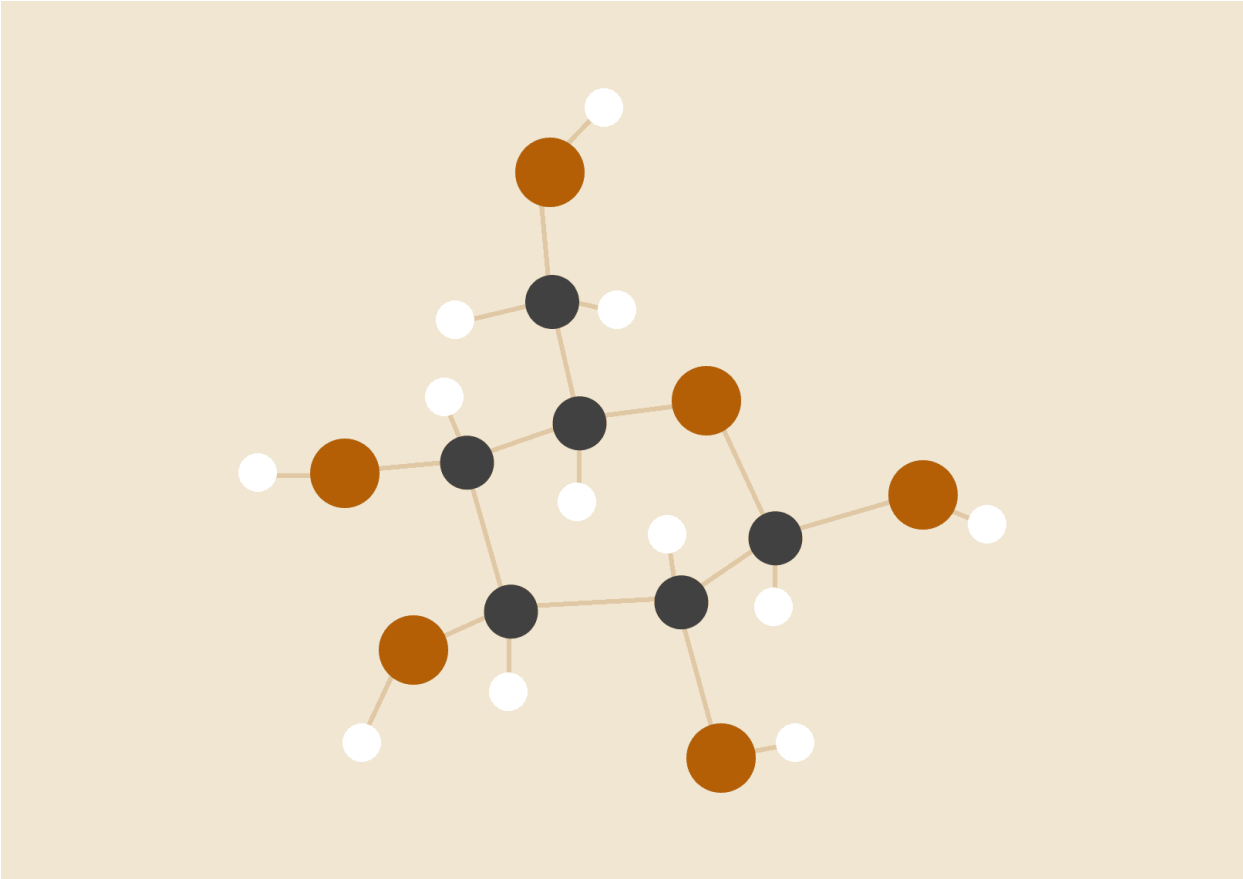
RAPPORT DE VALIDATION ET TESTS D'UNE APPLICATION DE GESTION DES UTILISATEURS



Georgy Guei

TESTS FONCTIONNELS, E2E

INTRODUCTION

Ce rapport documente les différentes étapes de validation et de tests d'une application de gestion des utilisateurs développée en PHP, elle est conçue selon une architecture MVC (Modèle-Vue-Contrôleur) et est conteneurisée à l'aide de Docker pour faciliter son déploiement. L'objectif du rapport est de décrire les tests réalisés, les résultats obtenus, et d'identifier les éventuels problèmes rencontrés ainsi que les solutions proposées. Les tests ont été effectués en utilisant différents outils, notamment PHPUnit pour les tests fonctionnels, Selenium pour les tests End-to-End (E2E), et JMeter pour les tests de performance.

