**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** |
| --- | --- |
| testAddUser() | Succès |
| testAddUserEmailException() | Succès |
| testUpdateUser() | Succès |
| testRemoveUser() | Succès |
| testGetUsers() | Succès |
| testInvalidUpdateThrowsException() | Succès |
| testInvalidDeleteThrowsException() | Succès |

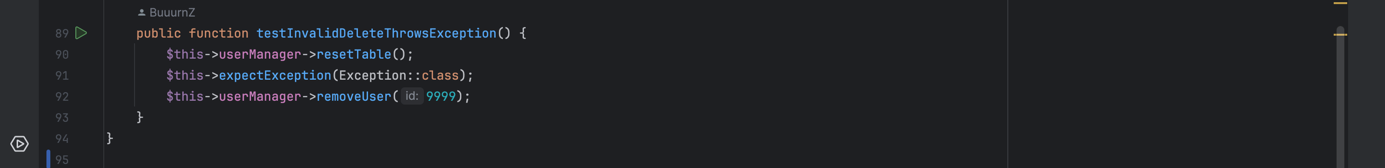
**Captures d'écran des résultats des tests PHPUnit** :

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

Une image contenant logiciel, Logiciel multimédia, texte, Logiciel de graphisme

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, logiciel, Logiciel multimédia, Logiciel de graphisme

Le contenu généré par l’IA peut être incorrect.Une image contenant logiciel, Logiciel multimédia, texte, Logiciel de graphisme

Le contenu généré par l’IA peut être incorrect.

* **Capture 2** : Résultat des différents tests unitaires Une image contenant texte, capture d’écran, Police

  Le contenu généré par l’IA peut être incorrect.

**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** :

* Une image contenant texte, logiciel, Logiciel multimédia, Logiciel de graphisme

  Le contenu généré par l’IA peut être incorrect.**Capture 1** : Code Selenium + Cypress

Une image contenant texte, logiciel, Logiciel multimédia, Logiciel de graphisme

Le contenu généré par l’IA peut être incorrect.

* Une image contenant texte, capture d’écran

  Le contenu généré par l’IA peut être incorrect.**Capture 2** : Résultat des tests

Une image contenant texte, logiciel, Logiciel multimédia, Icône d’ordinateur

Le contenu généré par l’IA peut être incorrect.

**2.3 Tests de Non-Régression**

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

Une image contenant texte, logiciel, Logiciel multimédia, Icône d’ordinateur

Le contenu généré par l’IA peut être incorrect.**Comparaison des résultats avant et après modification** :

Une image contenant logiciel, texte, Logiciel multimédia, Icône d’ordinateur

Le contenu généré par l’IA peut être incorrect.

| **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** :

* Une image contenant capture d’écran, Logiciel multimédia, logiciel, Logiciel de graphisme

  Le contenu généré par l’IA peut être incorrect.**Graphique** :Une image contenant capture d’écran, logiciel, Logiciel multimédia, Logiciel de graphisme

  Le contenu généré par l’IA peut être incorrect.

Une image contenant logiciel, Logiciel multimédia, Logiciel de graphisme, Montage

Le contenu généré par l’IA peut être incorrect.

**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 détecter. 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** :

1. **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