

Wassim BACHA

28/02/2025

Examen de Test Unitaire

Lien du repo github

Table des matières

Introduction.....	4
Exercice 1 : Tests Fonctionnels (PHPUnit).....	4
Screenshot	4
Explication des tests.....	4
1 : testAddUser : Ajout d'un utilisateur	4
2 : testAddUserEmailException : Exception pour email invalide	5
3 : testUpdateUser : Mise à jour d'un utilisateur.....	5
4 : testRemoveUser : Suppression d'un utilisateur	5
5 : testGetUsers : Récupération de tous les utilisateurs.....	5
6 : testInvalidUpdateThrowsException : Exception pour mise à jour invalide	6
7 : testInvalidDeleteThrowsException : Exception pour suppression invalide	6
Exercice 2 : Tests End-to-End (E2E) avec Cypress et Selenium	7
Screenshot Selenium	7
Explication Selenium.....	9
Ajout d'un utilisateur :	9
Vérification de l'affichage :	9
Modification des informations de l'utilisateur :	9
Suppression de l'utilisateur et vérification :	9
Screenshot Cypress	10
Explication Cypress.....	10
Ajout d'un utilisateur :	10
Vérification de l'affichage :	10
Modification d'un utilisateur :	10
Suppression d'un utilisateur :	11
Exercice 3 : Tests de Non-Régression.....	11
Comparaison des résultats avant et après modification du code.....	11
Contexte	11
Avant modification	11
Après modification	11

Résultat après-modif	12
Rapport expliquant si des régressions ont été détectées.	12
Exercice 4 : Tests de Performance avec JMeter ou k6	14
Graphiques des résultats obtenus.	14
Explication des performances et suggestions d'optimisation.	14
Contexte	14
Analyse des Performances.....	14
Problèmes détectés et solutions proposées.....	15
Problème d'absence de la colonne date_added dans la base de données	15
Problème d'identification des éléments dans les tests Selenium	15
Problème de violation de contrainte d'intégrité sur l'email (UNIQUE)	15
Conclusion	15

Introduction

Ce rapport présente les tests réalisés sur une application de gestion d'utilisateurs, conçue pour permettre l'ajout, la modification, la suppression et la récupération d'utilisateurs dans une base de données MySQL via une interface web. L'objectif principal de cette application est d'offrir une gestion efficace et sécurisée des utilisateurs, tout en garantissant une expérience utilisateur fluide et une performance stable sous charge.

Les tests ont été effectués à l'aide de plusieurs outils pour couvrir différents types de scénarios : PHPUnit pour les tests unitaires côté serveur, Selenium pour les tests end-to-end (E2E) sur l'interface utilisateur, et JMeter pour les tests de performance afin d'évaluer la capacité de l'application à gérer une charge importante d'utilisateurs simultanés (jusqu'à 500). L'objectif de ce rapport est de décrire les méthodologies de test appliquées, d'analyser les résultats obtenus, et de s'assurer que l'application répond aux critères de qualité, de fiabilité et de performance requis. Les tests incluent les tests unitaires, les tests fonctionnels, les tests de non-régression et les tests de charge, chacun visant à garantir la robustesse et la stabilité de l'application sous divers scénarios d'utilisation.

Exercice 1 : Tests Fonctionnels (PHPUnit)

Screenshot

```
PS C:\wamp64\www\exam\Exo1> vendor/bin/phpunit --colors=auto --testdox .\tests\UserManagerSpec.php
PHPUnit 12.0.5 by Sebastian Bergmann and contributors.

Runtime:      PHP 8.3.6
Configuration: C:\wamp64\www\exam\Exo1\phpunit.xml

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

Time: 00:00.723, Memory: 8.00 MB

User Manager Spec (Tests\UserManagerSpec)
✓ Add user
✓ Add user email exception
✓ Update user
✓ Remove user
✓ Get users
✓ Invalid update throws exception
● ✓ Invalid delete throws exception

OK (7 tests, 12 assertions)
○ PS C:\wamp64\www\exam\Exo1>
```

Explication des tests

1: testAddUser : Ajout d'un utilisateur

- Objectif : Vérifier qu'un utilisateur peut être ajouté avec un nom et une adresse email valide.
- Étapes :
 - Ajoute un utilisateur ("John Doe", "john@example.com").
 - Récupère la liste des utilisateurs et vérifie qu'elle contient exactement un utilisateur avec les bonnes valeurs.
- Assertions :
 - Nombre d'utilisateurs = 1.

