Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Факультет Инфокоммуникационных Технологий

Тестирование программного обеспечения

Лабораторная работа 3

Выполнил:

Колсанов Я. И.

Проверил:

Кочубеев Н. С.

Санкт-Петербург, 2024

Цель: научиться проектировать, писать и применять end-to-end (E2E) тесты для проверки работы всей системы или ключевых пользовательских сценариев. Провести анализ пользовательских путей и подготовить отчёт о проведённом тестировании.

Задачи:

1. Выбрать открытый проект на платформе GitHub.

2. Проанализировать функциональность проекта и определить ключевые пользовательские пути.

Обратить особое внимание на:

- Сценарии, связанные с критическими функциями системы;

- Граничные случаи и сценарии ошибок, которые могут возникать при использовании приложения;

- Важные точки взаимодействия, такие как работа с интерфейсом пользователя или отправка запросов на сервер.

3. Написать минимум два E2E теста, которые охватывают основные пользовательские сценарии.

Ход работы

Ссылка на репозиторий с тестами:

https://github.com/yaroslavkolsanov/TestPO

1. Выбор проекта на GitHub

Был выбран

https://github.com/VipReaL/web-larek-frontend/tree/main?tab=readme-ov-file

Он представляет из себя «Web-ларёк» - интернет-магазин с товарами для веб-разработчиков. В нем можно посмотреть каталог товаров, добавить товары в корзину и делать заказы. Стек технологий: HTML, SCSS, TS, WebPack.

Тесты написаны на TypeScript, использован инструмент E2E тестирования Playwright.

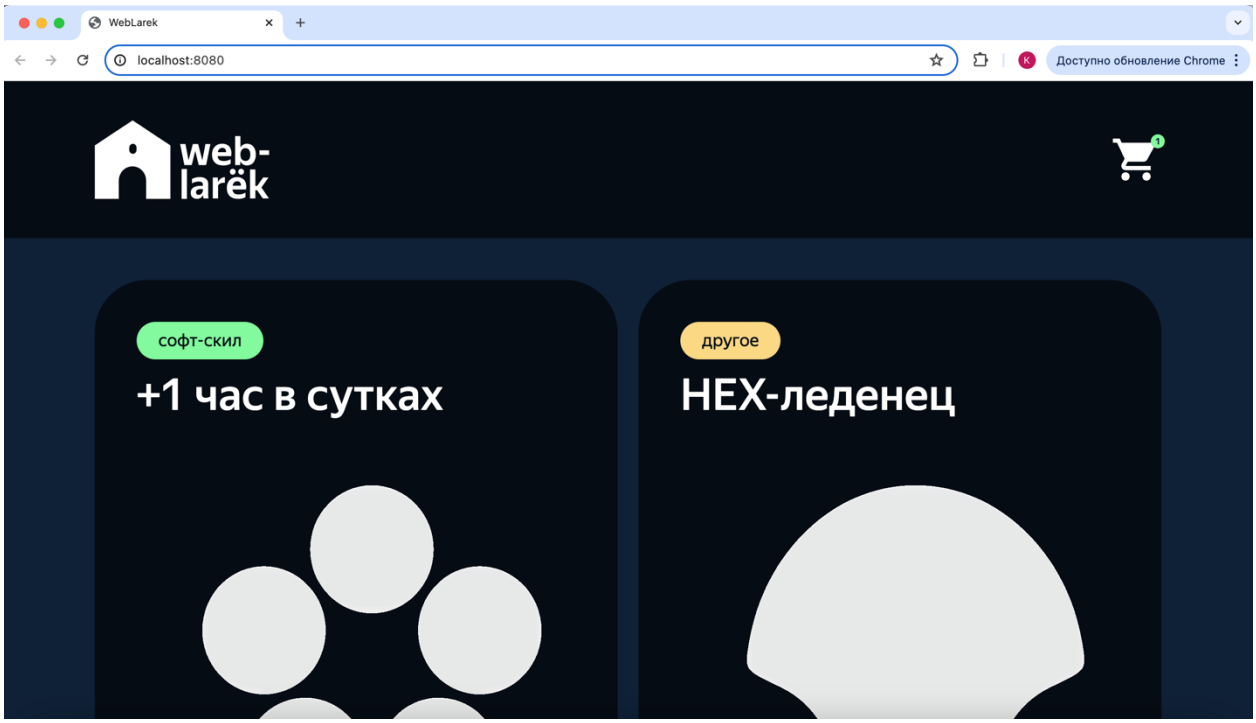

Рисунок 1 – Вид сайта в браузере

2. Анализ пользовательских сценариев

Ключевые пользовательские пути:

- Просмотр товаров на сайте

Пользователь просматривает каталог товаров, видит цены на каждый из них, заходит в карточку и читает описание.

