

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский политехнический университет»

Кафедра «Инфокогнитивные технологии»
Образовательная программа «Веб-технологии»

Лабораторная работа № 8
по дисциплине «Программная инженерия»

Выполнил студент
группы 181-321 Карпов Вадим

Проверил:
преподаватель Будылина Евгения Александровна

Москва
2020

Модульное тестирование

Для тестирования веб-приложения были выбраны следующие инструменты: Jest, vue-test-utils, vue-jest.

Модульное тестирование — фундаментальная часть разработки программного обеспечения. В модульных тестах выполняются небольшие фрагменты кода в изоляции для упрощения добавления новых функциональных возможностей и отслеживания ошибок.

Однофайловые компоненты Vue позволяют просто писать модульные тесты для компонентов в изоляции. Это помогает разрабатывать новую функциональность с уверенностью, что это не ломает работу существующей, и помогает другим разработчикам понять, как работает компонент.

Тесты на примере компонента формы входа

```
import { shallowMount, createLocalVue } from '@vue/test-utils'
import Vuex from 'vuex'
import LoginForm from '../components/LoginForm.vue'

const localVue = createLocalVue()

localVue.use(Vuex)

describe('Компонент Входа', () => {
  let actions
  let store
  let mutations

  beforeEach(() => {
    actions = {
      createUser: jest.fn(),
      signIn: jest.fn(),
      signOut: jest.fn()
    };
    mutations = {
      openLoginForm: jest.fn(),
      closeLoginForm: jest.fn(),
      updateUser: jest.fn(),
    }
  })

  store = new Vuex.Store({
    state: {
      loginForm: false,
      user: null,
    },
    getters: {
      getLoginFormState: jest.fn(),
      getUser: jest.fn(),
      isAuthenticated: jest.fn()
    },
    actions,
    mutations
  })
})
```

```

test('Компонент является экземпляром Vue', () => {
  const wrapper = shallowMount(LoginForm, { store, localVue })

  expect(wrapper.findComponent(LoginForm).vm).toBeTruthy()
})

test('Кнопка отправки по умолчанию в статичном состоянии', () => {
  const wrapper = shallowMount(LoginForm, { store, localVue });
  const btn = wrapper.findComponent({ ref: 'sign' });

  expect(btn.props('loading')).toBe(false)
})

test('Компонент формы закрывается по клику', () => {
  const wrapper = shallowMount(LoginForm, { store, localVue })
  const form = wrapper.find('#popUp');

  form.trigger('click')

  expect(mutations.closeLoginForm).toHaveBeenCalled()
})

test('Осуществляется переход на страницу регистрации', () => {
  const $route = {
    name: "Registration"
  };
  const push = jest.fn();
  const $router = {
    push: jest.fn()
  };
  const wrapper = shallowMount(LoginForm, { store, localVue, mocks: { $route, $router } })
  const regBtn = wrapper.find('#reg');

  regBtn.trigger('click')

  expect(wrapper.vm.$route.name).toBe($route.name)
})

test('Компонент формы отправляет данные на сервер', () => {
  const wrapper = shallowMount(LoginForm, { store, localVue })
  wrapper.setData({
    login: 'testName@mail.com',
    password: '123456789'
  })

  const signInButton = wrapper.findComponent({ ref: 'sign' });

  signInButton.trigger('click')

```

```

        expect(actions.signIn).toHaveBeenCalledWith(expect.anything(), expect.objectContaining({
          'email': 'testName@mail.com',
          'password': '123456789'
        }))
      })
    })
  })
}

```

Результаты тестирования системы

The screenshot shows a VS Code editor with a Vue.js project. The left sidebar displays the file explorer with a tree view of the project structure, including components, plugins, router, store, styles, tests, and views. The main editor area shows a test file with a single test case for the CartItem component. The test case verifies that the component displays the correct prop data (name: 'hello', price: 190) and that the item count is updated correctly. The terminal at the bottom shows the coverage summary and test results.

```

// ...
}
}
})
}

const inputNumber = wrapper.find( selector: '#itemCount')
inputNumber.trigger( eventName: 'change')

expect(mutations.setCartItemCount).toHaveBeenCalled()
})

test('Component Displays Prop Data Correctly', () => {
  const wrapper = mount(CartItem, { options: {
    store,
    localVue,
    propsData: {
      item: {
        name: 'hello',
        price: 190
      }
    }
  })

  const itemName = wrapper.find( selector: '#itemName')
  const itemPrice = wrapper.find( selector: '#itemPrice')

  expect(itemName.text()).toBe( expected: 'hello')
  expect(itemPrice.text()).toBe( expected: '190 P')
})
}

```

```

===== Coverage summary =====
Statements   : 100% ( 10/10 )
Branches     : 100% ( 2/2 )
Functions    : 100% ( 7/7 )
Lines        : 100% ( 10/10 )
=====

Test Suites: 1 passed, 1 total
Tests:       8 passed, 8 total
Snapshots:   0 total
Time:        2.227 s
Ran all test suites.

```