École Nationale des Sciences Appliquées d'Agadir

Développement Logiciel et Applicatif (DLA) - 2ème Année

TP 4: Tests Fonctionnels avec Cypress

Calculatrice Professionnelle avec Validation Email

Réalisé par :

BOUANANI Abderrahman

Encadré par :

Prof. Aimad QAZDAR

Année Universitaire : 2024-2025

Date : 19 janvier 2026

Table des matières

1 Introduction

Ce document constitue le compte rendu du TP 4, consacré aux tests fonctionnels avec Cypress. L'objectif principal était de créer une application web (calculatrice professionnelle) et d'implémenter une suite de tests end-to-end (E2E) pour valider son comportement. Ce TP met l'accent sur l'automatisation des tests d'interface utilisateur, la génération de données de test avec Faker, et la validation de règles métier complexes (validation d'email corporate).

1.1 Objectifs du TP

- Créer une application web fonctionnelle avec HTML/CSS/JavaScript
- Configurer un environnement de test Cypress
- Implémenter des tests E2E couvrant les cas nominaux et limites
- Utiliser Faker pour générer des données de test réalistes
- Valider des règles métier (validation email @company.com)

1.2 Dépôt GitHub

Le code complet du projet est disponible sur GitHub :
TP4 Cypress - GitHub Repository

2 Architecture du Projet

2.1 Structure des Fichiers

```
TP4_Cypress/
 ui/
     index.html              # Interface utilisateur
     app.js                  # Logique JavaScript
     cypress.config.js       # Configuration Cypress
     package.json            # Dépendances npm
     cypress/
         e2e/
             calculatrice.cy.js  # Suite de tests
```

2.2 Technologies Utilisées

- **Frontend** : HTML5, CSS3, JavaScript (Vanilla)
- **Tests** : Cypress v15.9.0
- **Génération de données** : @faker-js/faker v9.9.0
- **Serveur web** : http-server

3 Code Source

3.1 Interface HTML (index.html)

```
1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="UTF-8" />
5     <title>Calculatrice Professionnelle</title>
6     <style>
7         body { font-family: Arial, sans-serif; text-align: center; margin-top:
    40px; }
```

```
        input, button { font-size: 1.1em; margin: 6px; padding: 6px; }
        #email-status { margin-top: 10px; font-weight: bold; }
    </style>
</head>
<body>
    <h1>Ma Calculatrice Professionnelle</h1>
    <input type="text" id="input-nom-complet" placeholder="Nom complet" size="40
    "/><br/>
    <input type="email" id="input-email" placeholder="Email professionnel (
    @company.com)" /><br/>
    <input type="number" id="input-a" placeholder="Nombre A" />
    <input type="number" id="input-b" placeholder="Nombre B" />

    <button id="btn-add">+</button>
    <button id="btn-sub">-</button>
    <button id="btn-mul">x</button>
    <button id="btn-div">/</button>
    <br/><br/>

    <div>Resultat: <span id="output">?</span></div>
    <div id="email-status"></div>

    <script src="app.js"></script>
</body>
</html>
```

Listing 1 – index.html - Interface de la calculatrice

3.2 Logique JavaScript (app.js)

```
// Fonctions de calcul de base
function add(a, b) { return a + b; }
function sub(a, b) { return a - b; }
function mul(a, b) { return a * b; }
function div(a, b) { return b !== 0 ? a / b : 'Erreur'; }

// Validation : L'email doit finir par @company.com
function isValidCorporateEmail(email) {
    return email && email.endsWith('@company.com');
}

function updateEmailStatus(email) {
    const el = document.getElementById('email-status');
    if (!email) {
        el.textContent = '';
    } else if (isValidCorporateEmail(email)) {
        el.textContent = 'Email valide';
        el.style.color = 'green';
    } else {
        el.textContent = 'Doit se terminer par @company.com';
        el.style.color = 'red';
    }
}

// Regle : Addition bloquee si l'email est invalide
function isCalculationAllowed() {
    const email = document.getElementById('input-email').value;
    return !email || isValidCorporateEmail(email);
}

function displayResult(value) {
    document.getElementById('output').textContent = value;
}
```

```
34
35 // Ecouteur sur l'email
36 document.getElementById('input-email').addEventListener('input', (e) => {
37     updateEmailStatus(e.target.value);
38 });
39
40 // Bouton Addition (avec protection)
41 document.getElementById('btn-add').addEventListener('click', () => {
42     if (!isCalculationAllowed()) {
43         displayResult('Erreur: Email invalide');
44         return;
45     }
46     const a = parseFloat(document.getElementById('input-a').value) || 0;
47     const b = parseFloat(document.getElementById('input-b').value) || 0;
48     displayResult(add(a, b));
49 });
50
51 // Autres boutons (sans protection)
52 ['sub', 'mul', 'div'].forEach(op => {
53     document.getElementById(`btn-${op}`).addEventListener('click', () => {
54         const a = parseFloat(document.getElementById('input-a').value) || 0;
55         const b = parseFloat(document.getElementById('input-b').value) || 0;
56         let result;
57         switch (op) {
58             case 'sub': result = sub(a, b); break;
59             case 'mul': result = mul(a, b); break;
60             case 'div': result = div(a, b); break;
61         }
62         displayResult(result);
63     });
64 });
```