Documentation des Tests Fonctionnels

```php
<?php

require_once __DIR__ . '/../src/models/UserManager.php';

use PHPUnit\Framework\TestCase;

class UserManagerTest extends TestCase
{
  private $userManager;

  protected function setUp(): void
  {
    $this->userManager = new UserManage ();
    $this->truncateUsersTable();
  }

  private function truncateUsersTable(): void
  {
    $stmt = $this->userManager->getDB()->prepare("TRUNCATE TABLE users");
    $stmt->execute();
  }

  public function testAddUser()
  {
    $this->userManager->addUser('John Doe', 'john.doe@example.com');

    $users = $this->userManager->getUsers();
    $this->assertCount(1, $users);
    $this->assertEquals('John Doe', $users[0]['name']);
    $this->assertEquals('john.doe@example.com', $users[0]['email']);
  }

  public function testAddUserEmailException()
  {
    $this->expectException(InvalidArgumentException::class);

    $this->userManager->addUser('Jane Doe', 'invalid-email');
  }

  public function testUpdateUser()
  {
    $this->userManager->addUser('John Doe', 'john.doe@example.com');
    $users = $this->userManager->getUsers();
    $user = $users[0];

    $this->userManager->updateUser($user['id'], 'Johnathan Doe', 'johnathan.doe@example.com');

    $updatedUser = $this->userManager->getUser($user['id']);
    $this->assertEquals('Johnathan Doe', $updatedUser['name']);
    $this->assertEquals('johnathan.doe@example.com', $updatedUser['email']);
  }

  public function testRemoveUser()
  {
    $this->userManager->addUser('John Doe', 'john.doe@example.com');
    $users = $this->userManager->getUsers();
    $user = $users[0];

    $this->userManager->removeUser($user['id']);
    $this->expectException(Exception::class);
    $this->userManager->getUser($user['id']);
  }

  public function testGetUsers()
  {
    $this->userManager->addUser('John Doe', 'john.doe@example.com');
    $this->userManager->addUser('Jane Doe', 'jane.doe@example.com');

    $users = $this->userManager->getUsers();
    $this->assertCount(2, $users);
  }

  public function testInvalidUpdateThrowsException()
  {
    $this->expectException(Exception::class);

    $this->userManager->updateUser(999, 'Non Existent', 'non.existent@example.com');
  }

  public function testInvalidDeleteThrowsException()
  {
    $this->expectException(Exception::class);

    $this->userManager->removeUser(999); // ID inexistant
  }
}

```bash
Georgys-MacBook-Air:php georgyguei$ composer test
Composer could not detect the root package (georgygguei/php) version, defaulting to '1.0.0'. See https://getcomposer.org/root-version
> phpunit
PHPUnit 9.6.22 by Sebastian Bergmann and contributors.

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

Time: 00:00.026, Memory: 6.00 MB

OK (7 tests, 10 assertions)
Georgys-MacBook-Air:php georgygguei$
```
```

1. **testAddUser()**

Objectif : Vérifier que l'ajout d'un utilisateur à la base de données fonctionne correctement.

Description : Ce test ajoute un utilisateur "John Doe" avec l'email "john.doe@example.com" à la base de données en utilisant la méthode `addUser()`. Ensuite, il récupère la liste des utilisateurs à l'aide de la méthode `getUsers()` et vérifie que la liste contient un utilisateur avec les informations correctes.

Résultat : Succès. L'utilisateur "John Doe" a été ajouté correctement à la base de données et les informations récupérées sont conformes aux attentes.

2. **testAddUserEmailException()**

Objectif : Vérifier que l'ajout d'un utilisateur avec un email invalide génère une exception.

Description : Ce test tente d'ajouter un utilisateur "Jane Doe" avec un email invalide "invalid-email" en utilisant la méthode addUser(). Il s'attend à ce qu'une InvalidArgumentException soit lancée.

Résultat : Succès. Une InvalidArgumentException a été lancée comme attendu, indiquant que l'email invalide a été correctement détecté et géré.

3. **testUpdateUser()**

Objectif : Vérifier que la mise à jour des informations d'un utilisateur dans la base de données fonctionne correctement.

Description : Ce test ajoute un utilisateur "John Doe" avec l'email "john.doe@example.com" à la base de données, puis met à jour cet utilisateur en "Johnathan Doe" avec l'email "johnathan.doe@example.com" en utilisant la méthode updateUser(). Ensuite, il récupère l'utilisateur mis à jour à l'aide de la méthode getUser() et vérifie que les informations ont été mises à jour correctement.

Résultat : Succès. Les informations de l'utilisateur ont été mises à jour correctement dans la base de données et les informations récupérées après la mise à jour sont conformes aux attentes.

4. **testRemoveUser()**

Objectif : Vérifier que la suppression d'un utilisateur de la base de données fonctionne correctement.