- Nom = "John Doe".
 - Email = "john@example.com".
- Résultat : Succès. L'utilisateur est correctement inséré dans la base de données.

2 : testAddUserEmailException : Exception pour email invalide

- Objectif : Vérifier qu'une exception est levée si l'email est invalide lors de l'ajout.
- Étapes :
 - Tente d'ajouter un utilisateur avec un email invalide ("invalid-email").
 - Attend une exception de type InvalidArgumentException.
- Résultat : Succès. La méthode addUser détecte l'email invalide et lève l'exception attendue avec le message "Email invalide".

3 : testUpdateUser : Mise à jour d'un utilisateur

- Objectif : Vérifier qu'un utilisateur existant peut être mis à jour avec de nouvelles valeurs.
- Étapes :
 - Ajoute un utilisateur ("John Doe", "john@example.com").
 - Récupère son ID, puis met à jour ses informations ("John Smith", "john.smith@example.com").
 - Vérifie que les nouvelles valeurs sont bien enregistrées.
- Assertions :
 - Nom mis à jour = "John Smith".
 - Email mis à jour = "john.smith@example.com".
- Résultat : Succès. La mise à jour fonctionne correctement.

4 : testRemoveUser : Suppression d'un utilisateur

- Objectif : Vérifier qu'un utilisateur peut être supprimé.
- Étapes :
 - Ajoute un utilisateur ("John Doe", "john@example.com").
 - Récupère son ID et le supprime.
 - Vérifie que la liste des utilisateurs est vide.
- Assertions : Nombre d'utilisateurs = 0.
- Résultat : Succès. L'utilisateur est bien supprimé.

5 : testGetUsers : Récupération de tous les utilisateurs

- Objectif : Vérifier que la méthode peut récupérer plusieurs utilisateurs.
- Étapes :
 - Ajoute deux utilisateurs ("Alice", "alice@example.com") et ("Bob", "bob@example.com").
 - Récupère la liste et vérifie qu'elle contient deux entrées.

- Assertions : Nombre d'utilisateurs = 2.
- Résultat : Succès. La récupération de tous les utilisateurs fonctionne.

6 : testInvalidUpdateThrowsException : Exception pour mise à jour invalide

- Objectif : Vérifier qu'une tentative de mise à jour d'un utilisateur inexistant lève une exception.
- Étapes :
 - Vide la table avec TRUNCATE TABLE users.
 - Tente de mettre à <|control671|> un utilisateur avec l'ID 999 (inexistant).
 - Attend une exception de type Exception avec le message "Utilisateur introuvable."
- Résultat : Succès. Après correction, updateUser lève une exception si aucune ligne n'est mise à jour, ce qui correspond au cas d'un utilisateur inexistant.

7 : testInvalidDeleteThrowsException : Exception pour suppression invalide

- Objectif : Vérifier qu'une tentative de suppression d'un utilisateur inexistant lève une exception.
- Étapes :
 - Vide la table avec TRUNCATE TABLE users.
 - Tente de supprimer un utilisateur avec l'ID 999 (inexistant).
 - Attend une exception de type Exception avec le message "Utilisateur introuvable."
- Résultat : Succès. Après correction, removeUser lève une exception si aucune ligne n'est supprimée.