Listing 2 – app.js - Logique métier de la calculatrice

3.3 Suite de Tests Cypress (calculatrice.cy.js)

```
1  import { faker } from '@faker-js/faker';
2
3  describe('Calculatrice - Tests Fonctionnels Complets', () => {
4
5      // Avant chaque test, on visite la page de la calculatrice
6      beforeEach(() => {
7          cy.visit('http://localhost:8080');
8      });
9
10     // --- 1. Verification de l'interface ---
11     it('affiche correctement le titre de la calculatrice', () => {
12         cy.contains('Ma Calculatrice Professionnelle').should('be.visible');
13     });
14
15     // --- 2. Tests des operations mathematiques ---
16     it('additionne 7 et 3 (Operation simple)', () => {
17         cy.get('#input-a').type('7');
18         cy.get('#input-b').type('3');
19         cy.get('#btn-add').click();
20         cy.get('#output').should('have.text', '10');
21     });
22
23     it('soustrait 5 de 12 (Logique TODO completee)', () => {
24         cy.get('#input-a').type('12');
25         cy.get('#input-b').type('5');
26         cy.get('#btn-sub').click();
27         cy.get('#output').should('have.text', '7');
```

```
28      });
29
30      it('divise 12 par 3 (Logique TODO completee)', () => {
31          cy.get('#input-a').type('12');
32          cy.get('#input-b').type('3');
33          cy.get('#btn-div').click();
34          cy.get('#output').should('have.text', '4');
35      });
36
37      it('affiche "Erreur" lors d\'une division par zero', () => {
38          cy.get('#input-a').type('10');
39          cy.get('#input-b').type('0');
40          cy.get('#btn-div').click();
41          cy.get('#output').should('have.text', 'Erreur');
42      });
43
44      // --- 3. Tests Metier Avances (Email Corporate) avec Faker ---
45
46      it('Email valide (@company.com) -> Calcul autorise', () => {
47          // Generation de donnees aleatoires realistes
48          const prenom = faker.person.firstName();
49          const nom = faker.person.lastName();
50          // Force un email avec le bon domaine
51          const emailValide = faker.internet.email({
52              firstName: prenom,
53              lastName: nom,
54              provider: 'company.com'
55          });
56
57          // Interaction
58          cy.get('#input-nom-complet').type(`${prenom} ${nom}`);
59          cy.get('#input-email').type(emailValide);
60
61          // Verification visuelle du statut
62          cy.get('#email-status')
63            .should('contain', 'Email valide')
64            .and('have.css', 'color', 'rgb(0, 128, 0)'); // Vert
65
66          // Test d'une operation (Multiplication)
67          cy.get('#input-a').type('4');
68          cy.get('#input-b').type('2');
69          cy.get('#btn-mul').click();
70
71          // Validation du resultat
72          cy.get('#output').should('have.text', '8');
73      });
74
75      it('Email invalide (Gmail/Yahoo...) -> Blocage de l\'addition', () => {
76          // Generation d'un email quelconque
77          const emailInvalide = faker.internet.email({ provider: 'gmail.com' });
78
79          cy.get('#input-email').type(emailInvalide);
80
81          // Verification immediate du message d'erreur
82          cy.get('#email-status')
83            .should('contain', 'Doit se terminer par @company.com')
84            .and('have.css', 'color', 'rgb(255, 0, 0)'); // Rouge
85
86          // Tentative d'addition
87          cy.get('#input-a').type('5');
88          cy.get('#input-b').type('5');
89          cy.get('#btn-add').click();
90
```

```
91          // Le calcul ne doit PAS se faire
92          cy.get('#output').should('contain', 'Erreur: Email invalide');
93      });
94  });
```

Listing 3 – calculatrice.cy.js - Tests E2E complets

4 Résultats des Tests

4.1 Exécution des Tests

La suite de tests a été exécutée avec succès. Tous les 7 tests sont passés sans erreur.