- Добавление товара в корзину

  Пользователь выбирает товар, добавляет его в корзину.

- Проверка корзины

  Пользователь открывает корзину, проверяет товары и их количество.

- Оформление заказа

Пользователь вводит контактные данные, выбирает способ оплаты и завершает заказ.

Сценарии ошибок:

- Некорректный ввод данных в поля формы (например, ввод неверного адреса).
- Ожидание, что кнопки «Далее» и «Оплатить» становятся активными только при вводе всех необходимых данных.

3. Написание E2E тестов

В первом тесте была осуществлена проверка правильности вывода всех товаров на сайте, добавление одного товара в корзину и затем проверка содержимого корзины. Код теста представлен на рисунке ниже.

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

test('API should return list of products', async ({ page }) => {
  await page.goto('http://localhost:8080/');
  await page.waitForSelector('.gallery__item.card', { timeout: 5000 });
  const productList = page.locator('.gallery__item.card');
  const productCount = await productList.count();
  expect(productCount).toBeGreaterThan(0);

  for (let i = 0; i < productCount; i++) {
    const productName = await productList.nth(i).locator('.card__title').innerText();
    expect(productName).not.toBe('');
    const productPrice = await productList.nth(i).locator('.card__price').innerText();
    expect(productPrice).not.toBe('');
  }

  const firstProduct = productList.nth(0);
  const firstProductName = await firstProduct.locator('.card__title').innerText();
  const firstProductPrice = await firstProduct.locator('.card__price').innerText();
  await firstProduct.click();
  const addToBasketButton = page.locator('#modal-container.modal_active .card__button');
  await addToBasketButton.waitFor({ state: 'visible', timeout: 10000 });
  await addToBasketButton.click();
  const basketCounter = await page.locator('.header__basket-counter').innerText();
  expect(basketCounter).toBe('1');

  const basketButton = page.locator('.header__basket');
  await basketButton.click();
  const basketItemName = await page.locator('#modal-container.modal_active .basket__item .card__title').innerText();
  const basketItemPrice = await page.locator('#modal-container.modal_active .basket__item .card__price').innerText();
  expect(basketItemName).toBe(firstProductName);
  expect(basketItemPrice).toBe(firstProductPrice);
  const totalBasketPrice = await page.locator('#modal-container.modal_active .basket__price').innerText();
  expect(totalBasketPrice).toBe(firstProductPrice);

   const closeButton = page.locator('#modal-container.modal_active .modal__close');
   await closeButton.click();
});
```

Рисунок 2 – Код файла product.test.ts

Во втором тесте была осуществлена проверка заказа товара после добавления его в корзину. Сначала выбираются способ оплаты и адрес доставки. В следующем модульном окне вводятся контактные данные: email и номер телефона. Код теста представлен ниже.

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

test('Add first product to basket and complete the order', async ({ page }) => {
  await page.goto('http://localhost:8080/');

  await page.waitForSelector('.gallery__item.card', { timeout: 5000 });

  const productList = page.locator('.gallery__item.card');
  const firstProduct = productList.nth(0);
  const firstProductName = await firstProduct.locator('.card__title').innerText();
  const firstProductPrice = await firstProduct.locator('.card__price').innerText();
  await firstProduct.click();

  const addToBasketButton = page.locator('#modal-container.modal_active .card__button');
  await addToBasketButton.waitFor({ state: 'visible', timeout: 5000 });
  await addToBasketButton.click();

  const basketCounter = await page.locator('.header__basket-counter').innerText();
  expect(basketCounter).toBe('1');

  const basketButton = page.locator('.header__basket');
  await basketButton.click();

  const basketItemName = await page.locator('#modal-container.modal_active .basket__item .card__title').innerText();
  const basketItemPrice = await page.locator('#modal-container.modal_active .basket__item .card__price').innerText();
  expect(basketItemName).toBe(firstProductName);
  expect(basketItemPrice).toBe(firstProductPrice);

  const checkoutButton = page.locator('#modal-container.modal_active .basket__button');
  await checkoutButton.waitFor({ state: 'visible', timeout: 5000 });
  await checkoutButton.click();

  const addressInput = page.locator('#modal-container.modal_active input[placeholder="Введите адрес"]');
  await addressInput.waitFor({ state: 'visible', timeout: 5000 });

  const onlinePaymentButton = page.locator('#modal-container.modal_active .order__buttons .button[name="card"]');
  await onlinePaymentButton.click();
  await addressInput.fill('г. Санкт-Петербург, ул. Ломоносова, д. 9');

  const nextButton = page.locator('#modal-container.modal_active .button.order__button');
  await expect(nextButton).toBeEnabled();
  await nextButton.click();

  const emailInput = page.locator('#modal-container.modal_active input[placeholder="Введите Email"]');
  const phoneInput = page.locator('#modal-container.modal_active input[placeholder="+7 ("]');
  await emailInput.fill('yaroslav.kolsanov@mail.ru');
  await phoneInput.fill('+7 (953) 011-00-07');
  const payButton = page.locator('#modal-container.modal_active .button:has-text("Оплатить")');
  await expect(payButton).not.toBeDisabled();

  await payButton.click();

  const successMessage = page.locator('#modal-container.modal_active .order-success__title');
  await expect(successMessage).toHaveText('Заказ оформлен');
});
```

