

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №5

Автоматизация функционального тестирования приложений с
графическим интерфейсом пользователя

тема

Вариант 1

Преподаватель

А.С. Кузнецов

подпись, дата

Студент

В.А. Прекель

подпись, дата

Красноярск 2021

1 Цель работы

Изучить методологию автоматизации функционального тестирования приложения с графическим пользовательским интерфейсом.

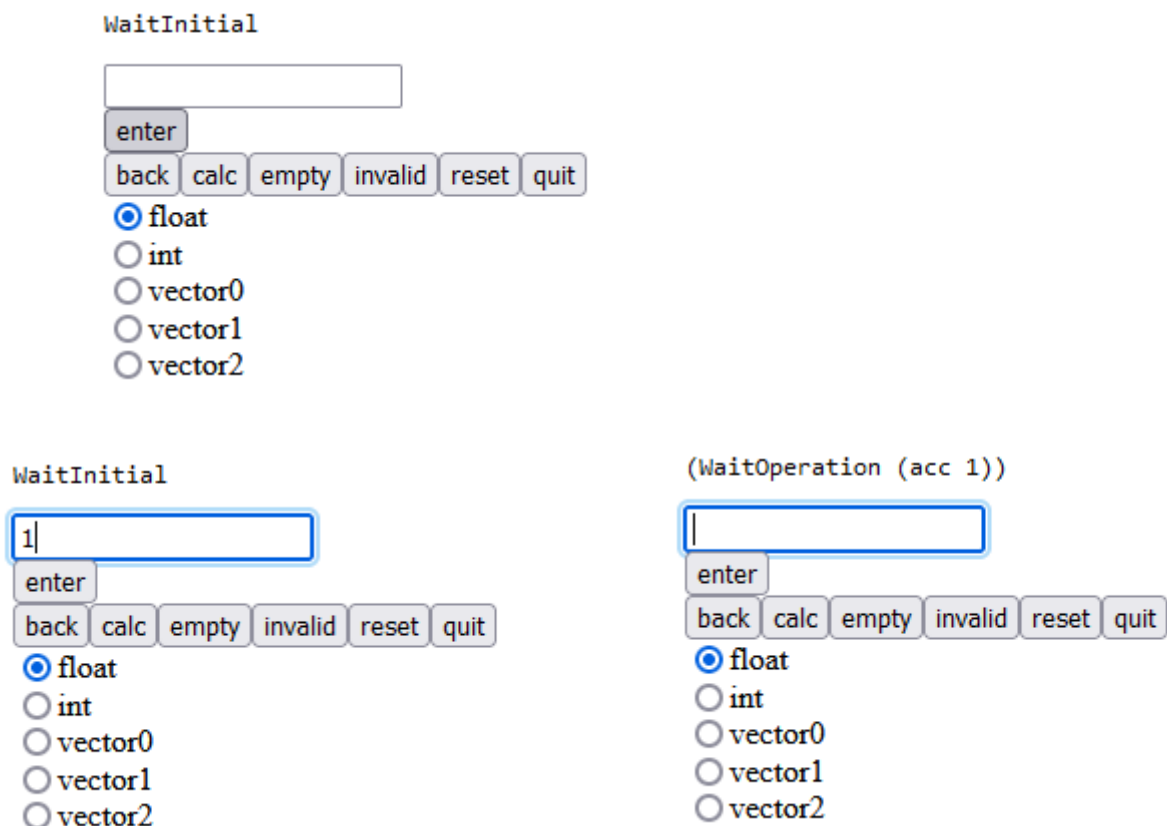
2 Общая постановка задачи

1. Научиться пользоваться фикстурами для автоматического заполнения компонентов;
2. Научиться проводить функциональное тестирование при помощи тестовых заглушек.

3 Ход работы

Для работы был выбран язык OCaml, фреймворк Bonsai для интерфейса, фреймворк для функционального тестирования Playwright. А так же для тестирования с заглушками библиотеки для написания тестов `ppx_inline_test`, `ppx_expect`, `bonsai.web_test`.

Реализован интерфейс для калькулятора из прошлых работ.



(WaitArgument (acc 1) (op Add))

enter

backcalcemptyinvalidresetquit

☒ float

☐ int

☐ vector0

☐ vector1

☐ vector2

(WaitArgument (acc 1) (op Add))

enter

backcalcemptyinvalidresetquit

☒ float

☐ int

☐ vector0

☐ vector1

☐ vector2

(Calculation (acc 1) (op Add) (arg 2))

enter

backcalcemptyinvalidresetquit

☒ float

☐ int

☐ vector0

☐ vector1

☐ vector2

(Calculation (acc 1) (op Add) (arg 2))

enter

backcalcemptyinvalidresetquit

☒ float

☐ int

☐ vector0

☐ vector1

☐ vector2

(WaitOperation (acc 3))

enter

backcalcemptyinvalidresetquit

☒ float

☐ int

☐ vector0

☐ vector1

☐ vector2

(Finish 3)

enter

backcalcemptyinvalidresetquit

☒ float

☐ int

☐ vector0

☐ vector1

☐ vector2

Рисунок 1 – Пошаговое пример работы «калькулятора»

Реализация выглядит так:

Листинг 1 – файл app/lib/lib.ml

```
open Core
open Bonsai_web
open Bonsai.Let_syntax
open Js_of_ocaml
open Lab_calculator

let quit () : never_returns =
  Dom_html.window##.location##reload;
  failwith "Quit"
;;

module MakeComponent (LF : Calculator.S) = struct
  let btn text action update =
    Vdom.(
      Node.button
      ~attr:(Attr.many [ Attr.class_ text; Attr.on_click (fun _ -> update action) ])
      [ Node.text text ])
  ;;

  let back = btn "back" LF.Back
  let calc = btn "calc" LF.Calculate
  let empty = btn "empty" LF.Empty
  let invalid = btn "invalid" (LF.Invalid (Error.create_s (Sexp.Atom "")))
  let reset = btn "reset" LF.Reset

  let quit =
    Vdom.(
      Node.button
      ~attr:
        (Attr.many
         [ Attr.class_ "quit"; Attr.on_click (fun _ -> never_returns @@ quit ()) ])
      [ Node.text "quit" ])
  ;;

  let btns update =
    Vdom.(
      Node.div
      [ back update; calc update; empty update; invalid update; reset update; quit ])
  ;;

  let form update =
    let%sub text, set_text = Bonsai.state [%here] (module String) ~default_model:"" in
    let%arr text = text
    and set_text = set_text
    and update = update in
    let update a = Effect.(set_text "" >>= fun () -> update a) in
    let on_enter () = LF.line_to_action text |> update in
    Vdom.(
      Node.div
```

```

[ Node.create
  "form"
  ~attr:
    (Attr.many
      [ Attr.on_submit (fun _ ->
        Effect.Many
          [ Vdom.Effect.Prevent_default
            ; Vdom.Effect.Stop_propagation
            ; on_enter ()
          ])
      ])
  [ Node.input
    ~attr:
      (Attr.many
        [ Attr.class_ "form"
          ; Attr.on_input (fun _ -> set_text)
          ; Attr.value_prop text
        ])
      []
    ]
  ; Node.button
    ~attr:
      (Attr.many [ Attr.class_ "enter"; Attr.on_click (fun _ -> on_enter ()) ])
    [ Node.text "enter" ]
  ; btns update
])

;;

let component =
  let%sub state, update =
    Bonsai.state_machine0
      [%here]
      (module struct
        type t = LF.state [@@deriving sexp, equal]
      end)
      (module struct
        type t = LF.action [@@deriving sexp, equal]
      end)
      ~default_model:LF.initial
      ~apply_action:(fun ~inject:_ ~schedule_event:_ state action ->
        LF.update ~action state)
  in
  let%sub form = form update in
  let%arr state = state
  and form = form in
  Vdom.(
    Node.div
      [ Node.pre
        ~attr:(Attr.many [ Attr.class_ "state" ])
        [ [%sexp (state : LF.state)] |> Sexp.to_string_hum |> Node.text ]
      ; form
    ])
  ;;

```

```

end

module LF = MakeComponent (Calculator.CalculatorFloat)
module LI = MakeComponent (Calculator.CalculatorInt)
module L0 = MakeComponent (Calculator.CalculatorVector0)
module L1 = MakeComponent (Calculator.CalculatorVector1)
module L2 = MakeComponent (Calculator.CalculatorVector2)

module Var = struct
  type t =
    | Float
    | Int
    | Vector0
    | Vector1
    | Vector2
  [@@deriving sexp, equal]
end

let app =
  let%sub state, set_state = Bonsai.state [%here] (module Var) ~default_model:Var.Float
  in
  let name = "variants" in
  let%sub lf = LF.component in
  let%sub li = LI.component in
  let%sub l0 = L0.component in
  let%sub l1 = L1.component in
  let%sub l2 = L2.component in
  let%arr state = state
  and set_state = set_state
  and lf = lf
  and li = li
  and l0 = l0
  and l1 = l1
  and l2 = l2 in
  Vdom.(
    let radio v a =
      Node.div
      [ Node.input
        ~attr:(
          (Attr.many
            [ Attr.class_ ("radio-" ^ v)
              ; Attr.type_ "radio"
              ; Attr.name name
              ; Attr.value v
              ; Attr.on_change (fun _ _ -> set_state a)
              ; (if [%equal: Var.t] state a then Attr.checked else Attr.empty)
            ])
          []
        ; Node.label [ Node.text v ]
      ]
    in
    Node.div
    ~attr:(Attr.class_ "root")

```

```

[ (match state with
  | Float -> lf
  | Int -> li
  | Vector0 -> l0
  | Vector1 -> l1
  | Vector2 -> l2)
; radio "float" Float
; radio "int" Int
; radio "vector0" Vector0
; radio "vector1" Vector1
; radio "vector2" Vector2
])
;;

```

Листинг 2 – файл app/main.ml

```

open Bonsai_web

let (_ : _ Start.Handle.t) =
  Start.start
  Start.Result_spec.just_the_view
  ~bind_to_element_with_id:"app"
  Lab_calculator_app_lib.Lib.app
;;

```

Листинг 3 – файл app/index.html

```

<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <script defer src="main.bc.js"></script>
</head>

<body>
  <div id="app"></div>
</body>

</html>

```

Проведём тестирование с помощью фреймворка Playwright – который позволяет запускать тесты в браузере:

Листинг 4 – файл app_test/tests/app.spec.ts

```

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

test.beforeEach(async ({ page }) => {
  await page.goto(process.env.APP_URL);
});

