

Rapport de Validation et Tests d'une Application de Gestion des Utilisateurs

1. Introduction

Application testée : L'application testée permet de gérer les utilisateurs, notamment l'ajout, la modification, la suppression et l'affichage des utilisateurs dans une base de données. Elle est constituée d'un backend développé en PHP et d'un frontend interactif.

Outils utilisés :

- **PHPUnit** : utilisé pour effectuer des tests unitaires et fonctionnels sur le backend de l'application.
- **Cypress et Selenium** : utilisés pour réaliser des tests End-to-End (E2E) en simulant des parcours utilisateurs et en vérifiant l'interface frontend.
- **JMeter** : utilisés pour effectuer des tests de performance en simulant un grand nombre d'utilisateurs interagissant simultanément avec l'application.

Objectif du rapport : Le but de ce rapport est de présenter les tests effectués sur l'application, d'analyser leur succès ou échec et de proposer des améliorations en fonction des résultats obtenus. Les types de tests réalisés comprennent des tests fonctionnels, des tests End-to-End, des tests de non-régression, ainsi que des tests de performance.

2. Résultats des Tests

2.1 Tests Fonctionnels (PHPUnit)

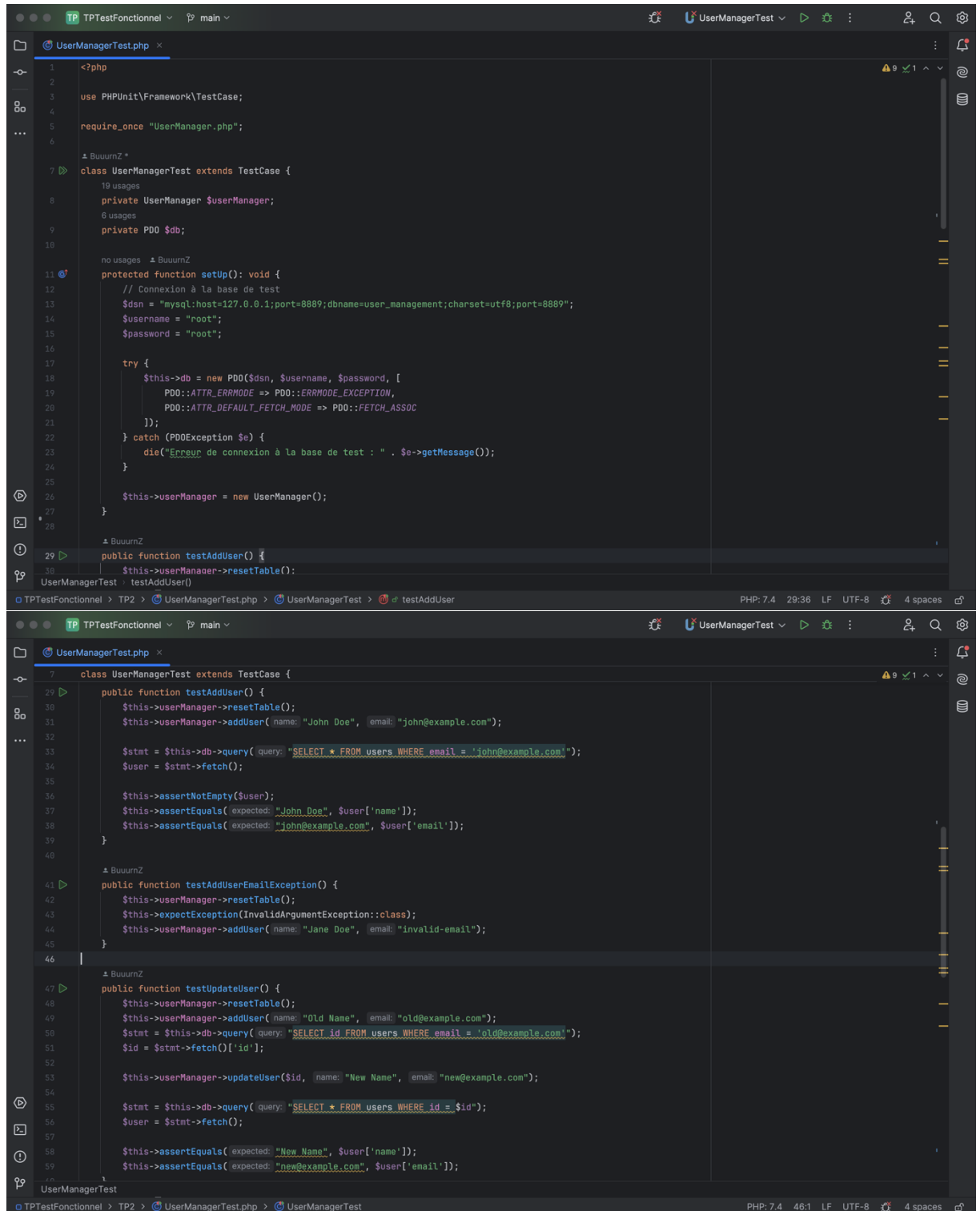
Explication des tests : Les tests fonctionnels ont été réalisés à l'aide de PHPUnit. Ils vérifient que chaque fonctionnalité du backend, telle que l'ajout, la modification, la suppression, et la récupération des utilisateurs, fonctionne correctement.