Description : Ce test ajoute un utilisateur "John Doe" avec l'email "john.doe@example.com" à la base de données, puis supprime cet utilisateur en utilisant la méthode removeUser(). Ensuite, il tente de récupérer l'utilisateur supprimé à l'aide de la méthode getUser() et s'attend à ce qu'une exception soit lancée, indiquant que l'utilisateur n'existe plus.

Résultat : Succès. L'utilisateur "John Doe" a été supprimé correctement de la base de données et une exception a été lancée comme attendu lors de la tentative de récupération de l'utilisateur supprimé.

5. **testGetUsers()**

Objectif : Vérifier que la récupération de la liste des utilisateurs fonctionne correctement.

Description : Ce test ajoute deux utilisateurs ("John Doe" et "Jane Doe") à la base de données en utilisant la méthode addUser(), puis récupère la liste des utilisateurs à l'aide de la méthode getUsers(). Il vérifie que la liste contient les deux utilisateurs ajoutés.

Résultat : Succès. La liste des utilisateurs récupérée contient correctement les deux utilisateurs ajoutés, confirmant que la récupération de la liste des utilisateurs fonctionne comme prévu.

6. **testInvalidUpdateThrowsException()**

Objectif : Vérifier qu'une tentative de mise à jour d'un utilisateur inexistant génère une exception.

Description : Ce test tente de mettre à jour un utilisateur inexistant avec l'ID 999 en utilisant la méthode updateUser(). Il s'attend à ce qu'une exception soit lancée.

Résultat : Succès. Une exception a été lancée comme attendu lors de la tentative de mise à jour de l'utilisateur inexistant, indiquant que le système gère correctement les tentatives de mise à jour d'utilisateurs non existants.

7. **testInvalidDeleteThrowsException()**

Objectif : Vérifier qu'une tentative de suppression d'un utilisateur inexistant génère une exception.

Description : Ce test tente de supprimer un utilisateur inexistant avec l'ID 999 en utilisant la méthode removeUser(). Il s'attend à ce qu'une exception soit lancée.

Résultat : Succès. Une exception a été lancée comme attendu lors de la tentative de suppression de l'utilisateur inexistant, indiquant que le système gère correctement les tentatives de suppression d'utilisateurs non existants.

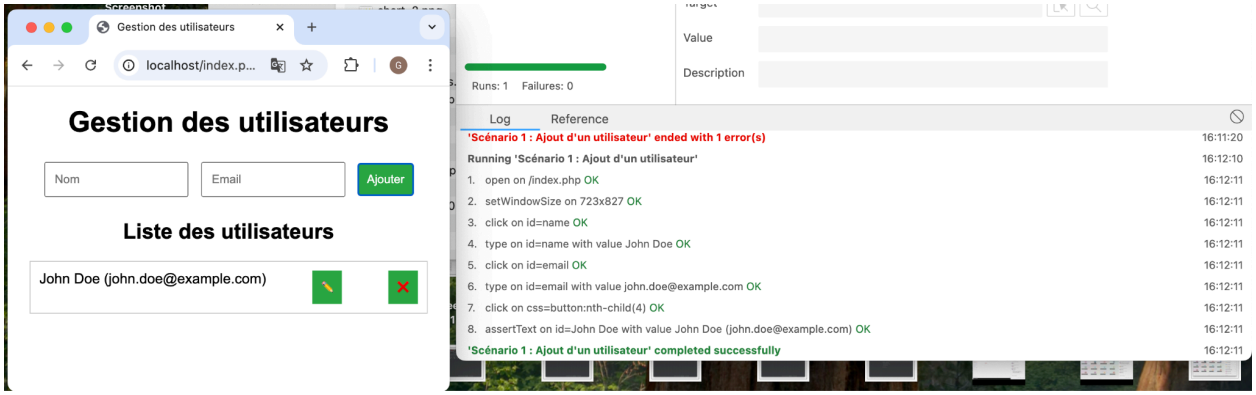
Conclusion

Les tests fonctionnels ont permis de valider les principales fonctionnalités du backend de l'application de gestion d'utilisateurs. Chaque test a réussi, confirmant que les opérations d'ajout, de mise à jour, de suppression et de récupération des utilisateurs fonctionnent correctement. Les exceptions sont gérées comme prévu pour les cas d'emails invalides et les utilisateurs inexistants.

Scénarios de Tests End-to-End (E2E)

Voici des explications détaillées pour chaque scénario de test End-to-End (E2E) utilisant SeleniumIDE. Chaque scénario est décrit avec les étapes précises à réaliser :

Scénario : Ajout d'un utilisateur



1. Ouvrir l'application

Objectif : Accéder à l'application pour commencer l'ajout d'un utilisateur.