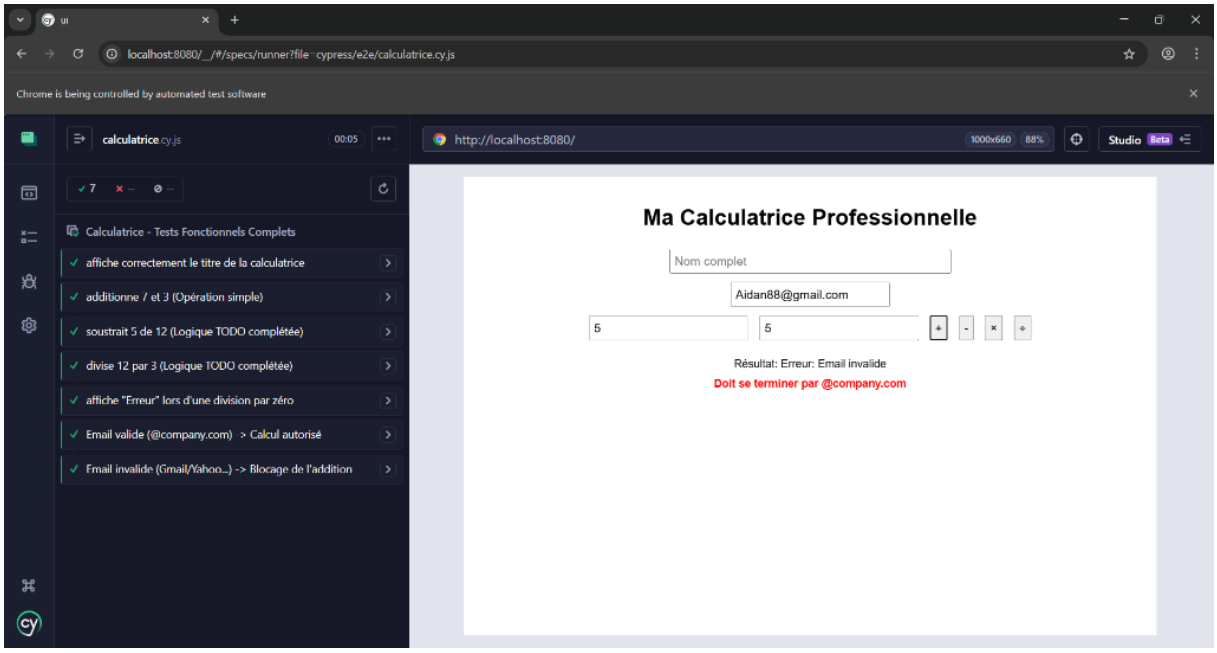

FIGURE 1 – Résultats des tests Cypress - 7/7 tests réussis

4.2 Détail des Tests

1. **Test d'interface** : Vérifie que le titre "Ma Calculatrice Professionnelle" s'affiche correctement
2. **Addition** : Teste $7 + 3 = 10$
3. **Soustraction** : Teste 12 - 5 = 7
4. **Division** : Teste $12 \div 3 = 4$
5. **Division par zéro** : Vérifie que la division par 0 affiche "Erreur"
6. **Email valide** : Teste qu'un email @company.com permet les calculs et affiche un statut vert
7. **Email invalide** : Vérifie que l'addition est bloquée avec un email non-corporate et affiche un message d'erreur rouge

5 Analyse et Points Clés

5.1 Intégration de Faker

L'utilisation de `@faker-js/faker` a permis de générer des données de test réalistes et variées. Plutôt que d'utiliser des valeurs statiques, chaque exécution de test utilise des noms et emails

différents, ce qui augmente la robustesse des tests.

5.2 Validation Métier

La règle métier implémentée (seuls les emails @company.com sont autorisés pour l'addition) démontre comment Cypress peut valider non seulement les calculs, mais aussi les règles de gestion complexes. Les tests vérifient :

- L'affichage visuel du statut (couleur verte/rouge)
- Le contenu textuel du message
- Le blocage effectif de l'opération

5.3 Gestion des Cas Limites

Le test de division par zéro illustre l'importance de tester les cas limites. L'application gère correctement cette situation en affichant "Erreur" plutôt que de crasher ou d'afficher un résultat incorrect.

5.4 Architecture de Tests

L'utilisation de `beforeEach()` garantit que chaque test démarre avec une page fraîche, évitant ainsi les effets de bord entre tests. Cette approche assure l'indépendance et la fiabilité de chaque test.

6 Conclusion

Ce TP a permis de maîtriser les fondamentaux des tests fonctionnels avec Cypress. L'installation et la configuration de l'environnement, bien que parfois délicate (problèmes de timeout résolus), ont abouti à un système de test robuste et complet.
Les 7 tests implémentés couvrent l'ensemble des fonctionnalités de l'application : interface, opérations mathématiques, gestion d'erreurs, et validation métier. L'intégration de Faker démontre comment automatiser la génération de données de test réalistes.
Ce projet illustre l'importance des tests E2E dans le développement web moderne, permettant de valider le comportement de l'application du point de vue de l'utilisateur final.