Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

Факультет Инфокоммуникационных Технологий

Лабораторная работа №1

по дисциплине «Тестирование программного обеспечения»

Выполнила:

Соколова Дарья Максимовна

Группа: K3323

Проверил:

Кочубеев Николай Сергеевич

Санкт-Петербург,

2024

Цель работы

Научиться писать unit-тесты для существующего проекта, проанализировать что и как тестировать и подготовить отчет о проведенном тестировании.

Задачи

1. Выбрать репозиторий с GitHub;
2. Проанализировать тестируемые функциональности;
3. Написать тесты с использованием AAA и FIRST principles;
4. Составить отчет по выполненной работе.

Ход работы

1. Выбор репозитория с GitHub

Для выполнения лабораторной работы был выбран проект https://github.com/ShaifArfan/react-todo-app. Это проект написан на React, представляет собой туду-лист с возможностью создавать, редактировать и удалять задачи, а также фильтровать по выполненным и невыполненным задачам.

2. Анализ тестируемых функциональностей

Выделим **основные функциональные элементы**, для которых необходимы тесты:

- addTodo: Добавление задачи;
- updateTodo: Редактирование задачи, отметка задачи как выполненной;
- deleteTodo: Удаление задачи;
- updateFilterStatus: Фильтрация задач.

К **критическим частям** системы относятся:

- Логика, связанная с добавлением, удалением и редактированием задач, состояние должно обновляться;
- Валидация формы добавления и редактирования задач – пустая задача не должна добавляться;
- Проверка фильтрации – должны отображаться правильным задачи в зависимости от выбранного фильтра.

Отметим важные случаи использования (**use cases**):

- Проверка, что при добавлении и редактировании задача с пустым текстом не сохраняется;
- При нажатии кнопки «Cancel» задача не должна сохраняться;
- Корректное изменение статуса задачи и ее верное отображение в соответствующем фильтре.

3. Написание тестов

Тесты были написаны с использованием AAA (arrange, act, assert) и FIRST (fast, isolated, repeatable, self-validating, timely) principles. Повторяющиеся части кода вынесены в функции (рисунок 1). Тесты представлены на рисунках 2-3.

.

```
import { test, expect } from '@playwright/test';

test.beforeEach(async ({ page }) => {
  await page.goto('https://wc-react-todo-app.netlify.app/');
});

async function addTask(page, title, status, toSubmit = true) {
  await page.locator('button[type="button"]:has-text("Add Task")').click();
  await page.fill('input[id="title"]', title);
  await page.getByLabel('Status').selectOption(status);
  if (toSubmit) {
    await page.locator('button[type="submit"]:has-text("Add Task")').click();
  } else {
    await page.locator('button:has-text("Cancel")').click();
  }
}

async function editTask(page, task, newTitle, newStatus) {
  const editButton = task.getByRole('button').nth(-1);
  await editButton.click();
  await page.fill('input[id="title"]', newTitle);
  await page.getByLabel('Status').selectOption(newStatus);
  await page.locator('button[type="submit"]:has-text("Update Task")').click();
}

async function deleteTask(page, i) {
  const deleteButton = page.getByRole("button").nth((i - 1) * 2 + 1)
  await deleteButton.click();
}

async function deleteAllTasks(page) {
  const wrapper = page.locator('div.app_content__wrapper__Mm7EF')
  const count = await wrapper.childElementCount;
  for (let i = 0; i < count; i++) {
    await deleteTask(page, i)
  }
}

async function expectTextVisible(page, text, isVisible = true) {
  const textLocator = page.locator(`text=${text}`);
  if (isVisible) {
    await expect(textLocator).toBeVisible();
  } else {
    await expect(textLocator).not.toBeVisible();
  }
}
```