Étape : Ouvrez le navigateur et accédez à l'URL de votre application (http://localhost/index.php).

2. Ajouter un utilisateur

Objectif : Saisir les informations d'un nouvel utilisateur et soumettre le formulaire.

Étapes :

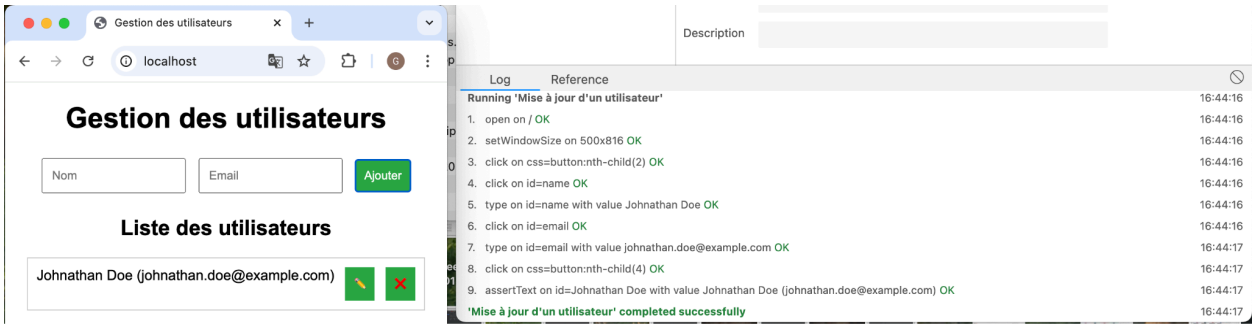
- Cliquez sur le champ de saisie Nom et entrez "John Doe".
- Cliquez sur le champ de saisie Email et entrez "john.doe@example.com".
- Cliquez sur le bouton Ajouter.

3. Vérifier l'ajout de l'utilisateur

Objectif : S'assurer que l'utilisateur ajouté apparaît dans la liste des utilisateurs.

Étape : Vérifiez que l'utilisateur "John Doe" avec l'email "john.doe@example.com" est affiché dans la liste des utilisateurs.

Scénario 2 : Mise à jour d'un utilisateur



1. Sélectionner l'utilisateur à mettre à jour

Objectif : Choisir l'utilisateur à mettre à jour parmi la liste des utilisateurs.

Étape : Cliquez sur le bouton `Modifier` (ou `Edit`) associé à l'utilisateur "John Doe".

2. Mettre à jour les informations de l'utilisateur

Objectif : Saisir les nouvelles informations et soumettre les modifications.

Étapes :

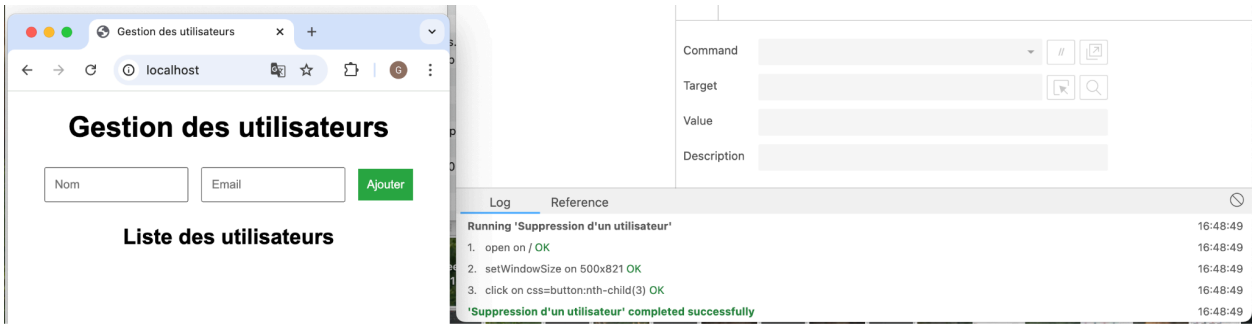
- Cliquez sur le champ de saisie `Nom` et entrez "Johnathan Doe".
- Cliquez sur le champ de saisie `Email` et entrez "johnathan.doe@example.com".
- Cliquez sur le bouton `Mettre à jour` (ou `Update`).

3. Vérifier la mise à jour de l'utilisateur

Objectif : S'assurer que les modifications de l'utilisateur sont correctement enregistrées et affichées.

Étape : Vérifiez que l'utilisateur "Johnathan Doe" avec l'email "johnathan.doe@example.com" est affiché dans la liste des utilisateurs.

Scénario 3 : Suppression d'un utilisateur



1. Sélectionner l'utilisateur à supprimer

Objectif : Choisir l'utilisateur à supprimer parmi la liste des utilisateurs.