```

```

test.describe('Generic tests', () => {
  test('Test init', async ({ page }, ti) => {
    await expect(page.locator('.form')).toHaveValue("");
    await expect(page.locator('.state')).toHaveText("WaitInitial");
    await expect(page.locator('.radio-float')).toBeChecked();
    await expect(page.locator('.radio-int')).not.toBeChecked();
    await expect(page.locator('.radio-vector0')).not.toBeChecked();
    await expect(page.locator('.radio-vector1')).not.toBeChecked();
    await expect(page.locator('.radio-vector2')).not.toBeChecked();
    await page.locator('.root').screenshot({ path: `screenshots/${ti.project.name}/1-
init.png` });
  });
})

test.describe('Test float calculator', () => {
  test('Simple test', async ({ page }, ti) => {
    await expect(page.locator('.state')).toHaveText("WaitInitial");
    await page.locator('.form').fill("1");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(WaitOperation (acc 1))");
    await page.locator('.form').fill("+");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(WaitArgument (acc 1) (op Add))");
    await page.locator('.form').fill("2");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(Calculation (acc 1) (op Add) (arg
2))");
    await page.locator('.form').fill("=");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(WaitOperation (acc 3))");
    await page.locator('.form').fill("");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(Finish 3)");
    await page.locator('.root').screenshot({ path: `screenshots/${ti.project.name}/2-
float.png` });
  });
})

test('Testing Back button', async ({ page }, ti) => {
  await expect(page.locator('.state')).toHaveText("WaitInitial");
  await page.locator('.form').fill("1");
  await page.locator('.enter').click();
  await expect(page.locator('.state')).toHaveText("(WaitOperation (acc 1))");
  await page.locator('.back').click();
  await expect(page.locator('.state')).toHaveText("WaitInitial");
  await page.locator('.form').fill("3");
  await page.locator('.enter').click();
  await expect(page.locator('.state')).toHaveText("(WaitOperation (acc 3))");
  await page.locator('.form').fill("+");
  await page.locator('.enter').click();
  await expect(page.locator('.state')).toHaveText("(WaitArgument (acc 3) (op Add))");
  await page.locator('.form').fill("2");
  await page.locator('.enter').click();

```



```

    await expect(page.locator('.state')).toHaveText("(Calculation (acc 3) (op Add) (arg
2))");
    await page.locator('.form').fill("=");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(WaitOperation (acc 5))");
    await page.locator('.form').fill("");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(Finish 5)");
    await page.locator('.root').screenshot({ path: `screenshots/${ti.project.name}/3-
float-back.png` });
  });

test('Testing Reset button', async ({ page }, ti) => {
  await expect(page.locator('.state')).toHaveText("WaitInitial");
  await page.locator('.form').fill("1");
  await page.locator('.enter').click();
  await expect(page.locator('.state')).toHaveText("(WaitOperation (acc 1))");
  await page.locator('.back').click();
  await expect(page.locator('.state')).toHaveText("WaitInitial");
  await page.locator('.form').fill("3");
  await page.locator('.enter').click();
  await expect(page.locator('.state')).toHaveText("(WaitOperation (acc 3))");
  await page.locator('.form').fill("+");
  await page.locator('.enter').click();
  await expect(page.locator('.state')).toHaveText("(WaitArgument (acc 3) (op Add))");
  await page.locator('.reset').click();
  await expect(page.locator('.state')).toHaveText("WaitInitial");
  await page.locator('.form').fill("1");
  await page.locator('.enter').click();
  await expect(page.locator('.state')).toHaveText("(WaitOperation (acc 1))");
  await page.locator('.form').fill("+");
  await page.locator('.enter').click();
  await expect(page.locator('.state')).toHaveText("(WaitArgument (acc 1) (op Add))");
  await page.locator('.form').fill("2");
  await page.locator('.enter').click();
  await expect(page.locator('.state')).toHaveText("(Calculation (acc 1) (op Add) (arg
2))");
  await page.locator('.form').fill("=");
  await page.locator('.enter').click();
  await expect(page.locator('.state')).toHaveText("(WaitOperation (acc 3))");
  await page.locator('.form').fill("");
  await page.locator('.enter').click();
  await expect(page.locator('.state')).toHaveText("(Finish 3)");
  await page.locator('.root').screenshot({ path: `screenshots/${ti.project.name}/4-
float-reset.png` });
});

test('Testing Empty and Calc buttons', async ({ page }, ti) => {
  await expect(page.locator('.state')).toHaveText("WaitInitial");
  await page.locator('.form').fill("1");
  await page.locator('.enter').click();
  await expect(page.locator('.state')).toHaveText("(WaitOperation (acc 1))");
  await page.locator('.back').click();

```

```

    await expect(page.locator('.state')).toHaveText("WaitInitial");
    await page.locator('.form').fill("1");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(Wait0operation (acc 1))");
    await page.locator('.form').fill("+");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(WaitArgument (acc 1) (op Add))");
    await page.locator('.form').fill("2");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(Calculation (acc 1) (op Add) (arg
2))");
    await page.locator('.calc').click();
    await expect(page.locator('.state')).toHaveText("(Wait0operation (acc 3))");
    await page.locator('.empty').click();
    await expect(page.locator('.state')).toHaveText("(Finish 3)");
    await page.locator('.root').screenshot({ path: `screenshots/${ti.project.name}/5-
float-empty-calc.png` });
  });
})

test.describe('Test vector2 calculator', () => {
  test('Simple test', async ({ page }, ti) => {
    await page.locator('.radio-vector2').check()
    await expect(page.locator('.state')).toHaveText("WaitInitial");
    await page.locator('.form').fill("(1,-123)");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(Wait0operation (acc (1 (-123
()))))");
    await page.locator('.form').fill("+");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(WaitArgument (acc (1 (-123 ()))
(op Add))");
    await page.locator('.form').fill("(2,12)");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(Calculation (acc (1 (-123 ())) (op
Add) (arg (2 (12 ())))))");
    await page.locator('.form').fill("=");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(Wait0operation (acc (3 (-111
()))))");
    await page.locator('.form').fill("");
    await page.locator('.enter').click();
    await expect(page.locator('.state')).toHaveText("(Finish (3 (-111 ())))");
    await page.locator('.root').screenshot({ path: `screenshots/${ti.project.name}/6-
vector2.png` });
  });
})