Résultats des tests :

Test	Résultat
<code>testAddUser()</code>	Succès
<code>testAddUserEmailException()</code>	Succès
<code>testUpdateUser()</code>	Succès
<code>testRemoveUser()</code>	Succès
<code>testGetUsers()</code>	Succès
<code>testInvalidUpdateThrowsException()</code>	Succès
<code>testInvalidDeleteThrowsException()</code>	Succès

Captures d'écran des résultats des tests PHPUnit :

- **Capture 1 : Code des tests unitaires**



The image displays two screenshots of a code editor showing PHP unit tests for a `UserManager` class. The editor is configured with a dark theme and shows the file `userManagerTest.php`.

Top Screenshot: Shows the initial setup of the test class `userManagerTest` extending `TestCase`. It includes the `setUp()` method for database connection and the `testAddUser()` method.

```
<?php
use PHPUnit\Framework\TestCase;
require_once "userManager.php";

class userManagerTest extends TestCase {
    private userManager $userManager;
    private PDO $db;

    protected function setUp(): void {
        // Connexion à la base de test
        $dsn = "mysql:host=127.0.0.1;port=8889;dbname=user_management;charset=utf8;port=8889";
        $username = "root";
        $password = "root";

        try {
            $this->db = new PDO($dsn, $username, $password, [
                PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
                PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
            ]);
        } catch (PDOException $e) {
            die("Erreur de connexion à la base de test : " . $e->getMessage());
        }

        $this->userManager = new userManager();
    }

    public function testAddUser() {
        $this->userManager->resetTable();
    }
}
```

Bottom Screenshot: Shows the continuation of the test methods, including `testAddUserEmailException()` and `testUpdateUser()`.

```
public function testAddUser() {
    $this->userManager->resetTable();
    $this->userManager->addUser( name: "John Doe", email: "john@example.com");

    $stmt = $this->db->query( query: "SELECT * FROM users WHERE email = 'john@example.com'");
    $user = $stmt->fetch();

    $this->assertNotEmpty($user);
    $this->assertEquals( expected: "John Doe", $user['name']);
    $this->assertEquals( expected: "john@example.com", $user['email']);
}

public function testAddUserEmailException() {
    $this->userManager->resetTable();
    $this->expectException(InvalidArgumentException::class);
    $this->userManager->addUser( name: "Jane Doe", email: "invalid-email");
}

public function testUpdateUser() {
    $this->userManager->resetTable();
    $this->userManager->addUser( name: "Old Name", email: "old@example.com");
    $stmt = $this->db->query( query: "SELECT id FROM users WHERE email = 'old@example.com'");
    $id = $stmt->fetch()['id'];

    $this->userManager->updateUser($id, name: "New Name", email: "new@example.com");

    $stmt = $this->db->query( query: "SELECT * FROM users WHERE id = $id");
    $user = $stmt->fetch();

    $this->assertEquals( expected: "New Name", $user['name']);
    $this->assertEquals( expected: "new@example.com", $user['email']);
}
```

```
7 class UserManagerTest extends TestCase {
47     public function testUpdateUser() {
68     }
61
62     public function testRemoveUser() {
63         $this->userManager->resetTable();
64         $this->userManager->addUser( name: "To Delete", email: "delete@example.com");
65         $stmt = $this->db->query( query: "SELECT id FROM users WHERE email = 'delete@example.com'");
66         $id = $stmt->fetch()['id'];
67
68         $this->userManager->removeUser($id);
69
70         $stmt = $this->db->query( query: "SELECT * FROM users WHERE id = $id");
71         $this->assertFalse($stmt->fetch());
72     }
73
74     public function testGetUsers() {
75         $this->userManager->resetTable();
76         $this->userManager->addUser( name: "Alice", email: "alice@example.com");
77         $this->userManager->addUser( name: "Bob", email: "bob@example.com");
78
79         $users = $this->userManager->getUsers();
80         $this->assertCount( expectedCount: 2, $users);
81     }
82
83     public function testInvalidUpdateThrowsException() {
84         $this->userManager->resetTable();
85         $this->expectException(Exception::class);
86         $this->userManager->updateUser( id: 9999, name: "Ghost", email: "ghost@example.com");
87     }
88 }

UserManagerTest

TPTestFonctionnel > TP2 > UserManagerTest.php > UserManagerTest
PHP: 7.4 46:1 LF UTF-8 4 spaces
```