Étape : Cliquez sur le bouton `Supprimer` (ou `Delete`) associé à l'utilisateur "Johnathan Doe".

2. Confirmer la suppression

Objectif : Confirmer l'action de suppression si nécessaire.

Étape : Confirmez l'action de suppression si une fenêtre de confirmation apparaît.

3. Vérifier la suppression de l'utilisateur

Objectif : S'assurer que l'utilisateur supprimé n'apparaît plus dans la liste des utilisateurs.

Étape : Vérifiez que l'utilisateur "Johnathan Doe" n'est plus affiché dans la liste des utilisateurs.

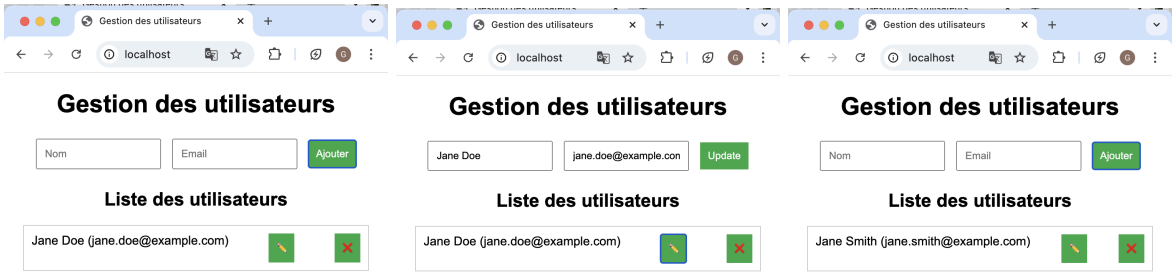
Conclusion

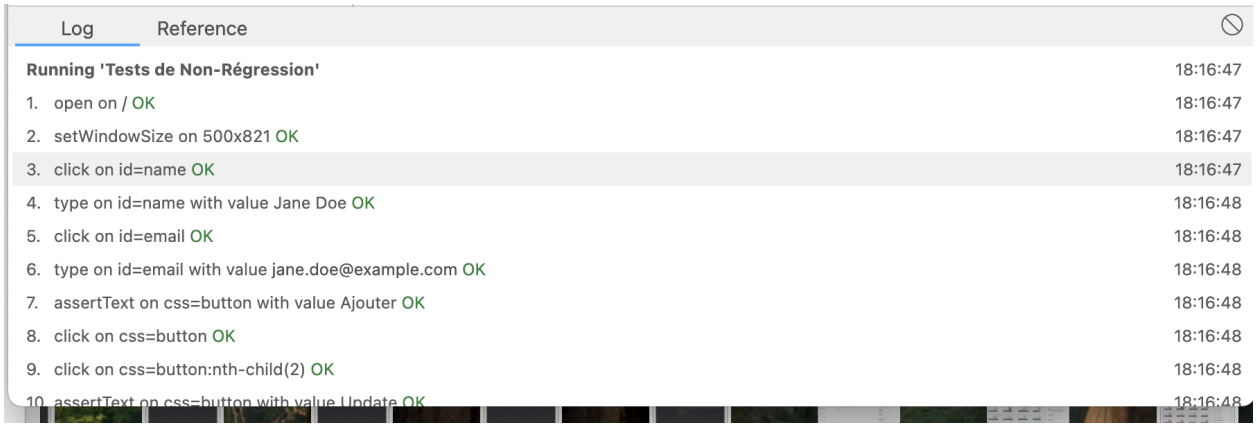
Les scénarios de tests End-to-End (E2E) avec SeleniumIDE permettent de valider les principales interactions utilisateur avec l'application de gestion d'utilisateurs. Chaque scénario est conçu pour tester une fonctionnalité clé de l'application : ajout, mise à jour, suppression et affichage des utilisateurs. En suivant ces scénarios, vous pouvez vous assurer que l'application fonctionne correctement du point de vue de l'utilisateur final. N'hésitez pas à enregistrer ces actions dans SeleniumIDE pour automatiser les tests et garantir la fiabilité continue de l'application. 🚀