```

Листинг 5 – запуск тестов Playwright

```
vladislav@DESKTOP-NK6MA9B:~/Projects/software-testing/app_test$ npm run test  
  
> app_test@1.0.0 test  
> cross-env APP_URL=http://127.0.0.1:8080 playwright test tests/app.spec.ts
```

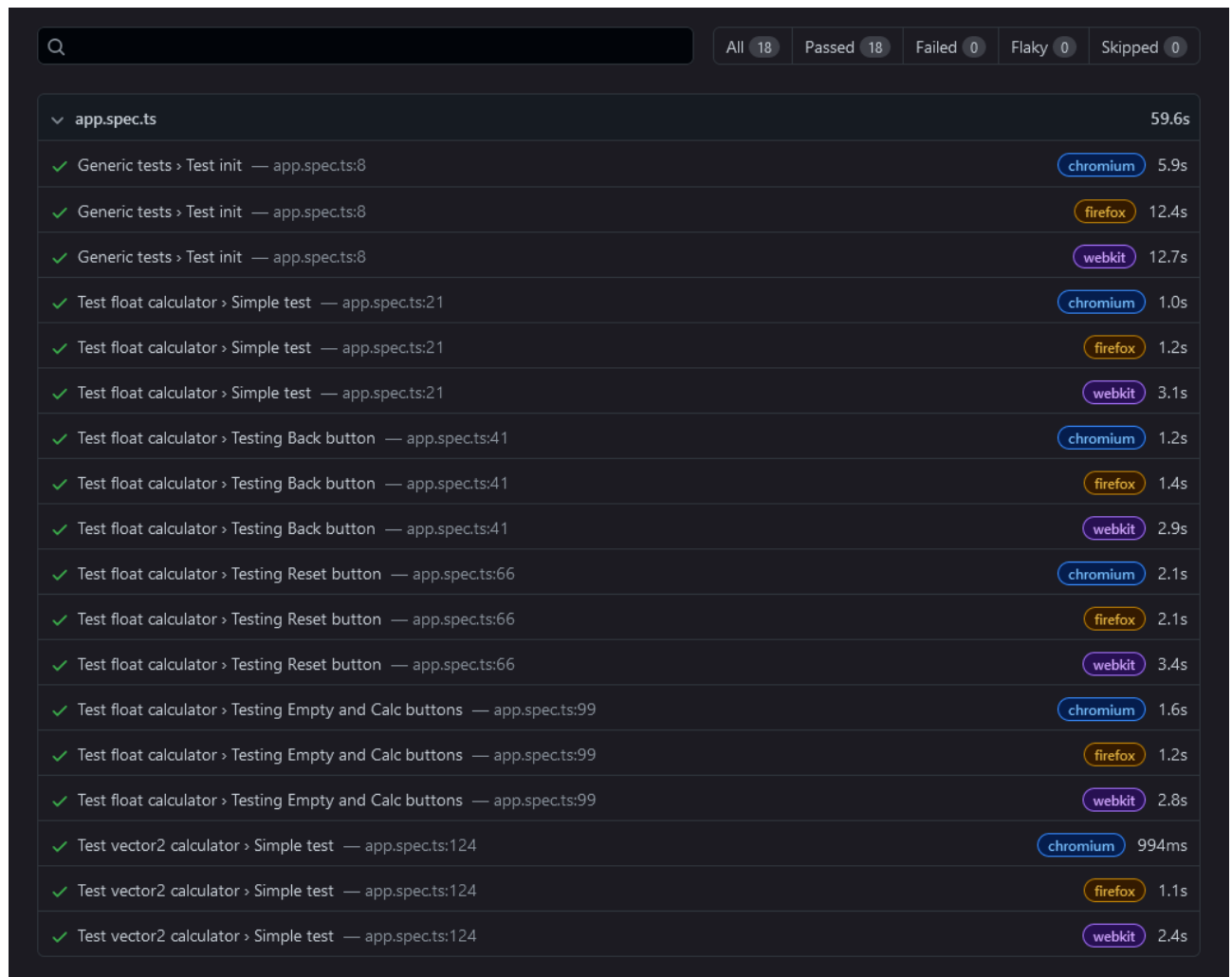
Running 18 tests using 3 workers

```
[WebServer] (node:8329) [DEP0066] DeprecationWarning:  
OutgoingMessage.prototype._headers is deprecated  
Slow test file: [webkit] > app.spec.ts (27s)  
Slow test file: [firefox] > app.spec.ts (19s)  
Consider splitting slow test files to speed up parallel execution
```

18 passed (29s)