Рисунок 1 - Функции

```
// Тест на удаление задачи
test('should delete a task', async ({ page }) => {
  const taskToDelete = 'Task to be deleted';

  await addTask(page, taskToDelete, 'Incomplete')
  await deleteTask(page, 1)

  await expectTextVisible(page, taskToDelete, false)
});


// Тест на фильтрацию задач
test('should filter tasks by status', async ({ page }) => {
  const incompleteTask = 'Incomplete Task';
  const completedTask = 'Completed Task';
  const incompleteStatus = 'Incomplete'
  const completedStatus = 'Completed'

  await deleteAllTasks(page)
  await addTask(page, incompleteTask, incompleteStatus)
  await addTask(page, completedTask, completedStatus)

  await page.click('select#status');
  await page.locator('select#status').selectOption('Completed');

  await expectTextVisible(page, incompleteTask, false)
  await expectTextVisible(page, completedTask)
});


// Тест на выполнение задачи
test('should not see task if completed', async ({ page }) => {
  const taskToCompelete = 'Task to Complete';
  const taskNotToComplete = 'Task not to Complete';
  const incompleteStatus = 'Incomplete'

  await deleteAllTasks(page)
  await addTask(page, taskNotToComplete, incompleteStatus)
  await addTask(page, taskToCompelete, incompleteStatus)

  await page.click('select#status');
  await page.locator('select#status').selectOption(incompleteStatus);

  const task = page.locator('div.todoItem_item__fnR7B').nth(-1)
  await task.locator('div.todoItem_svgBox__z1vm6').click()

  await expectTextVisible(page, taskToCompelete, false)
  await expectTextVisible(page, taskNotToComplete)
});
```

Рисунок 2 – Тесты

```
// Тест на добавление задачи
test('should add a new task', async ({ page }) => {
  const firstTask = 'First Task'
  const taskStatus = 'Incomplete'

  await addTask(page, firstTask, taskStatus)
  await expectTextVisible(page, firstTask)
});

// Тест на добавление пустой задачи
test('should not add an empty task', async ({ page }) => {
  const emptyTask = ''
  const taskStatus = 'Incomplete'
  const errorMessage = 'Please enter a title'

  await addTask(page, emptyTask, taskStatus)
  await expectTextVisible(page, errorMessage)
});

// Тест на отмену добавления задачи
test('should cancel adding a task', async ({ page }) => {
  const secondTask = 'Second Task'
  const taskStatus = 'Incomplete'

  await addTask(page, secondTask, taskStatus, false)
  await expectTextVisible(page, secondTask, false)
});

// Тест на редактирование задачи
test('should edit an existing task', async ({ page }) => {
  const originalTask = "Task to Edit";
  const updatedTask = "Task after Editing";
  const originalStatus = "Incomplete"
  const newStatus = "Completed"
  await addTask(page, originalTask, originalStatus)

  const task = page.locator('div.todoItem_item__fnR7B').nth(-1)
  await editTask(page, task, updatedTask, newStatus)

  await expectTextVisible(page, originalTask, false)
  await expectTextVisible(page, updatedTask, true)
});
```

Рисунок 3 - Тесты

Тесты подтвердили корректность работы приложения (рисунок 4).

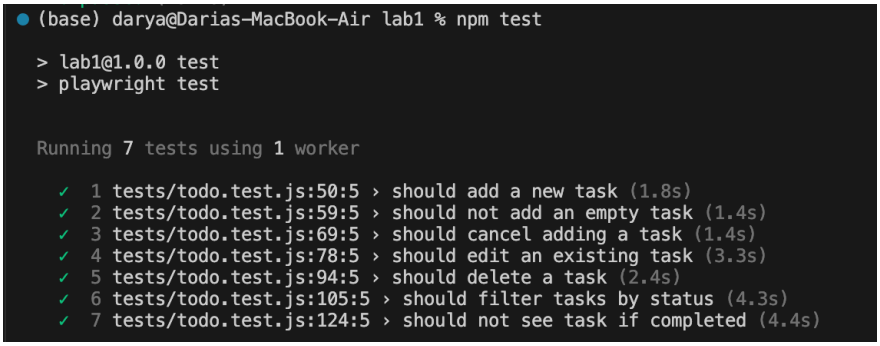

Рисунок 4 – Запуск тестов

Вывод

В данной лабораторной работе был выбран репозиторий с GitHub, проанализирована тестируемая функциональность проекта, написаны тесты с использованием AAA и FIRST principles, а также составлен отчет по выполненной работе.

Ссылка на репозиторий с тестами:
https://github.com/daryasokolova04/testing_PO