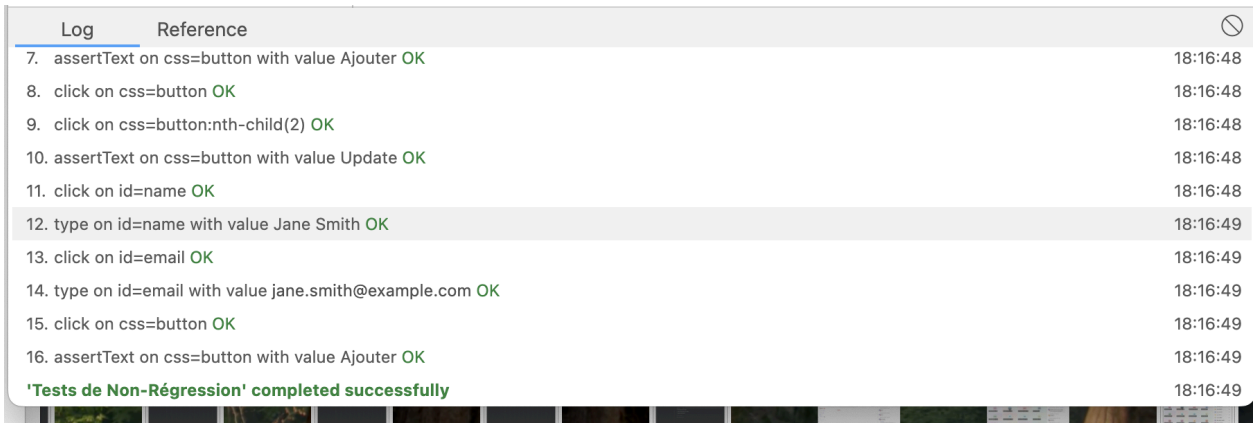
Tests de Non-Régression

Scénario : Mise à jour des informations d'un utilisateur

Objectif : Vérifier que la fonctionnalité de mise à jour des informations d'un utilisateur fonctionne correctement et que le texte du bouton "Ajouter" change comme attendu.







1. **Ouvrir l'application**

Étape : Ouvrez le navigateur et accédez à l'URL de votre application (par exemple, http://localhost).

2. **Ajouter un utilisateur (si nécessaire)**

Étapes :

- Cliquez sur le champ de saisie Nom et entrez "Jane Doe".
- Cliquez sur le champ de saisie Email et entrez "jane.doe@example.com".
- Cliquez sur le bouton Ajouter.

3. **Sélectionner l'utilisateur à mettre à jour**

Étape : Cliquez sur le bouton Modifier (ou Edit) associé à l'utilisateur "Jane Doe".

4. **Vérifier le changement de texte du bouton "Submit"**

Étape : Vérifiez que le texte du bouton "Submit" passe de "Ajouter" à "Update".

5. **Mettre à jour les informations de l'utilisateur**

Étapes :

- Cliquez sur le champ de saisie Nom et entrez "Jane Smith".
- Cliquez sur le champ de saisie Email et entrez "jane.smith@example.com".
- Cliquez sur le bouton Update (précédemment "Submit").

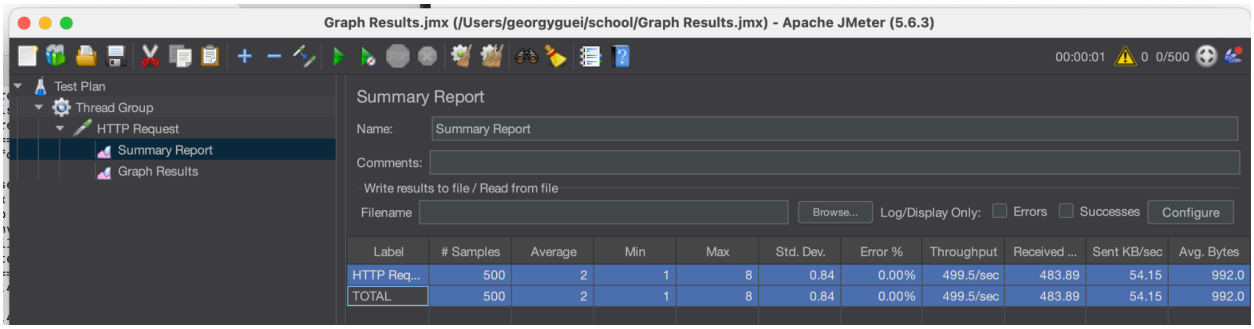
6. **Vérifier la mise à jour des informations de l'utilisateur**