To open last HTML report run:

```
npx playwright show-report
```



Q	All 18	Passed 18	Failed 0	Flaky 0	Skipped 0
▼ app.spec.ts					59.6s
✓ Generic tests > Test init — app.spec.ts:8	chromium				5.9s
✓ Generic tests > Test init — app.spec.ts:8	firefox				12.4s
✓ Generic tests > Test init — app.spec.ts:8	webkit				12.7s
✓ Test float calculator > Simple test — app.spec.ts:21	chromium				1.0s
✓ Test float calculator > Simple test — app.spec.ts:21	firefox				1.2s
✓ Test float calculator > Simple test — app.spec.ts:21	webkit				3.1s
✓ Test float calculator > Testing Back button — app.spec.ts:41	chromium				1.2s
✓ Test float calculator > Testing Back button — app.spec.ts:41	firefox				1.4s
✓ Test float calculator > Testing Back button — app.spec.ts:41	webkit				2.9s
✓ Test float calculator > Testing Reset button — app.spec.ts:66	chromium				2.1s
✓ Test float calculator > Testing Reset button — app.spec.ts:66	firefox				2.1s
✓ Test float calculator > Testing Reset button — app.spec.ts:66	webkit				3.4s
✓ Test float calculator > Testing Empty and Calc buttons — app.spec.ts:99	chromium				1.6s
✓ Test float calculator > Testing Empty and Calc buttons — app.spec.ts:99	firefox				1.2s
✓ Test float calculator > Testing Empty and Calc buttons — app.spec.ts:99	webkit				2.8s
✓ Test vector2 calculator > Simple test — app.spec.ts:124	chromium				994ms
✓ Test vector2 calculator > Simple test — app.spec.ts:124	firefox				1.1s
✓ Test vector2 calculator > Simple test — app.spec.ts:124	webkit				2.4s

Рисунок 2 – отчёт Playwright

В конце каждого теста делается снимок экрана, некоторые из них:

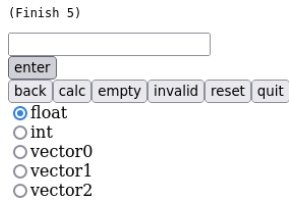


Рисунок 3 – Тест 'Testing Back button', браузер firefox

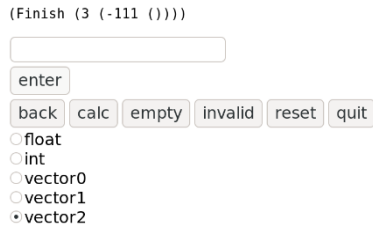


Рисунок 4 – Тест 'Test vector2 calculator', браузер на основе webkit

Проблему синхронизации потоков решает сам фреймворк. При таком методе тестирования существует проблема того, что элемент ещё не доступен, но фреймворк решает эту проблему благодаря функции “Auto-waiting”: «Перед выполнением действий Playwright проводит ряд проверок элементов на выполнимость, чтобы убедиться, что эти действия ведут себя так, как ожидается. Он автоматически ожидает прохождения всех соответствующих проверок и только после этого выполняет запрошенное действие. Если необходимые проверки не пройдут в течение заданного тайм-аута, действие завершится с ошибкой `TimeoutError`.»

К сожалению, этот метод тестирования ограничен тем, что тестирует готовое приложение целиком, без возможности внедрения подставных объектов (вернее, Playwright предоставляет возможность эмулировать запросы к серверу, но это в работе не используется). Поэтому можно использовать тестируемые возможности фреймворка Bonsai, а именно его модуля `Bonsai.web_test`, который позволяет тестировать компоненты и подкомпоненты путём рендеринга виртуального DOM. Далее тест компонента целиком и тест подкомпонента с применением подставного объекта.

Листинг 6 – файл `app/test/test.ml`

[open Core](#)
[open Bonsai_web_test](#)

```

open Lab_calculator
open Lab_calculator_app_lib.Lib

let%test_module "full component tests" =
  (module struct
    let%expect_test "full component test" =
      let handle = Handle.create (Result_spec.vdom Fn.id) app in
      Handle.show handle;
      [%expect
        {
          <div class="root">
            <div>
              <pre class="state"> WaitInitial </pre>
              <div>
                <form onsubmit>
                  <input class="form" #value="" oninput> </input>
                </form>
                <button class="enter" onclick> enter </button>
                <div>
                  <button class="back" onclick> back </button>
                  <button class="calc" onclick> calc </button>
                  <button class="empty" onclick> empty </button>
                  <button class="invalid" onclick> invalid </button>
                  <button class="reset" onclick> reset </button>
                  <button class="quit" onclick> quit </button>
                </div>
              </div>
            </div>
            <div>
              <input type="radio" name="variants" value="float" checked="" class="radio-
float" onchange> </input>
              <label> float </label>
            </div>
            <div>
              <input type="radio" name="variants" value="int" class="radio-int" onchange>
</input>
              <label> int </label>
            </div>
            <div>
              <input type="radio" name="variants" value="vector0" class="radio-vector0"
onchange> </input>
              <label> vector0 </label>
            </div>
            <div>
              <input type="radio" name="variants" value="vector1" class="radio-vector1"
onchange> </input>
              <label> vector1 </label>
            </div>
            <div>
              <input type="radio" name="variants" value="vector2" class="radio-vector2"
onchange> </input>
              <label> vector2 </label>
            </div>
          }
        ]
      )
    end
  end)

```

```

    </div> |});
Handle.input_text handle ~get_vdom:Fn.id ~selector:".form" ~text:"123";
Handle.show_diff handle;
[%expect
{
  <div class="root">
    <div>
      <pre class="state"> WaitInitial </pre>
      <div>
        <form onsubmit>
- |   <input class="form" #value="" oninput> </input>
+ |   <input class="form" #value="123" oninput> </input>
        </form>
        <button class="enter" onclick> enter </button>
        <div>
          <button class="back" onclick> back </button>
          <button class="calc" onclick> calc </button>
          <button class="empty" onclick> empty </button>
          <button class="invalid" onclick> invalid </button>
          <button class="reset" onclick> reset </button>
          <button class="quit" onclick> quit </button>
        </div>
      </div>
    </div>
    <div>
      <input type="radio" name="variants" value="float" checked=""
class="radio-float" onchange> </input>
      <label> float </label>
    </div> |});
Handle.click_on handle ~get_vdom:Fn.id ~selector:".enter";
Handle.show_diff handle;
[%expect
{
  <div class="root">
    <div>
- |   <pre class="state"> WaitInitial </pre>
+ |   <pre class="state"> (WaitOperation (acc 123)) </pre>
    <div>
      <form onsubmit>
- |   <input class="form" #value="123" oninput> </input>
+ |   <input class="form" #value="" oninput> </input>
      </form>
      <button class="enter" onclick> enter </button>
      <div>
        <button class="back" onclick> back </button>
        <button class="calc" onclick> calc </button>
        <button class="empty" onclick> empty </button>
        <button class="invalid" onclick> invalid </button>
        <button class="reset" onclick> reset </button>
        <button class="quit" onclick> quit </button>
      </div>
    </div>
  </div>
}