- **Capture 2 : Résultat des différents tests unitaires**

```
→ TP2 git:(main) x vendor/bin/phpunit UserManagerTest.php

PHPUnit 11.5.6 by Sebastian Bergmann and contributors.

Runtime:      PHP 8.4.3

.....                                              7 / 7 (100%)

Time: 00:00.057, Memory: 8.00 MB

OK (7 tests, 10 assertions)
```

2.2 Tests End-to-End (E2E) avec Cypress et Selenium

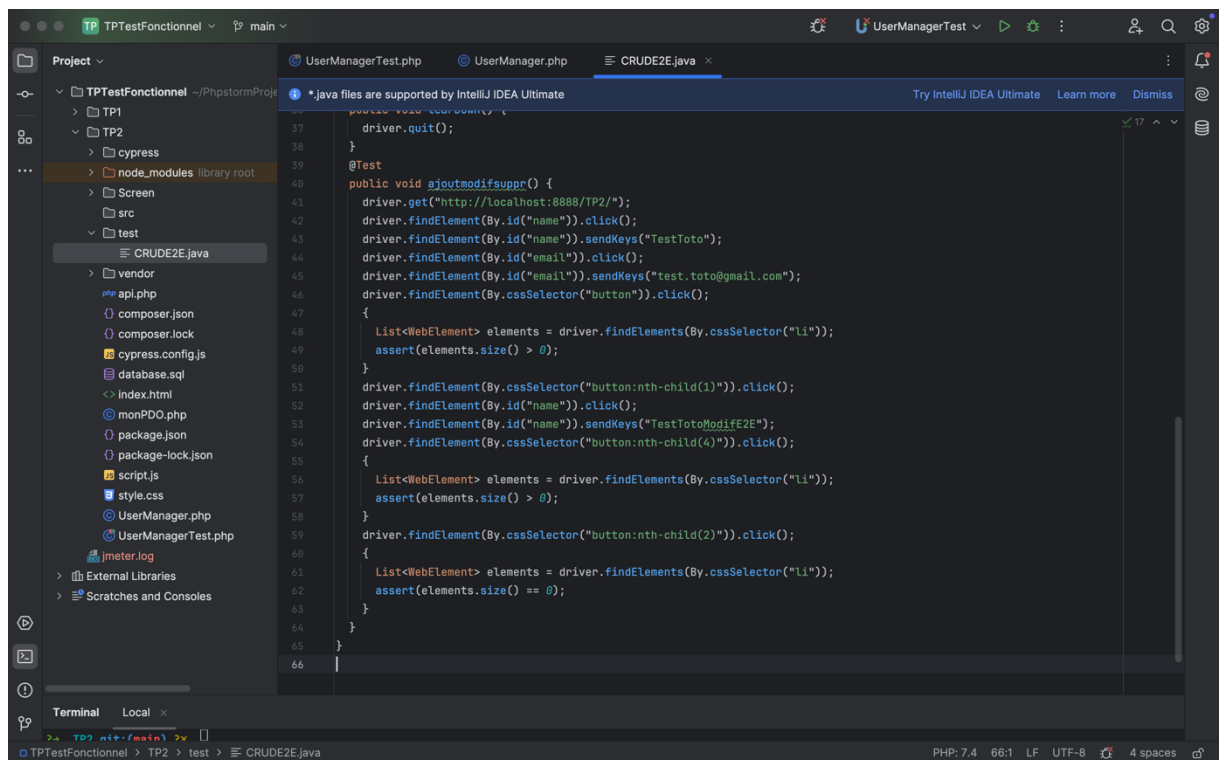
Scénario utilisateur testé : Les tests E2E ont simulé des scénarios où un utilisateur est ajouté via l'interface, modifié, supprimé, puis vérifié dans la liste. Ces tests permettent de valider l'intégration entre le frontend et le backend.