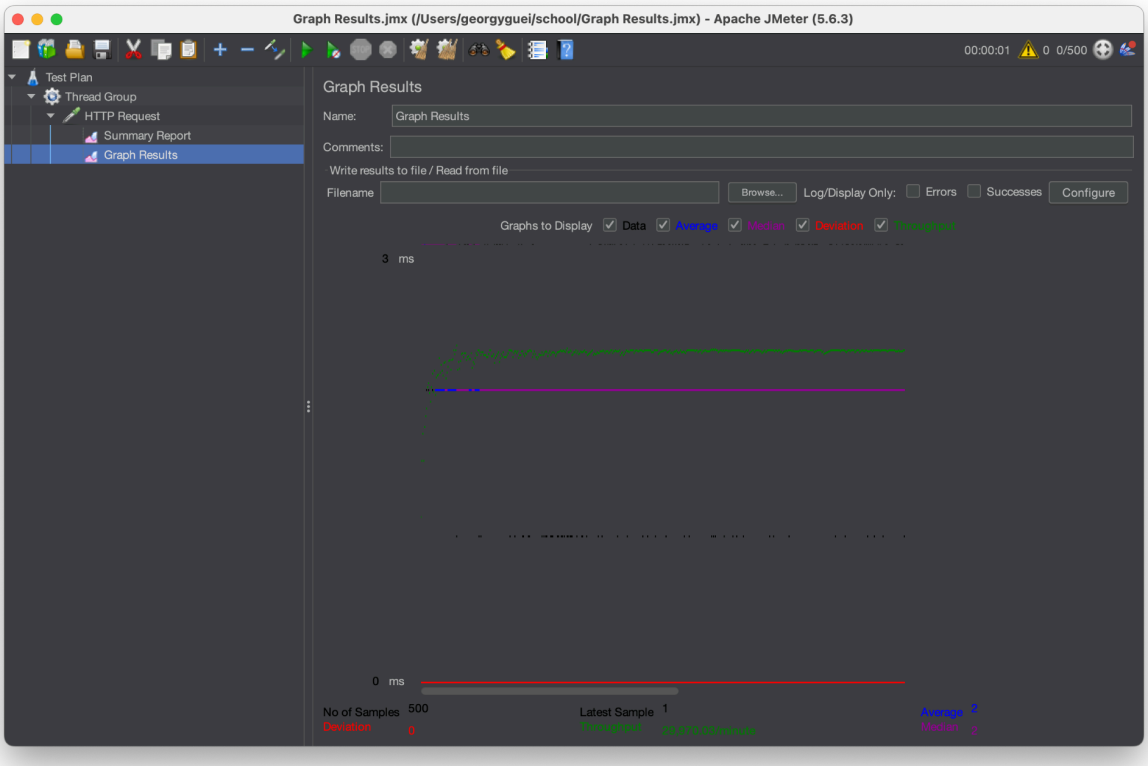
Étape : Vérifiez que les informations de l'utilisateur sont mises à jour correctement et que le texte du bouton "Submit" redevient "Ajouter" après la mise à jour.

7. **Vérifier que la fonctionnalité d'ajout fonctionne toujours**

Étape : Ajoutez un nouvel utilisateur (par exemple, "John Doe" avec l'email "john.doe@example.com") pour s'assurer que la fonctionnalité d'ajout fonctionne toujours correctement.

