МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра Вычислительные системы и технологии

ОТЧЕТ

По лабораторной работе № 2

по дисциплине

Тестирование ПО

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Скорынин С.С.

(подпись)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Захаров Д.С.

(подпись)

С22-СИБ

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2025

**Цель работы:** знакомство с автоматизацией функционального тестирования при помощи C#, NUnit и Selenium WebDriver.

**Введение:**

Современные программные приложения становятся всё более сложными, что повышает требования к их качеству и надежности. Одним из ключевых этапов обеспечения качества является функциональное тестирование, которое проверяет соответствие системы заявленным требованиям. Однако ручное тестирование часто оказывается трудоемким, медленным и подверженным человеческим ошибкам.

Автоматизация функционального тестирования позволяет значительно ускорить процесс проверки, повысить его точность и обеспечить возможность частого запуска тестов, что особенно важно в условиях непрерывной интеграции и доставки (CI/CD). В данной работе рассматриваются инструменты автоматизации тестирования на стеке **C#, NUnit и Selenium WebDriver**, которые предоставляют мощные возможности для создания, выполнения и поддержки автотестов веб-приложений.

**Описание проделанной работы:**

1. **Изучение инструментов тестирования**:

Для автоматизации функционального тестирования веб-приложения использовались следующие инструменты:

* **NUnit** – фреймворк для написания и запуска тестов в .NET, предоставляющий удобные атрибуты для структурирования тестовых сценариев.
* **Selenium WebDriver** – библиотека для автоматизации взаимодействия с браузером, позволяющая имитировать действия пользователя (клики, ввод текста, навигацию).
* **ChromeDriver** – драйвер для управления браузером Google Chrome.

Для повышения надежности тестов были применены:

* **Явные ожидания (WebDriverWait)** – вместо фиксированных задержек (Thread.Sleep), что делает тесты стабильнее и быстрее.
* **Обработка исключений** – автоматическое создание скриншотов при падении тестов для упрощения анализа ошибок.
* **Гибкие селекторы** – использование нескольких вариантов CSS-селекторов для устойчивости к изменениям вёрстки.

1. **Разработка UI-тестов**

Были написаны автотесты для проверки ключевых функций сайта **Neocities**:

**Основные тестовые сценарии:**

* **Проверка заголовка страницы (Test1\_CheckPageTitle)** – валидация корректности отображения названия сайта.
* **Проверка видимости основных элементов (Test2\_CheckMainElementsVisibility)** – логотипа, кнопок регистрации и входа.
* **Поиск сайтов (Test3\_SearchSites)** – проверка работы поиска по ключевому слову.
* **Тестирование формы регистрации (Test4\_OpenSignUpForm, Test5\_InvalidSignUpAttempt)** – открытие формы и проверка валидации неверных данных.
* **Тестирование формы входа (Test6\_OpenLoginForm, Test7\_InvalidLoginAttempt)** – проверка обработки некорректных учетных данных.
* **Навигация по сайту (Test8\_NavigateToBrowsePage, Test9\_FilterSitesByTag)** – переход на страницу просмотра сайтов и фильтрация по тегам.
* **Просмотр деталей сайта (Test10\_ViewSiteDetails)** – проверка открытия карточки сайта.
* **Проверка страницы с обучающими материалами (Test11\_CheckTutorialsPage)** – валидация наличия и содержания руководств.

**Особенности реализации:**

* **Параметризованные ожидания** – использование WebDriverWait для динамического ожидания элементов.
* **Мульти-ассерты (Assert.Multiple)** – группировка проверок для удобства анализа.
* **Гибкие селекторы** – альтернативные варианты поиска элементов для устойчивости тестов.
* **Автоматические скриншоты** – сохранение снимков экрана при ошибках с понятными именами файлов.

#### ****Реализация устойчивых тестов****

Для повышения стабильности тестов были применены следующие подходы:

* **Обработка cookie-баннеров** – автоматическое закрытие всплывающих уведомлений, если они появляются.
* **Оптимизация времени ожидания** – уменьшение ImplicitWait в пользу явных ожиданий.
* **Улучшенная обработка ошибок** – перехват исключений с детализацией в логах.

#### ****Анализ результатов****

Все тесты успешно проходят проверку основных сценариев взаимодействия с сайтом. В случае падения теста:

* Генерируется **скриншот** с именем, соответствующим названию теста.
* Выводится **подробное сообщение об ошибке**, включая локатор элемента и контекст теста.

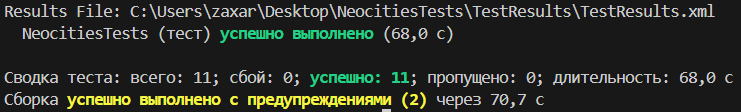


Рис.1 (Результат выполнения тестов для сайта Neocities.org)

**Вывод к работе:**

В ходе работы были успешно автоматизированы ключевые сценарии тестирования веб-приложения **Neocities** с использованием **NUnit** и **Selenium WebDriver**. Реализованные тесты отличаются:

* **Стабильностью** за счет явных ожиданий и обработки динамических элементов.
* **Удобством отладки**благодаря автоматическим скриншотам и детализированным сообщениям об ошибках.
* **Гибкостью**благодаря использованию альтернативных селекторов и модульной структуре.