProductPageTests.CatalogTest.TestCatalog) ... ok  
testBuyProducts (ProductPageTests.ProductBuyTest.TestProductBuy) ... ok  
  
-----  
Ran 6 tests in 89.777s  
  
OK
```

```
OK
testDelProduct (CartTest.TestCart) ... ok
testEmptyCart (CartTest.TestCart) ... ok
testProductsCart (CartTest.TestCart) ... ok

-----

Ran 3 tests in 37.558s

OK
testExit (UserTests.ProfileTest.TestProfile) ... ok
testProfilePage (UserTests.ProfileTest.TestProfile) ... ok
testRegistration (UserTests.Registration.TestRegistration) ... ok
testRegistrationError (UserTests.Registration.TestRegistration) ... ok
testGoReg (UserTests.SignInTest.TestSignIn) ... ok
testSignIn (UserTests.SignInTest.TestSignIn) ... ok
testSignInError (UserTests.SignInTest.TestSignIn) ... ok

-----

Ran 7 tests in 89.766s

OK
```

## Заключение

В ходе выполнения курсовой работы реализовали автоматизированное тестирование веб-приложений. Научились работать с таким инструментом тестирования, как selenium. Написали тесты для отдельных веб-страниц, их различных компонентов, таких как карусель, корзина, протестировали переходы на различные страницы через ссылки. Протестировали такие элементы, как кнопки, чекбоксы, поля ввода и т.д. Протестировали написанными тестами используемый проект и получили результаты.