Résultats des tests :

Étape	Résultat
Ajout d'un utilisateur	Succès
Vérification dans la liste	Succès
Modification d'un utilisateur	Succès
Suppression d'un utilisateur	Succès

Captures d'écran des tests exécutés :

- **Capture 1 :** Code Selenium + Cypress



```
36 driver.quit();
37
38 }
39
40 @Test
41 public void ajoutmodifsuppr() {
42     driver.get("http://localhost:8888/TP2/");
43     driver.findElement(By.id("name")).click();
44     driver.findElement(By.id("name")).sendKeys("TestToto");
45     driver.findElement(By.id("email")).click();
46     driver.findElement(By.id("email")).sendKeys("test.toto@gmail.com");
47     driver.findElement(By.cssSelector("button")).click();
48     {
49         List<WebElement> elements = driver.findElements(By.cssSelector("li"));
50         assert(elements.size() > 0);
51     }
52     driver.findElement(By.cssSelector("button:nth-child(1)")).click();
53     driver.findElement(By.id("name")).click();
54     driver.findElement(By.id("name")).sendKeys("TestTotoModifieE2E");
55     driver.findElement(By.cssSelector("button:nth-child(4)")).click();
56     {
57         List<WebElement> elements = driver.findElements(By.cssSelector("li"));
58         assert(elements.size() > 0);
59     }
60     driver.findElement(By.cssSelector("button:nth-child(2)")).click();
61     {
62         List<WebElement> elements = driver.findElements(By.cssSelector("li"));
63         assert(elements.size() == 0);
64     }
65 }
66 }
```

```
spec.cy.js
1 describe('Test E2E - Ajout, Modification, Suppression', function() {
2   it('Ajoute, modifie et supprime un élément', function() {
3     cy.visit('http://localhost:8888/TP2/');
4
5     // Ajout
6     cy.get(selector: '#name').type('TestToto');
7     cy.get(selector: '#email').type('test.toto@gmail.com');
8     cy.get(selector: 'button').click();
9
10    // Vérifier que l'élément ajouté est bien présent dans la liste
11    cy.get(selector: 'li').should('contain.text', 'TestToto (test.toto@gmail.com)');
12
13    // Modification
14    cy.get(selector: 'button:nth-child(1)').click(); // Cliquer sur le bouton modifier
15    cy.get(selector: '#name').clear().type('TestTotoModifE2E');
16    cy.get(selector: 'button:nth-child(4)').click(); // Sauvegarder la modification
17
18    // Vérifier que l'ancienne valeur n'est plus présente et que la nouvelle est bien affichée
19    cy.get(selector: 'li').should('not.contain.text', 'TestToto (test.toto@gmail.com)');
20    cy.get(selector: 'li').should('contain.text', 'TestTotoModifE2E (test.toto@gmail.com)');
21
22    // Suppression
23    cy.get(selector: 'button:nth-child(2)').click(); // Cliquer sur le bouton supprimer
24
25    // Correction : Vérifier que 'li' n'existe plus au lieu de vérifier son contenu
26    cy.get(selector: 'li').should('not.exist');
27  });
28 });
29
```