```

```

    <div>
      <input type="radio" name="variants" value="float" checked="" class="radio-
float" onchange> </input>
      <label> float </label>
    </div> |}];
Handle.input_text handle ~get_vdom:Fn.id ~selector:".form" ~text:"-";
Handle.show_diff handle;
[%expect
{|
  <div class="root">
    <div>
      <pre class="state"> (WaitOperation (acc 123)) </pre>
      <div>
        <form onsubmit>
-|      <input class="form" #value="" oninput> </input>
+|      <input class="form" #value="-" oninput> </input>
        </form>
        <button class="enter" onclick> enter </button>
        <div>
          <button class="back" onclick> back </button>
          <button class="calc" onclick> calc </button>
          <button class="empty" onclick> empty </button>
          <button class="invalid" onclick> invalid </button>
          <button class="reset" onclick> reset </button>
          <button class="quit" onclick> quit </button>
        </div>
      </div>
    </div>
    <div>
      <input type="radio" name="variants" value="float" checked="" class="radio-
float" onchange> </input>
      <label> float </label>
    </div> |}];
Handle.click_on handle ~get_vdom:Fn.id ~selector:".enter";
Handle.show_diff handle;
[%expect
{|
  <div class="root">
    <div>
-|      <pre class="state"> (WaitOperation (acc 123)) </pre>
+|      <pre class="state"> (WaitArgument (acc 123) (op Sub)) </pre>
      <div>
        <form onsubmit>
-|      <input class="form" #value="-" oninput> </input>
+|      <input class="form" #value="" oninput> </input>
        </form>
        <button class="enter" onclick> enter </button>
        <div>
          <button class="back" onclick> back </button>
          <button class="calc" onclick> calc </button>
          <button class="empty" onclick> empty </button>
          <button class="invalid" onclick> invalid </button>
          <button class="reset" onclick> reset </button>

```

```

        <button class="quit" onclick> quit </button>
      </div>
    </div>
  </div>
  <div>
    <input type="radio" name="variants" value="float" checked="" class="radio-
float" onchange> </input>
    <label> float </label>
  </div> |}];
Handle.input_text handle ~get_vdom:Fn.id ~selector:".form" ~text:"123";
Handle.show_diff handle;
[%expect
{|
  <div class="root">
    <div>
      <pre class="state"> (WaitArgument (acc 123) (op Sub)) </pre>
      <div>
        <form onsubmit>
- |   <input class="form" #value="" oninput> </input>
+ |   <input class="form" #value="123" oninput> </input>
        </form>
        <button class="enter" onclick> enter </button>
        <div>
          <button class="back" onclick> back </button>
          <button class="calc" onclick> calc </button>
          <button class="empty" onclick> empty </button>
          <button class="invalid" onclick> invalid </button>
          <button class="reset" onclick> reset </button>
          <button class="quit" onclick> quit </button>
        </div>
      </div>
    </div>
    <div>
      <input type="radio" name="variants" value="float" checked="" class="radio-
float" onchange> </input>
      <label> float </label>
    </div> |}];
Handle.click_on handle ~get_vdom:Fn.id ~selector:".enter";
Handle.show_diff handle;
[%expect
{|
  <div class="root">
    <div>
- |   <pre class="state"> (WaitArgument (acc 123) (op Sub)) </pre>
+ |   <pre class="state"> (Calculation (acc 123) (op Sub) (arg 123)) </pre>
      <div>
        <form onsubmit>
- |   <input class="form" #value="123" oninput> </input>
+ |   <input class="form" #value="" oninput> </input>
        </form>
        <button class="enter" onclick> enter </button>
        <div>
          <button class="back" onclick> back </button>

```



```

        <button class="calc" onclick> calc </button>
        <button class="empty" onclick> empty </button>
        <button class="invalid" onclick> invalid </button>
        <button class="reset" onclick> reset </button>
        <button class="quit" onclick> quit </button>
    </div>
</div>
</div>
<div>
    <input type="radio" name="variants" value="float" checked="" class="radio-
float" onchange> </input>
    <label> float </label>
</div> |}];
Handle.input_text handle ~get_vdom:Fn.id ~selector:".form" ~text:"=";
Handle.show_diff handle;
[%expect
{|
    <div class="root">
        <div>
            <pre class="state"> (Calculation (acc 123) (op Sub) (arg 123)) </pre>
            <div>
                <form onsubmit>
-|         <input class="form" #value="" oninput> </input>
+|         <input class="form" #value="=" oninput> </input>
                </form>
                <button class="enter" onclick> enter </button>
                <div>
                    <button class="back" onclick> back </button>
                    <button class="calc" onclick> calc </button>
                    <button class="empty" onclick> empty </button>
                    <button class="invalid" onclick> invalid </button>
                    <button class="reset" onclick> reset </button>
                    <button class="quit" onclick> quit </button>
                </div>
            </div>
        </div>
        <div>
            <input type="radio" name="variants" value="float" checked="" class="radio-
float" onchange> </input>
            <label> float </label>
        </div> |}];
Handle.click_on handle ~get_vdom:Fn.id ~selector:".enter";
Handle.show_diff handle;
[%expect
{|
    <div class="root">
        <div>
-|     <pre class="state"> (Calculation (acc 123) (op Sub) (arg 123)) </pre>
+|     <pre class="state"> (WaitOperation (acc 0)) </pre>
        <div>
            <form onsubmit>
-|         <input class="form" #value="=" oninput> </input>
+|         <input class="form" #value="" oninput> </input>