Рисунок 3 – Код файла order.test.ts

В заключительном третьем тесте была осуществлена проверка обработки некорректных данных при заказе, а именно адреса, состоящего из одной буквы «А». Код теста представлен на рисунке.

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

test('Check invalid address input (one letter)', async ({ page }) => {
  await page.goto('http://localhost:8080/');

  await page.waitForSelector('.gallery__item.card', { timeout: 5000 });

  const productList = page.locator('.gallery__item.card');
  const firstProduct = productList.nth(0);
  const firstProductName = await firstProduct.locator('.card__title').innerText();
  const firstProductPrice = await firstProduct.locator('.card__price').innerText();
  await firstProduct.click();

  const addToBasketButton = page.locator('#modal-container.modal_active .card__button');
  await addToBasketButton.waitFor({ state: 'visible', timeout: 5000 });
  await addToBasketButton.click();

  const basketCounter = await page.locator('.header__basket-counter').innerText();
  expect(basketCounter).toBe('1');

  const basketButton = page.locator('.header__basket');
  await basketButton.click();

  const basketItemName = await page.locator('#modal-container.modal_active .basket__item .card__title').innerText();
  const basketItemPrice = await page.locator('#modal-container.modal_active .basket__item .card__price').innerText();
  expect(basketItemName).toBe(firstProductName);
  expect(basketItemPrice).toBe(firstProductPrice);

  const checkoutButton = page.locator('#modal-container.modal_active .basket__button');
  await checkoutButton.waitFor({ state: 'visible', timeout: 5000 });
  await checkoutButton.click();

  const addressInput = page.locator('#modal-container.modal_active input[placeholder="Введите адрес"]');
  await addressInput.waitFor({ state: 'visible', timeout: 5000 });

  await addressInput.fill('A');

  const nextButton = page.locator('#modal-container.modal_active .button.order__button');
  await expect(nextButton).toBeDisabled();

  await expect(page.locator('#modal-container.modal_active .form__errors')).toHaveText('Укажите настоящий адрес');
});
```

Рисунок 4 – Код файла invalid-address.test.ts

4. Результаты тестирования

Тесты успешно пройдены:

```
yaroslav.kolsanov@MacBook-Pro-Aroslav-3 web-larek-frontend-main % npx playwright test

Running 3 tests using 3 workers

  ✓  1 tests/order.test.ts:3:5 › Add first product to basket and complete the order (2.4s)
  ✓  2 tests/product.test.ts:3:5 › API should return list of products (2.0s)
  ✓  3 tests/invalid-address.test.ts:3:5 › Check invalid address input (one letter) (1.9s)

  3 passed (3.1s)
```

Рисунок 5 – Результаты тестирования

В результате тестирования были проверены все ключевые пользовательские сценарии веб-приложения, включая выбор товаров,

добавление товаров в корзину, оформление заказа и обработку ошибок при некорректном вводе данных. В одном из негативных тестов, когда был введён некорректный адрес (одна буква), приложение корректно отреагировало на ошибку, отключив кнопку "Далее" до исправления ввода. Тесты подтвердили стабильность основных функций, однако были выявлены зоны для улучшения в обработке ошибок, таких как улучшенная проверка валидности данных на этапе ввода. Так, например, валидность адреса определяется только длиной введенной строки, что, конечно же, не обеспечивает ввод достоверной информации.

Вывод: в результате выполнения лабораторной работы были написаны end-to-end (E2E) тесты для проверки ключевых пользовательских сценариев.