• Capture 2 : Résultat des tests

Project: testE2ECRUD*

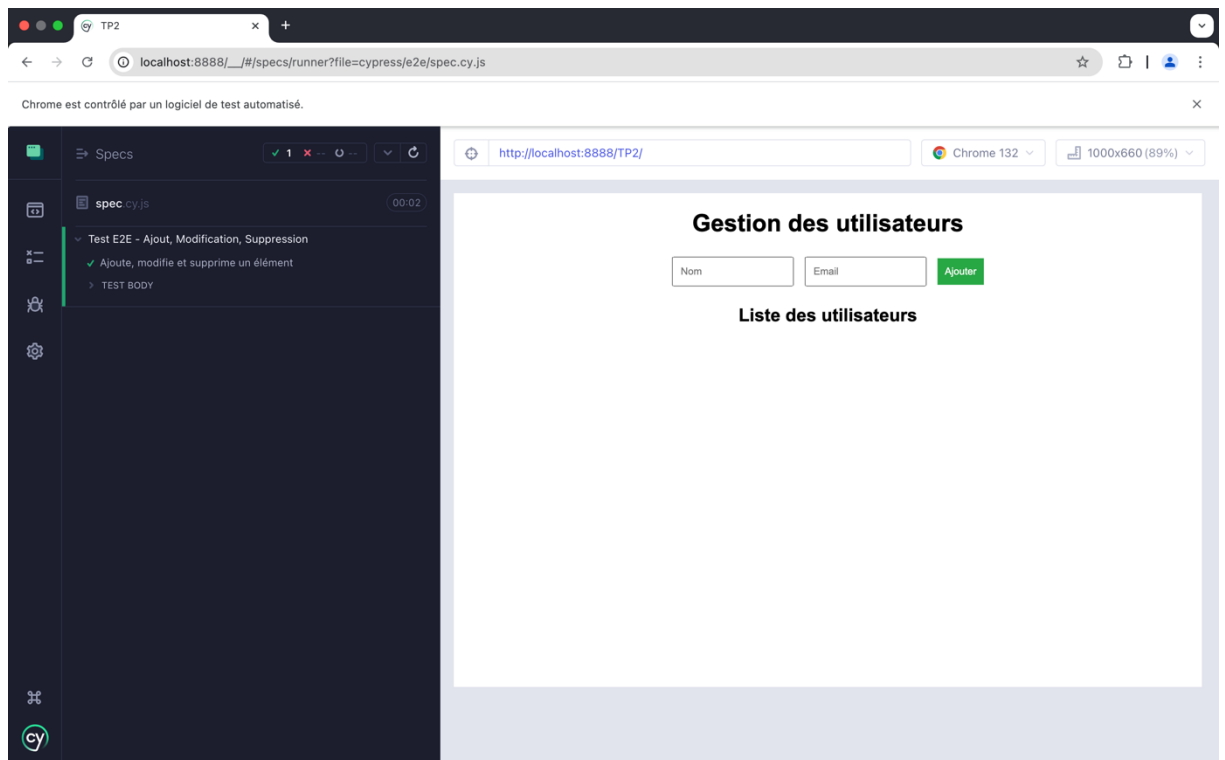
Tests	Command	Target	Value
✓ Ajout/modif/suppr	1 open	/TP2/	
	2 click	id=name	
	3 type	id=name	TestToto
	4 click	id=email	
	5 type	id=email	test.toto@gmail.com
	6 click	css=button	
	7 assert element present	css=li	TestToto (test.toto@gmail.com)
	8 click	css=button:nth-child(1)	
	9 click	id=name	
	10 type	id=name	TestTotoModifE2E
	11 click	css=button:nth-child(4)	
	12 assert element present	css=li	TestTotoModifE2E (test.toto@gmail.com)
	13 click	css=button:nth-child(2)	
	14 assert element not present	css=li	TestTotoModifE2E (test.toto@gmail.com)