```

```

        </form>
        <button class="enter" onclick> enter </button>
        <div>
            <button class="back" onclick> back </button>
            <button class="calc" onclick> calc </button>
            <button class="empty" onclick> empty </button>
            <button class="invalid" onclick> invalid </button>
            <button class="reset" onclick> reset </button>
            <button class="quit" onclick> quit </button>
        </div>
    </div>
</div>
<div>
    <input type="radio" name="variants" value="float" checked="" class="radio-
float" onchange> </input>
    <label> float </label>
</div> |}];
Handle.input_text handle ~get_vdom:Fn.id ~selector:".form" ~text:"";
Handle.show_diff handle;
[%expect {|}|];
Handle.click_on handle ~get_vdom:Fn.id ~selector:".enter";
Handle.show_diff handle;
[%expect
{|
    <div class="root">
        <div>
-|    <pre class="state"> (WaitOperation (acc 0)) </pre>
+|    <pre class="state"> (Finish 0) </pre>
        <div>
            <form onsubmit>
                <input class="form" #value="" oninput> </input>
            </form>
            <button class="enter" onclick> enter </button>
            <div>
                <button class="back" onclick> back </button>
                <button class="calc" onclick> calc </button>
                <button class="empty" onclick> empty </button>
                <button class="invalid" onclick> invalid </button>
                <button class="reset" onclick> reset </button>
                <button class="quit" onclick> quit </button>
            </div>
        </div>
    </div>
</div>
<div> |}]

;;
end)
;;

let%test_module "test with stub" =
  (module struct
    module StubCalculator : Calculator.S = struct
      include Calc.MakeStateMachine (struct
        type num = float [@@deriving sexp, equal]

```

```

    type op = [ `Sub ] [@@deriving sexp, equal]

    let calculate _ b c = Some (b -. c)
end)

let line_to_action ?quit:_ = function
| "number1" -> Num 123.
| "number2" -> Num (-100.)
| "minus" -> Op `Sub
| _ -> assert false
;;
end

module Component = MakeComponent (StubCalculator)
open Bonsai

let%expect_test "form test" =
  (* Arrange *)
  let state_var = Bonsai.Var.create None in
  let update_var =
    Bonsai.Var.create (fun ac -> Effect.return @@ Bonsai.Var.set state_var (Some ac))
  in
  let update = Bonsai.Var.value update_var in
  let handle = Handle.create (Result_spec.vdom Fn.id) (Component.form update) in
  (* Act, Assert: initial state *)
  Handle.show handle;
  print_s [%sexp (Bonsai.Var.get state_var : StubCalculator.action option)];
  [%expect
  {
    <div>
      <form onsubmit>
        <input class="form" #value="" oninput> </input>
      </form>
      <button class="enter" onclick> enter </button>
      <div>
        <button class="back" onclick> back </button>
        <button class="calc" onclick> calc </button>
        <button class="empty" onclick> empty </button>
        <button class="invalid" onclick> invalid </button>
        <button class="reset" onclick> reset </button>
        <button class="quit" onclick> quit </button>
      </div>
    </div>
  }
  ];
  (* Act, Assert: input number1, but do not click *)
  Handle.input_text handle ~get_vdom:Fn.id ~selector:".form" ~text:"number1";
  Handle.show handle;
  print_s [%sexp (Bonsai.Var.get state_var : StubCalculator.action option)];
  [%expect
  {
    <div>
      <form onsubmit>
        <input class="form" #value="number1" oninput> </input>

```

```

    </form>
    <button class="enter" onclick> enter </button>
    <div>
        <button class="back" onclick> back </button>
        <button class="calc" onclick> calc </button>
        <button class="empty" onclick> empty </button>
        <button class="invalid" onclick> invalid </button>
        <button class="reset" onclick> reset </button>
        <button class="quit" onclick> quit </button>
    </div>
</div>
() |}];
(* Act, Assert: click on enter *)
Handle.click_on handle ~get_vdom:Fn.id ~selector:".enter";
Handle.show handle;
print_s [%sexp (Bonsai.Var.get state_var : StubCalculator.action option)];
[%expect
{|
<div>
  <form onsubmit>
    <input class="form" #value="" oninput> </input>
  </form>
  <button class="enter" onclick> enter </button>
  <div>
    <button class="back" onclick> back </button>
    <button class="calc" onclick> calc </button>
    <button class="empty" onclick> empty </button>
    <button class="invalid" onclick> invalid </button>
    <button class="reset" onclick> reset </button>
    <button class="quit" onclick> quit </button>
  </div>
</div>
((Num 123)) |}];
(* Act, Assert: type end enter number2 *)
Handle.input_text handle ~get_vdom:Fn.id ~selector:".form" ~text:"number2";
Handle.click_on handle ~get_vdom:Fn.id ~selector:".enter";
Handle.show handle;
print_s [%sexp (Bonsai.Var.get state_var : StubCalculator.action option)];
[%expect
{|
<div>
  <form onsubmit>
    <input class="form" #value="" oninput> </input>
  </form>
  <button class="enter" onclick> enter </button>
  <div>
    <button class="back" onclick> back </button>
    <button class="calc" onclick> calc </button>
    <button class="empty" onclick> empty </button>
    <button class="invalid" onclick> invalid </button>
    <button class="reset" onclick> reset </button>
    <button class="quit" onclick> quit </button>
  </div>

