Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ITMO University

ЛАБОРАТОРНАЯ РАБОТА №3

По дисциплине Тестирование программного обеспечения

Тема работы End-to-end тестирование

Обучающийся: Белисов Глеб Андреевич

Факультет ПИН

Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи

Образовательная программа Программирование в инфокоммуникационных системах

2025

Описание проекта

Для выполнения работы был выбран проект TodoMVC в реализации на React. У него простой интерфейс и понятная структура. Присутствуют все основные сценарии работы с задачами: добавление, выполнение, удаление, фильтрация и очистка.
Это классический пример «списка задач», что делает его удобным для демонстрации методов E2E-тестирования.

Тесты

Тест 1 — Добавление новой задачи

Добавление задач — это базовая функция TodoMVC. Если она не работает, все остальные функции теряют смысл, так как нечего выполнять, фильтровать или удалять. Тест проверяет основной пользовательский путь «создание задачи».

Проверяет базовый функционал приложения, работу интерфейса и корректного отображения элементов.

```
def test_add_new_todo():
    with sync_playwright() as p:
        browser = p.chromium.launch(headless=False, slow_mo=100)
        page = browser.new_page()

        page.goto("https://todomvc.com/examples/react/dist/")

        page.fill(".new-todo", "Buy Milk")
        page.keyboard.press("Enter")

        assert page.locator(".todo-list li").count() == 1
        assert page.locator(".todo-list li label").inner_text() == "Buy Milk"

        browser.close()
```

Тест 2 — Завершение задачи и фильтры

После добавления задач пользователю важно уметь отслеживать их состояние. Фильтры помогают управлять списком, особенно при большом количестве задач. Этот тест проверяет не только изменение статуса задачи, но и корректность фильтров

Проверка логики изменения статуса задач, работы фильтров и корректного UX. Обеспечение правильной работы функций, зависящих от статуса задач.

```python
def test_complete_todo_and_filter():
    with sync_playwright() as p:
        browser = p.chromium.launch(headless=False, slow_mo=100)
        page = browser.new_page()

        page.goto("https://todomvc.com/examples/react/dist/")

        page.fill(".new-todo", "Write Report")
        page.keyboard.press("Enter")

        page.click(".toggle")

        assert "completed" in page.locator(".todo-list li").get_attribute("class")

        page.click("text=Active")
        assert page.locator(".todo-list li").count() == 0

        page.click("text=Completed")
        assert page.locator(".todo-list li").count() == 1

        browser.close()
```

Тест 3 — Удаление задачи

Удаление задач — важная функция для управления списком и поддержания порядка. Тест проверяет, что кнопка удаления работает и изменения корректно отображаются в интерфейсе.

```python
def test_delete_todo():
    with sync_playwright() as p:
        browser = p.chromium.launch(headless=False, slow_mo=100)
        page = browser.new_page()

        page.goto("https://todomvc.com/examples/react/dist/")

        page.fill(".new-todo", "Task to delete")
        page.keyboard.press("Enter")

        page.hover(".todo-list li")
        page.click(".destroy", force=True)

        assert page.locator(".todo-list li").count() == 0

        browser.close()
```

Тест 4 — Добавление пустой задачи

Проверка граничных случаев (валидности ввода) — ключевая часть тестирования. Пустые задачи должны игнорироваться, чтобы избежать некорректных данных в списке.

Проверяет валидацию пользовательского ввода и устойчивость приложения к некорректным действиям пользователя.

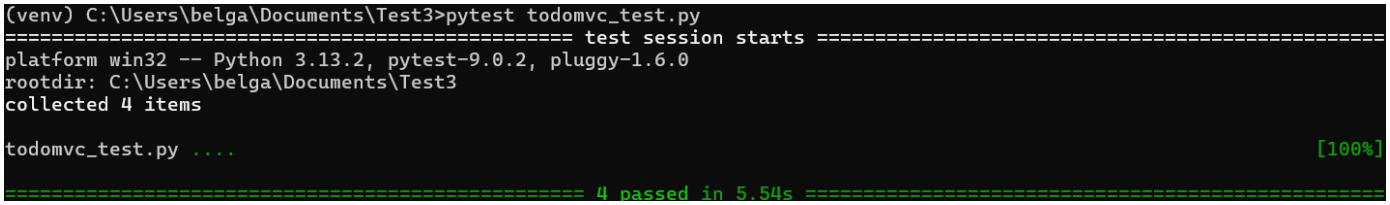
```python
def test_add_empty_todo():
    with sync_playwright() as p:
        browser = p.chromium.launch(headless=False, slow_mo=100)
        page = browser.new_page()
        page.goto("https://todomvc.com/examples/react/dist/")

        page.fill(".new-todo", " ")
        page.keyboard.press("Enter")

        assert page.locator(".todo-list li").count() == 0

        browser.close()
```

Результаты



Все ключевые сценарии TodoMVC работают корректно. Пользовательский интерфейс стабилен, а логика приложения правильно обрабатывает действия пользователя. Применение Playwright показало себя удобным инструментом для написания и запуска E2E-тестов. Лабораторная работа продемонстрировала процесс анализа пользовательских сценариев, проектирования и выполнения E2E-тестов для веб-приложений. Созданные тесты покрывают основные функции приложения и могут служить основой для регулярного автоматического контроля качества.