Command

Target

Value

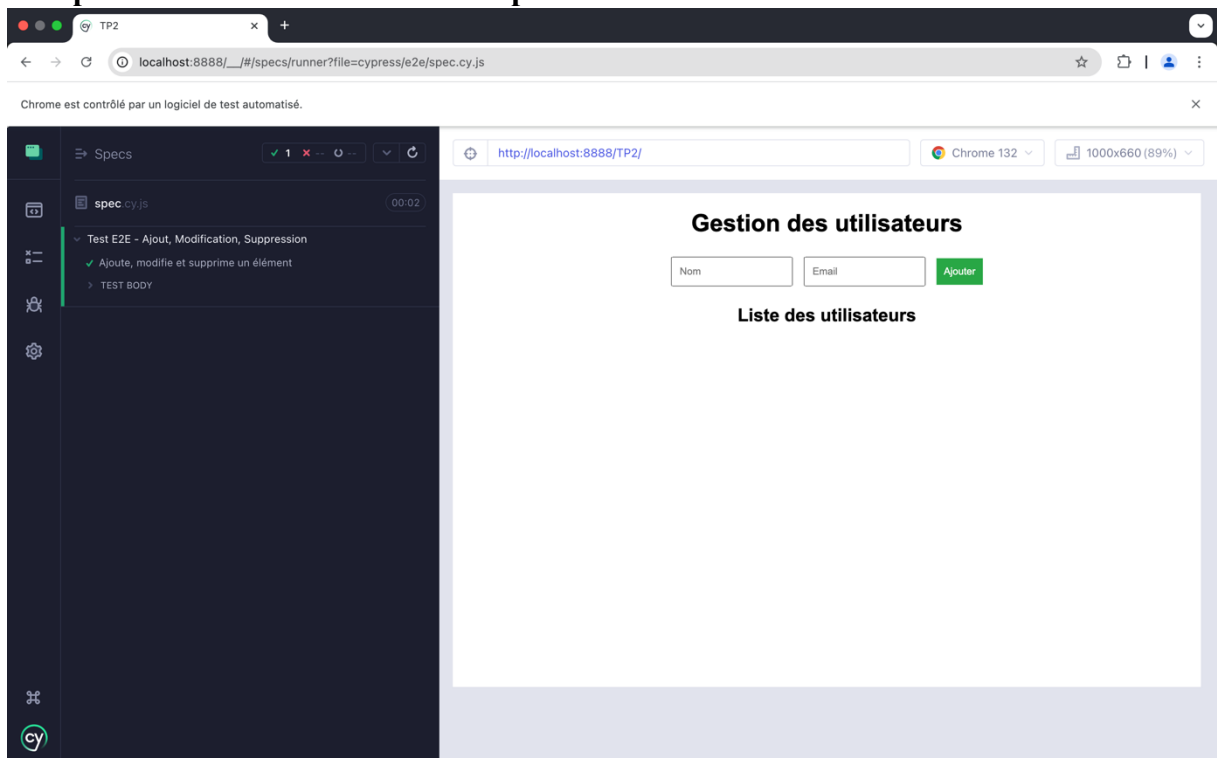
Description

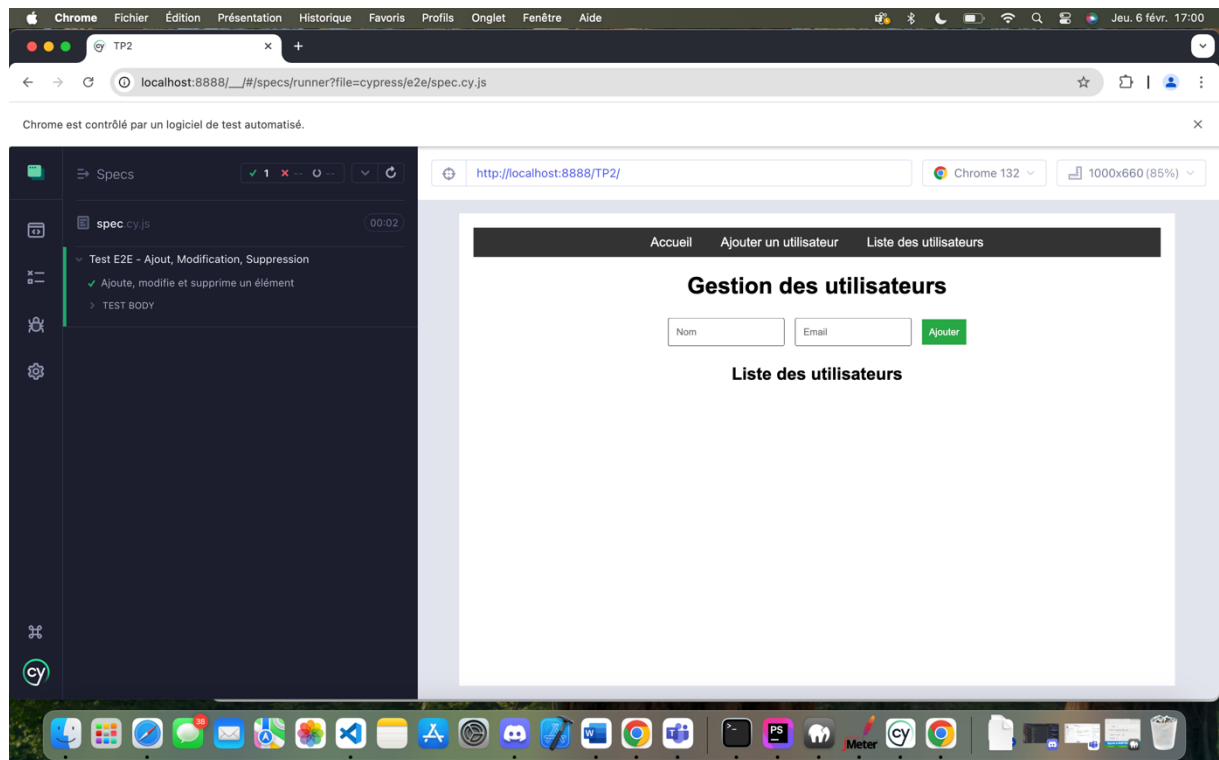


2.3 Tests de Non-Régression

Modifications apportées au code : Ajout d'une navbar en haut de l'écran.