```

```

    </div>
    ((Num -100)) |});
(* Act, Assert: type end enter minus *)
Handle.input_text handle ~get_vdom:Fn.id ~selector:".form" ~text:"minus";
Handle.click_on handle ~get_vdom:Fn.id ~selector:".enter";
Handle.show handle;
print_s [%sexp (Bonsai.Var.get state_var : StubCalculator.action option)];
[%expect
  {}
  <div>
    <form onsubmit>
      <input class="form" #value="" oninput> </input>
    </form>
    <button class="enter" onclick> enter </button>
    <div>
      <button class="back" onclick> back </button>
      <button class="calc" onclick> calc </button>
      <button class="empty" onclick> empty </button>
      <button class="invalid" onclick> invalid </button>
      <button class="reset" onclick> reset </button>
      <button class="quit" onclick> quit </button>
    </div>
  </div>
  ((Op Sub)) |});
(* Act, Assert: click on back *)
Handle.click_on handle ~get_vdom:Fn.id ~selector:".back";
Handle.show handle;
print_s [%sexp (Bonsai.Var.get state_var : StubCalculator.action option)];
[%expect
  {}
  <div>
    <form onsubmit>
      <input class="form" #value="" oninput> </input>
    </form>
    <button class="enter" onclick> enter </button>
    <div>
      <button class="back" onclick> back </button>
      <button class="calc" onclick> calc </button>
      <button class="empty" onclick> empty </button>
      <button class="invalid" onclick> invalid </button>
      <button class="reset" onclick> reset </button>
      <button class="quit" onclick> quit </button>
    </div>
  </div>
  (Back) |});
(* Act, Assert: click on calc *)
Handle.click_on handle ~get_vdom:Fn.id ~selector:".calc";
Handle.show handle;
print_s [%sexp (Bonsai.Var.get state_var : StubCalculator.action option)];
[%expect
  {}
  <div>
    <form onsubmit>

```

```

        <input class="form" #value="" oninput> </input>
    </form>
    <button class="enter" onclick> enter </button>
    <div>
        <button class="back" onclick> back </button>
        <button class="calc" onclick> calc </button>
        <button class="empty" onclick> empty </button>
        <button class="invalid" onclick> invalid </button>
        <button class="reset" onclick> reset </button>
        <button class="quit" onclick> quit </button>
    </div>
</div>
(Calculate) |}};
(* Act, Assert: click on empty *)
Handle.click_on handle ~get_vdom:Fn.id ~selector:".empty";
Handle.show handle;
print_s [%sexp (Bonsai.Var.get state_var : StubCalculator.action option)];
[%expect
{
    <div>
        <form onsubmit>
            <input class="form" #value="" oninput> </input>
        </form>
        <button class="enter" onclick> enter </button>
        <div>
            <button class="back" onclick> back </button>
            <button class="calc" onclick> calc </button>
            <button class="empty" onclick> empty </button>
            <button class="invalid" onclick> invalid </button>
            <button class="reset" onclick> reset </button>
            <button class="quit" onclick> quit </button>
        </div>
    </div>
    (Empty) |}};
(* Act, Assert: click on invalid *)
Handle.click_on handle ~get_vdom:Fn.id ~selector:".invalid";
Handle.show handle;
print_s [%sexp (Bonsai.Var.get state_var : StubCalculator.action option)];
[%expect
{
    <div>
        <form onsubmit>
            <input class="form" #value="" oninput> </input>
        </form>
        <button class="enter" onclick> enter </button>
        <div>
            <button class="back" onclick> back </button>
            <button class="calc" onclick> calc </button>
            <button class="empty" onclick> empty </button>
            <button class="invalid" onclick> invalid </button>
            <button class="reset" onclick> reset </button>
            <button class="quit" onclick> quit </button>
        </div>
    </div>
}

```

```

    </div>
    ((Invalid "")) |});
(* Act, Assert: click on reset *)
Handle.click_on handle ~get_vdom:Fn.id ~selector:".reset";
Handle.show handle;
print_s [%sexp (Bonsai.Var.get state_var : StubCalculator.action option)];
[%expect
  {
    <div>
      <form onsubmit>
        <input class="form" #value="" oninput> </input>
      </form>
      <button class="enter" onclick> enter </button>
      <div>
        <button class="back" onclick> back </button>
        <button class="calc" onclick> calc </button>
        <button class="empty" onclick> empty </button>
        <button class="invalid" onclick> invalid </button>
        <button class="reset" onclick> reset </button>
        <button class="quit" onclick> quit </button>
      </div>
    </div>
    (Reset) |});
(* Act, Assert: click on quit (should fail because quit triggers reload) *)
Handle.click_on handle ~get_vdom:Fn.id ~selector:".quit";
Handle.show handle;
print_s [%sexp (Bonsai.Var.get state_var : StubCalculator.action option)];
[%expect.unreachable]
[@@expect.uncaught_exn
  {} ("TypeError: Cannot read properties of undefined (reading 'reload')") |}]
;;
end)
;;

```

4 Вывод

В данной работе мы ознакомились с методологией автоматизации функционального тестирования приложения с графическим пользовательским интерфейсом. Исходный код доступен на [GitHub](#).