

Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. Н.П. ОГАРЁВА»

(ФГБОУ ВО «МГУ им. Н.П. Огарёва»)

Факультет математики и информационных технологий

Кафедра фундаментальной информатики

ОТЧЁТ ПО ТЕСТИРОВАНИЮ
по дисциплине: Методы тестирования программных продуктов

Автор отчёта по лабораторной работе _____ Е. А. Буйнов
подпись, дата

Обозначение работы

Направление подготовки 02.04.02 Фундаментальная информатика и
информационные технологии

Руководитель работы,
канд. тех. наук

_____ А.В. Попов
подпись дата

Саранск 2025

Лабораторная работа №3.

Автоматизированное функциональное тестирование веб-сайта с помощью фреймворка Selenium.

1. Описание предмета тестирования

Avito — российская онлайн-платформа для размещения объявлений о товарах, услугах, вакансиях и недвижимости.

2. Описание окружения тестирования

Тип устройства: Ноутбук

Процессор: AMD Ryzen 7 6800H 8-Core 3.2GHz

Видеокарта: AMD Radeon Graphics

Оперативная память: 16 Gb, DDR5, 6400MHz

Количество ядер: физических 8, логических 16

Операционная система: Windows 11

Разрешение экрана: 2560 × 1440

Антивирус: нет

Расширения: нет

Браузер: Google Chrome

Характеристики программного обеспечения:

Operating System: Windows 11 (64-bit)

Browser: Google Chrome

Automation Tool: Selenium WebDriver (Python bindings)

Programming Language: Python (Version 3.10)

Libraries: Selenium

IDE: VS Code (for writing and executing scripts)

3. Use cases (пользовательские сценарии)

Ниже приведены тест кейсы для Wildberries, как положительные, так и отрицательные сценарии.

Тест кейс 1 (Позитивный): Регистрация нового пользователя.

Кейс: Пользователь регистрируется через email.

Шаги:

1. Переход на главную страницу.
2. Нажатие кнопки «Войти».
3. Выбор регистрации через email.
4. Ввод валидных данных.
5. Подтверждение регистрации.

Ожидаемый результат: Успешное создание аккаунта.

Код тест кейса:

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

def test_registration():
    driver = webdriver.Chrome()
    driver.get("https://www.avito.ru")
    try:
        # Шаг 1: Открыть форму регистрации
        login_btn = WebDriverWait(driver, 10).until(
            EC.element_to_be_clickable((By.XPATH, "//button[contains(text(),
'Войти')]"))
        login_btn.click()
        time.sleep(2)

        # Шаг 2: Выбрать регистрацию через email
        reg_link = driver.find_element(By.XPATH, "//a[contains(text(),
'Зарегистрироваться')]")
        reg_link.click()
        time.sleep(3)
```

Шаг 3: Заполнить данные

```
email_field = driver.find_element(By.ID, "email")
email_field.send_keys("valid_email@example.com")
password_field = driver.find_element(By.ID, "password")
password_field.send_keys("SecurePass123!")
confirm_field = driver.find_element(By.ID, "confirmPassword")
confirm_field.send_keys("SecurePass123!")
time.sleep(2)
```

Шаг 4: Отправить форму

```
submit_btn = driver.find_element(By.XPATH, "//button[@type='submit']")
submit_btn.click()
time.sleep(5)
```

Проверка успешной регистрации

```
assert "Добро пожаловать" in driver.page_source
```

finally:

```
driver.quit()
```

Сохраненные при выполнении теста скриншоты:

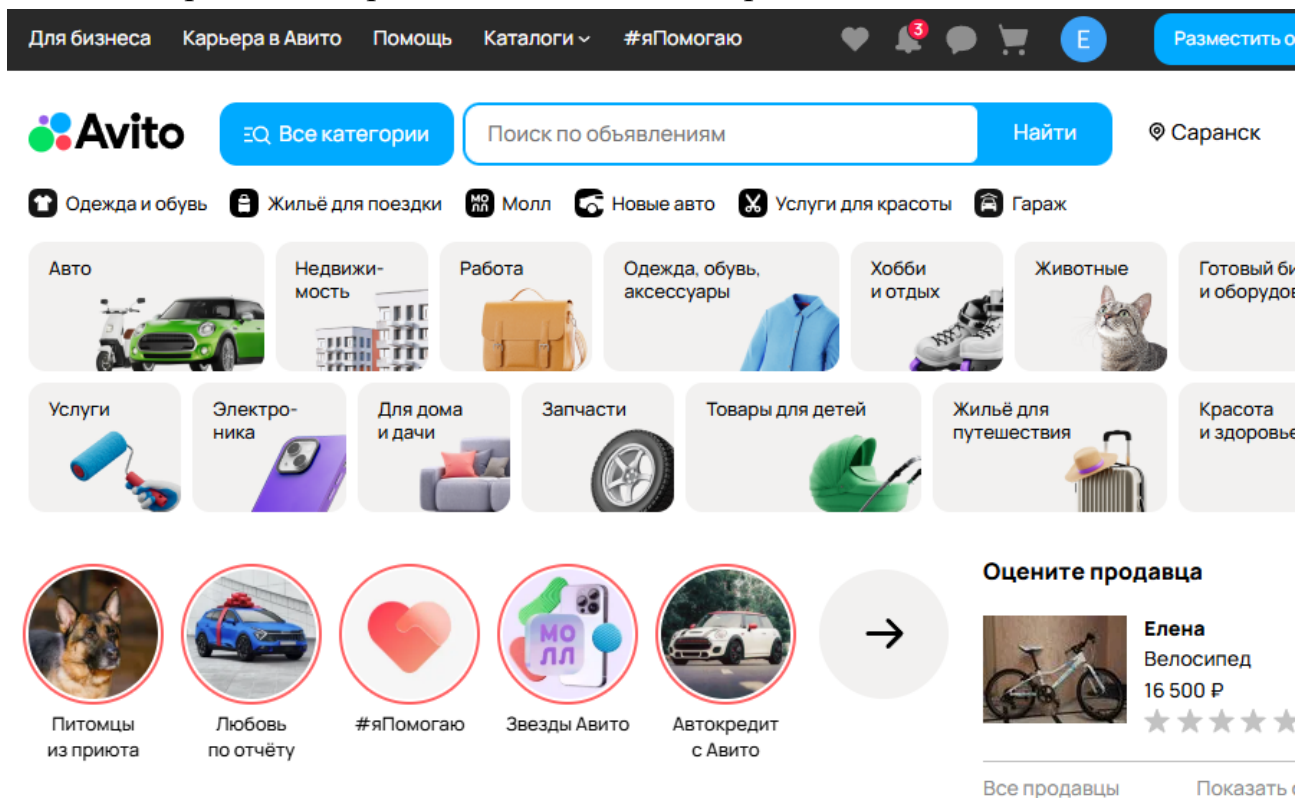


Рисунок 1 – Главная страница сайта после регистрации

Тест кейс 2 (Позитивный): Расчет стоимости доставки.

Кейс: Пользователь рассчитывает стоимость доставки товара.

Шаги:

1. Поиск товара.
2. Переход на страницу товара.
3. Нажатие кнопки «Рассчитать доставку».
4. Ввод параметров.

Ожидаемый результат: Отображение стоимости и сроков.

Код тест кейса:

```
def test_delivery_calculation():
    driver = webdriver.Chrome()
    driver.get("https://www.avito.ru/velosipedy")
    try:
        # Шаг 1: Выбрать товар
        item = WebDriverWait(driver, 10).until(
            EC.element_to_be_clickable((By.XPATH, "//div[@class='item'][1]"))
        )
        item.click()
        time.sleep(3)

        # Шаг 2: Открыть калькулятор
        calc_btn = driver.find_element(By.XPATH, "//button[contains(text(),
'Рассчитать доставку')]]")
        calc_btn.click()
        time.sleep(2)

        # Шаг 3: Заполнить параметры
        from_city = driver.find_element(By.ID, "fromCity")
        from_city.send_keys("Москва")
        to_city = driver.find_element(By.ID, "toCity")
        to_city.send_keys("Санкт-Петербург")
        time.sleep(2)