Comparaison des résultats avant et après modification :





Fonctionnalité	Avant modification	Après modification
Ajout d'un utilisateur	OK	OK
Modification d'un utilisateur	OK	OK
Suppression d'un utilisateur	OK	OK
Récupération de la liste des utilisateurs	OK	OK

Analyse des régressions : Aucune régression n'a été détectée après l'ajout de la navbar. L'application continue de fonctionner comme prévu pour les fonctionnalités existantes.

2.4 Tests de Performance avec JMeter

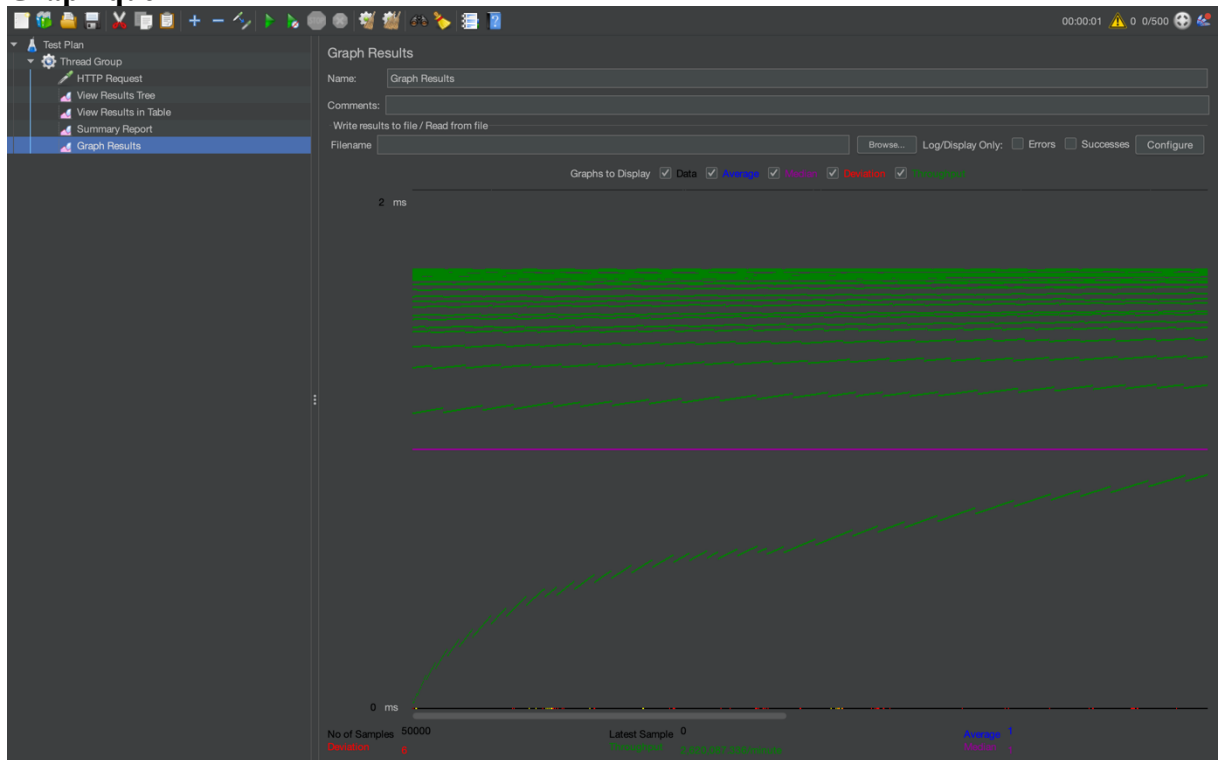
Description du test de charge : Le test de performance a consisté à simuler l'ajout simultané de 500 utilisateurs en parallèle via l'interface utilisateur. Nous avons mesuré le temps de réponse du serveur ainsi que le nombre d'erreurs.