Tests de Performance avec JMeter



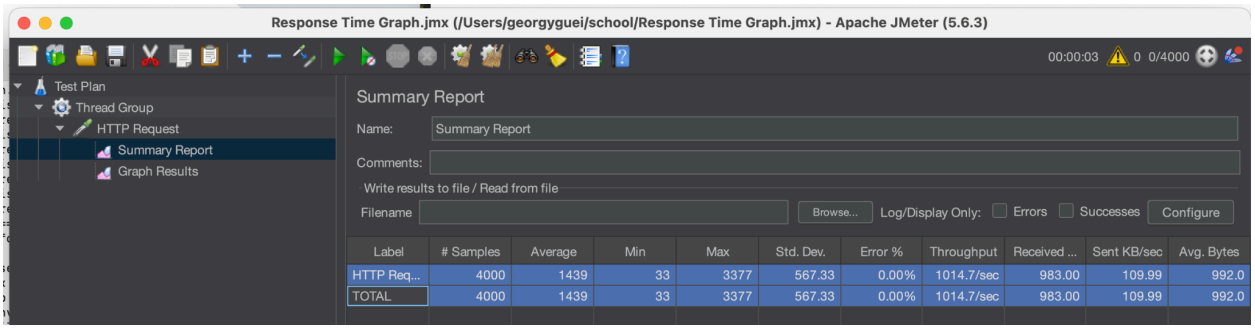


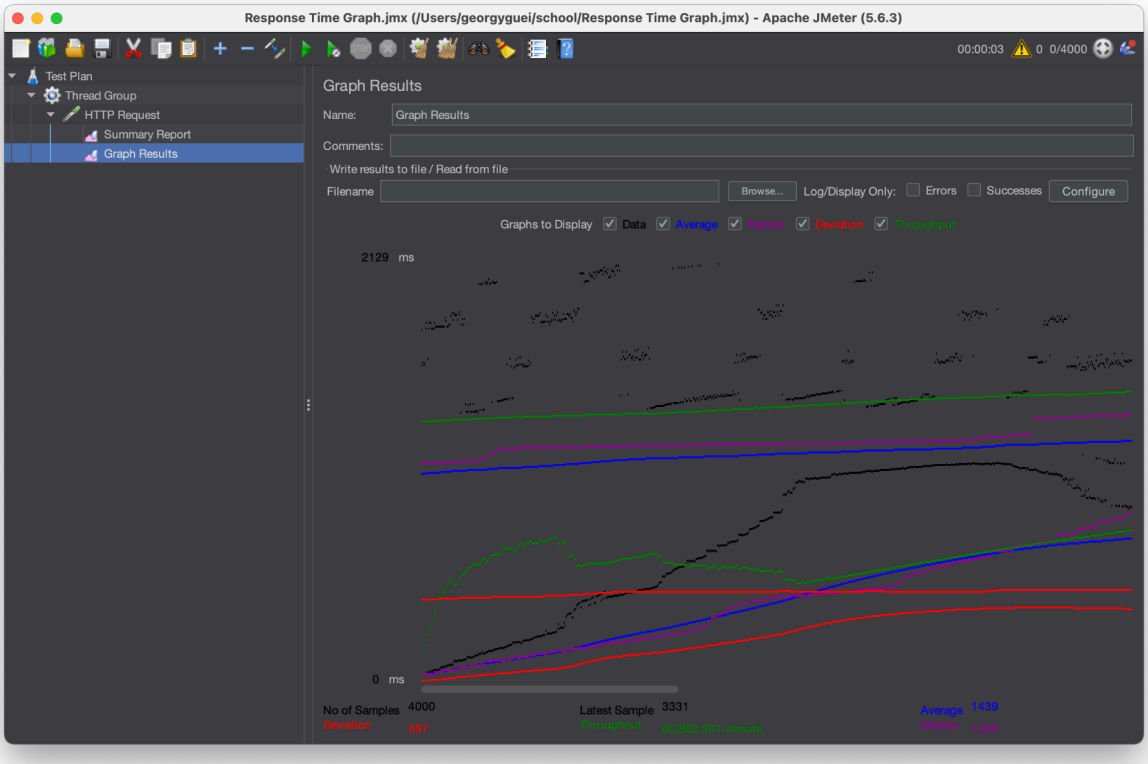
interprétation :

- Le temps de réponse moyen d'environ 0.83 secondes est important à considérer pour l'expérience utilisateur. Il est nécessaire de déterminer si ce temps de réponse est acceptable ou si des améliorations sont nécessaires.
- Le fait que le pourcentage d'erreurs soit de 0.0 est un bon indicateur, signifiant

que le serveur a réussi à gérer toutes les requêtes simulées sans problème.

- Un nombre moyen de requêtes par seconde d'environ 499 suggère une capacité de traitement significative par le serveur.





Interprétation :

- Le temps de réponse moyen d'environ 567 secondes est considérablement plus élevé que lors du test précédent (0.83 secondes), ce qui indique une dégradation des performances sous une charge plus importante. Ce temps de réponse pourrait

ne pas être acceptable pour une expérience utilisateur fluide.
- Le fait que le pourcentage d'erreurs soit de 0.0 est un bon indicateur, signifiant que le serveur a réussi à gérer toutes les requêtes simulées sans problème.
- Un nombre moyen de requêtes par seconde d'environ 1014 suggère une capacité de traitement importante, mais le temps de réponse élevé suggère que le serveur pourrait être proche de sa limite.

Conclusion

Les résultats montrent une dégradation significative des performances sous une charge de 4000 utilisateurs simulés par rapport à 500 utilisateurs. Le temps de réponse moyen de 567 secondes est élevé et devrait être examiné de près. Ces données révèlent que l'application a des difficultés à gérer une charge de 4000 utilisateurs. L'analyse des résultats indique également qu'un goulot d'étranglement potentiel pourrait se trouver au niveau du traitement des requêtes par le serveur sous charge élevée, des mesures d'optimisation sont indispensables pour garantir des performances acceptables.

Bilan

Pour conclure, les tests réalisés ont permis de valider les principales fonctionnalités de l'application de gestion des utilisateurs, de vérifier leur bon fonctionnement du point de vue de l'utilisateur final, et d'évaluer les performances sous charge. Les tests de non-régression ont confirmé que les modifications apportées n'ont pas introduit de nouveaux problèmes. Les résultats des tests montrent que l'application est robuste et performante, cependant, des pistes d'amélioration pour optimiser encore davantage l'expérience utilisateur sont nécessaires.