        # Проверка результата
        result = WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.CLASS_NAME, "delivery-cost"))
        )
        assert in result.text
    finally:
        driver.quit()
```

Сохраненные при выполнении теста скриншоты:

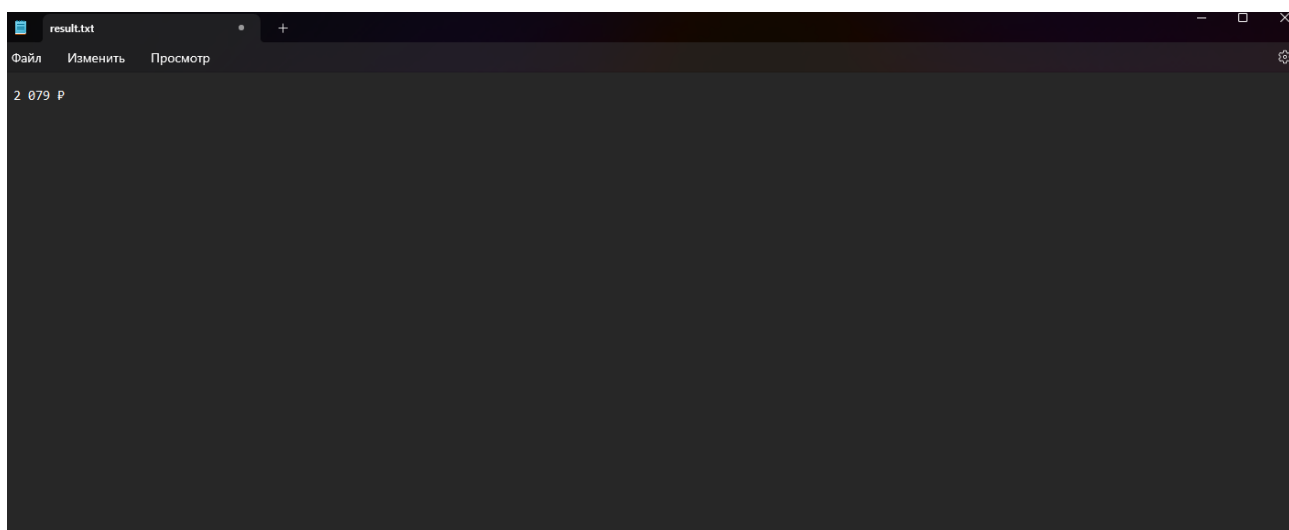


Рисунок 2 – Стоимость доставки в файле «result.txt»

Тест кейс 3 (Негативный): Оплата с некорректной картой.

Кейс: Ввод неверных данных карты.

Шаги:

1. Переход к оплате.
2. Ввод невалидных данных.
3. Попытка оплаты.

Ожидаемый результат: Сообщение об ошибке.

Код тест кейса:

```
def test_invalid_payment():
    driver = webdriver.Chrome()
    driver.get("https://www.avito.ru/item123")
    try:
        # Шаг 1: Перейти к оплате
        pay_btn = WebDriverWait(driver, 10).until(
            EC.element_to_be_clickable((By.XPATH,
            "//button[contains(text(), 'Купить с доставкой')]"))
        pay_btn.click()
        time.sleep(3)

        # Шаг 2: Ввод данных карты
        card_num = driver.find_element(By.ID,
            "cardNumber")
        card_num.send_keys("1234-5678-9012-3456")
        expiry = driver.find_element(By.ID, "expiryDate")
        expiry.send_keys("01/25")
        cvv = driver.find_element(By.ID, "cvv")
        cvv.send_keys("12")
        time.sleep(2)

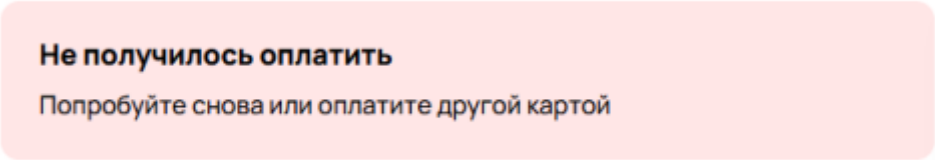
        # Шаг 3: Отправить форму
        submit_pay = driver.find_element(By.XPATH,
            "//button[@type='submit']")
        submit_pay.click()
        time.sleep(5)

        # Проверка ошибки
        error_msg = WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.CLASS_NAME,
            "error-message"))
        )
```



```
assert "Неверные данные карты" in error_msg.text
finally:
    driver.quit()
```

Сохраненные при выполнении теста скриншоты:

A screenshot of a payment error message displayed in a light red rounded rectangle. The text is in Russian. The first line is bold, and the second line is in a smaller font.

Не получилось оплатить

Попробуйте снова или оплатите другой картой

Рисунок 3 – Главная страница