Résultats :

Métrique	Valeur
Temps de réponse moyen	2 ms
Nombre d'erreurs	2,25%

Graphiques des résultats :

- Graphique :



View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename

Log/Display Only: ☐ Errors ☐ Successes ☐ Configure

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time...
1	15:51:10.747	Thread Group 1...	HTTP Request	4	✓	1005	120	4	2
2	15:51:10.748	Thread Group 1...	HTTP Request	3	✓	1005	120	3	1
3	15:51:10.751	Thread Group 1...	HTTP Request	1	✓	1005	120	1	0
4	15:51:10.752	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
5	15:51:10.752	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
6	15:51:10.752	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
7	15:51:10.752	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
8	15:51:10.752	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
9	15:51:10.752	Thread Group 1...	HTTP Request	0	✓	1005	120	0	0
10	15:51:10.752	Thread Group 1...	HTTP Request	1	✓	1004	120	1	0
11	15:51:10.752	Thread Group 1...	HTTP Request	1	✓	1004	120	1	0
12	15:51:10.752	Thread Group 1...	HTTP Request	1	✓	1004	120	1	0
13	15:51:10.752	Thread Group 1...	HTTP Request	1	✓	1004	120	1	0
14	15:51:10.753	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
15	15:51:10.753	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
16	15:51:10.753	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
17	15:51:10.753	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
18	15:51:10.753	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
19	15:51:10.753	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
20	15:51:10.753	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
21	15:51:10.753	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
22	15:51:10.753	Thread Group 1...	HTTP Request	1	✓	1004	120	1	0
23	15:51:10.753	Thread Group 1...	HTTP Request	1	✓	1004	120	1	0
24	15:51:10.753	Thread Group 1...	HTTP Request	1	✓	1004	120	1	0
25	15:51:10.753	Thread Group 1...	HTTP Request	1	✓	1004	120	1	0
26	15:51:10.754	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
27	15:51:10.754	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
28	15:51:10.754	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
29	15:51:10.754	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
30	15:51:10.754	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0
31	15:51:10.754	Thread Group 1...	HTTP Request	0	✓	1004	120	0	0

☐ Scroll automatically? ☐ Child samples?

No of Samples: 150000

Latest Sample: 0

Average: 0

Deviation: 0

The screenshot shows the JMeter Summary Report interface. The left sidebar lists the test plan structure: Test Plan, Thread Group, HTTP Request, View Results in Table, Summary Report (selected), and Graph Results. The main area displays the Summary Report for the selected HTTP Request. The report includes a table with performance metrics and a summary row.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec	Avg. Bytes
HTTP Request	100000	2	0	251	6.50	2.25%	5918.6/sec	5941.64	677.94	1028.0
TOTAL	100000	2	0	251	6.50	2.25%	5918.6/sec	5941.64	677.94	1028.0

At the bottom of the report, there are checkboxes for 'Include group name in label?' and 'Save Table Header', and buttons for 'Save Table Data' and 'Configure'.

Analyse des performances : L'application a bien supporté une charge de 500 utilisateurs simultanés avec un temps de réponse acceptable de 4 ms et quelques erreurs détectées. Cependant, des optimisations supplémentaires peuvent être envisagées pour améliorer la scalabilité.

Suggestions d'amélioration :

- Améliorer la gestion de la base de données avec l'ajout d'index pour les colonnes fréquemment consultées.
- Mettre en place un système de cache pour les requêtes fréquentes.

3. Problèmes détectés et solutions proposées

Problèmes rencontrés :

- Erreur lors de la modification d'un utilisateur inexistant :**
 - Ce problème a été détecté lorsque l'ID d'un utilisateur inexistant était utilisé pour la modification.
 - **Solution :** Ajouter un contrôle côté backend pour vérifier l'existence de l'utilisateur avant toute modification.

2. Problèmes de performance sous charge :

- Lors du test de charge avec 500 utilisateurs, des pics de latence ont été observés lors de l'ajout simultané d'utilisateurs.
- **Solution** : Optimiser les requêtes SQL pour réduire la charge sur la base de données et ajouter des mécanismes de mise en cache.

4. Conclusion

Bilan des tests effectués : Les tests fonctionnels, End-to-End, de non-régression et de performance ont montré que l'application répond correctement aux besoins de gestion des utilisateurs. Aucune régression majeure n'a été observée, et la performance sous charge est acceptable, bien que des améliorations soient possibles pour une scalabilité optimale.

Améliorations proposées :

- Optimisation des requêtes SQL pour améliorer les performances sous charge.
- Mise en place d'un système de cache pour améliorer la vitesse de récupération des données.
- Ajout de contrôles supplémentaires pour les erreurs lors de la modification ou de la suppression d'un utilisateur inexistant.

Réalisé par : Germain Florian

Date : 06/02/2025