УДК 004.774:37

ОРГАНИЗАЦИЯ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ ОБРАЗОВАТЕЛЬНЫХ ВЕБ-ПРИЛОЖЕНИЙ БЕЗ ИСПОЛЬЗОВАНИЯ СЛОЖНЫХ ФРЕЙМВОРКОВ

Казанцев В.В.
Научный руководитель – канд. пед. наук, доцент Козлов С.В.
Университет СмолГУ
valera.kaz2017@yandex.ru

Введение

Современные образовательные веб-приложения требуют постоянного обновления, что делает автоматизированное тестирование неотъемлемой частью разработки. Особую значимость приобретает тестирование пользовательских интерфейсов, от корректной работы которых зависит качество взаимодействия обучающегося с материалом. Сложились два основных подхода к тестированию: сквозной (E2E) и модульный (компонентный). E2E-подход, обеспечивая высокую достоверность, сопряжён со значительными временными и вычислительными затратами [1]. Альтернативой выступает компонентное тестирование с использованием лёгких инструментов, таких как Vitest и Vue Test Utils [2]. Однако в отечественной литературе недостаточно освещено применение данных инструментов для тестирования именно образовательных интерфейсов — отсутствуют типовые решения для проверки форм, асинхронных операций и взаимодействия компонентов, что и определяет актуальность данной работы.

Основная часть

В отличие от традиционного E2E-тестирования, требующего развёртывания сложной инфраструктуры и запуска браузера, предлагаемый подход базируется на изолированной проверке компонентов с использованием связки Vitest и Vue Test Utils. Данный подход позволяет выявлять ошибки на ранних этапах разработки, не требует значительных вычислительных ресурсов и обеспечивает высокую скорость выполнения тестов за счёт работы в среде Node.js без необходимости рендеринга в реальном браузере. Как было отмечено во введении, в отечественной литературе недостаточно освещены вопросы применения лёгких инструментов тестирования именно в контексте образовательных интерфейсов и отсутствуют типовые решения для проверки форм, асинхронных операций и взаимодействия компонентов. Далее предлагаются конкретные реализации тестов для указанных сценариев, которые могут служить основой при построении системы тестирования образовательного веб-приложения. Формы ввода являются ключевым элементом любого образовательного веб-приложения: через них осуществляется приём ответов обучающихся, регистрация пользователей, отправка результатов и обратная связь. Корректная работа форм напрямую влияет на качество взаимодействия обучающегося с материалом, поэтому их тестирование требует особого внимания. Рассмотрим тестирование формы ввода ответа на задачу на примере vue-компонента AnswerForm: «import {mount} from '@vue/test-utils'; import {describe, it, expect} from 'vitest';import AnswerForm from './AnswerForm.vue'; describe('AnswerForm', () => {it('отображает сообщение об ошибке при пустом ответе', async () => {const wrapper = mount(AnswerForm); await wrapper.find('button[type="submit"]').trigger('click'); expect(wrapper.find('.error').exists()).toBe(true);expect(wrapper.find('.error').text()).toContain ('Поле ответа не может быть пустым');}); it('вызывает emit при отправке валидного ответа', async()=> {const wrapper=mount(AnswerForm); await wrapper.find('input').setValue('42');await wrapper.find('button[type="submit"]').trigger('click');

expect(wrapper.emitted('submit')).toBeTruthy();expect(wrapper.emitted('submit')[0]).toEqual(['42']);});});». Функция mount создаёт изолированный экземпляр компонента для тестирования. Метод find ищет элементы по CSS-селекторам. Метод trigger эмулирует события пользователя. Метод setValue устанавливает значение в поле ввода. Метод emitted проверяет, какие события были сгенерированы компонентом. Тест проверяет два сценария: отображение ошибки при пустом вводе и корректную эмиссию события при валидном ответе. Образовательные веб-приложения, как правило, строятся из множества взаимодействующих компонентов. Корректность передачи данных между компонентами напрямую влияет на целостность образовательного процесса — ошибка на этом уровне может привести к потере ответа обучающегося, неверному подсчёту баллов или отображению не того учебного материала. Рассмотрим тестирование такого взаимодействия на примере связи между списком вопросов QuestionList и дочерним компонентом QuestionItem: «it('передаёт выбранный ответ в родительский компонент', async () => {const wrapper = mount(QuestionList, { props: { questions: [{ id: 1, text: 'Вопрос', options: ['A', 'B'] }]}}); await wrapper.findComponent({ name: 'QuestionItem' }).vm.$emit('answer', { questionId: 1, answer: 'A' }); expect(wrapper.emitted('answer-selected')).toBeTruthy(); expect(wrapper.emitted('answer-selected')[0][0]).toEqual({questionId: 1, answer: 'A'});});». Данный тест проверяет, что родительский компонент QuestionList корректно обрабатывает событие, поступившее от дочернего компонента QuestionItem. На первом этапе с помощью метода mount создаётся экземпляр родительского компонента, которому через параметр props передаются тестовые данные, что имитирует реальную ситуацию, когда список вопросов получает данные из БД. Метод findComponent с параметром { name: 'QuestionItem' } находит внутри родительского компонента экземпляр дочернего компонента по его имени. Затем через обращение к vm.$emit программно генерируется событие answer с тестовыми данными. После эмуляции события проверяется, что родительский компонент сгенерировал собственное событие answer-selected (метод emitted возвращает массив всех событий), а также что переданные данные соответствуют ожидаемым. Вторая проверка гарантирует, что родитель не просто отреагировал на событие, но и корректно обработал поступившую информацию, не исказив её при передаче выше по иерархии компонентов или в хранилище состояния. Особенность данного подхода заключается в том, что тестируется именно связь между компонентами, а не каждый из них по отдельности. Это позволяет выявить ошибки, которые не видны при изолированном тестировании.

Выводы

Предложенный подход к тестированию образовательных интерфейсов на базе Vitest и Vue Test Utils позволяет существенно сократить временные и вычислительные затраты по сравнению с традиционными E2E-фреймворками. Тесты выполняются быстро, не требуют запуска браузера, просты в написании и поддержке. Разработанные тестовые сценарии для форм и взаимодействия компонентов могут быть непосредственно использованы при создании и сопровождении образовательных веб-приложений на Vue.js.

Литература

1. Дудак А.А. ОПТИМИЗАЦИЯ ПРОЦЕССОВ РАЗРАБОТКИ И ТЕСТИРОВАНИЯ С ПОМОЩЬЮ VITE, STORYBOOK И VITEST // Инновационная наука. - 2024. - № 9-1. - С. 21-25.
2. Yerburgh E. Testing Vue.js Applications . - 1-е изд. - Нью-Йорк: Manning, 2018. - 272 с.