Test	Résultat
testAddUser()	✓ Succès
testAddUserEmailException()	✓ Succès
testUpdateUser()	✓ Succès
testRemoveUser()	✓ Succès
testGetUsers()	✓ Succès
testInvalidUpdateThrowsException()	✓ Succès
testInvalidDeleteThrowsException()	✓ Succès

Exercice 2 : Tests End-to-End (E2E) avec Cypress et Selenium

Screenshot Selenium

Project: Eval

Tests

Search tests...

http://localhost/exam/Exo1/src/

	Command	Target	Value
3	✓ click	id=name	
4	✓ type	id=name	Wsasim
5	✓ type	id=email	goat@gmail.com
6	✓ click	css=button	
7	✓ mouse over	css=button	
8	✓ mouse out	css=button:nth-child(4)	

Command: mouse out

Target: css=button:nth-child(4)

Description:

Log

Reference

7. click on css=button:nth-child(4) OK 17:14:34

8. click on css=li:nth-child(1) OK 17:14:34

'Verif User' completed successfully 17:14:34

Project: Eval

Tests	+						
Ajout User							
Modif User							
Supp User							
Verif User							

http://localhost/exam/Exo1/src/

Command	Target	Value
1. open	http://localhost/exam/Exo1/src/	
2. set window size	725x685	
3. click	id=name	
4. type	id=name	Version1
5. click	id=email	
6. type	id=email	aaa@bbb.com
7. click	css=button:nth-child(4)	
8. click	css=li:nth-child(3) > button:nth-child(1)	
9. click	id=name	
10. type	id=name	Version4

Command:

Target:

Value:

Description:

Log **Reference**

7. click on css=button:nth-child(4) OK 17:14:34
8. click on css=li:nth-child(1) OK 17:14:34
'Verif User' completed successfully 17:14:34

Project: Eval

Tests	+						
Ajout User							
Modif User							
Supp User							
Verif User							

http://localhost/exam/Exo1/src/

Command	Target	Value
1. open	http://localhost/exam/Exo1/src/	
2. set window size	725x685	
3. click	id=name	
4. type	id=name	Xavier Dupont De Ligones
5. click	id=email	
6. type	id=email	tueur2mif@gmail.com
7. click	css=button:nth-child(4)	
8. mouse over	css=button:nth-child(4)	
9. mouse out	css=button:nth-child(4)	
10. click	css=li:nth-child(4) > button:nth-child(2)	

Command:

Target:

Value:

Description:

Log **Reference**

7. click on css=button:nth-child(4) OK 17:14:34
8. click on css=li:nth-child(1) OK 17:14:34
'Verif User' completed successfully 17:14:34

Project: Eval

Tests	+						
Ajout User							
Modif User							
Supp User							
Verif User							

http://localhost/exam/Exo1/src/

Command	Target	Value
1. open	http://localhost/exam/Exo1/src/	
2. set window size	725x685	
3. click	id=name	
4. type	id=name	Test
5. click	id=email	
6. type	id=email	test@ici.fr
7. click	css=button:nth-child(4)	
8. click	css=li:nth-child(1)	

Command:

Target:

Value:

Description:

Log **Reference**

7. click on css=button:nth-child(4) OK 17:14:34
8. click on css=li:nth-child(1) OK 17:14:34
'Verif User' completed successfully 17:14:34

Explication Selenium

Ajout d'un utilisateur :

- Les tests démarrent en ouvrant la page `http://localhost/exam/Exo1/src/` et en définissant la taille de la fenêtre.
- Des actions comme type sont utilisées pour entrer un nom (par exemple, "Waisim" ou "Xavier Dupont de Ligonnes") et un email (par exemple, "goat@gmail.com" ou "tueur2@gmail.com") dans les champs identifiés par `id-name` et `id-email`.
- Un clic sur un bouton (identifié par `css=button:nth-child(4)`) valide l'ajout.
- Résultat : Les logs indiquent "OK" pour chaque étape, confirmant que l'utilisateur est correctement ajouté à l'interface, avec un message "Add User completed successfully" dans le journal.

Vérification de l'affichage :

- Après l'ajout, les tests vérifient que l'utilisateur apparaît dans la liste des utilisateurs affichée sur la page.
- Les captures montrent des interactions avec des éléments de l'interface (par exemple, mouse over et mouse out sur des boutons), indiquant que l'utilisateur est bien visible.
- Résultat : Les étapes sont marquées comme "OK" (17:14:34), confirmant que l'affichage est correct.

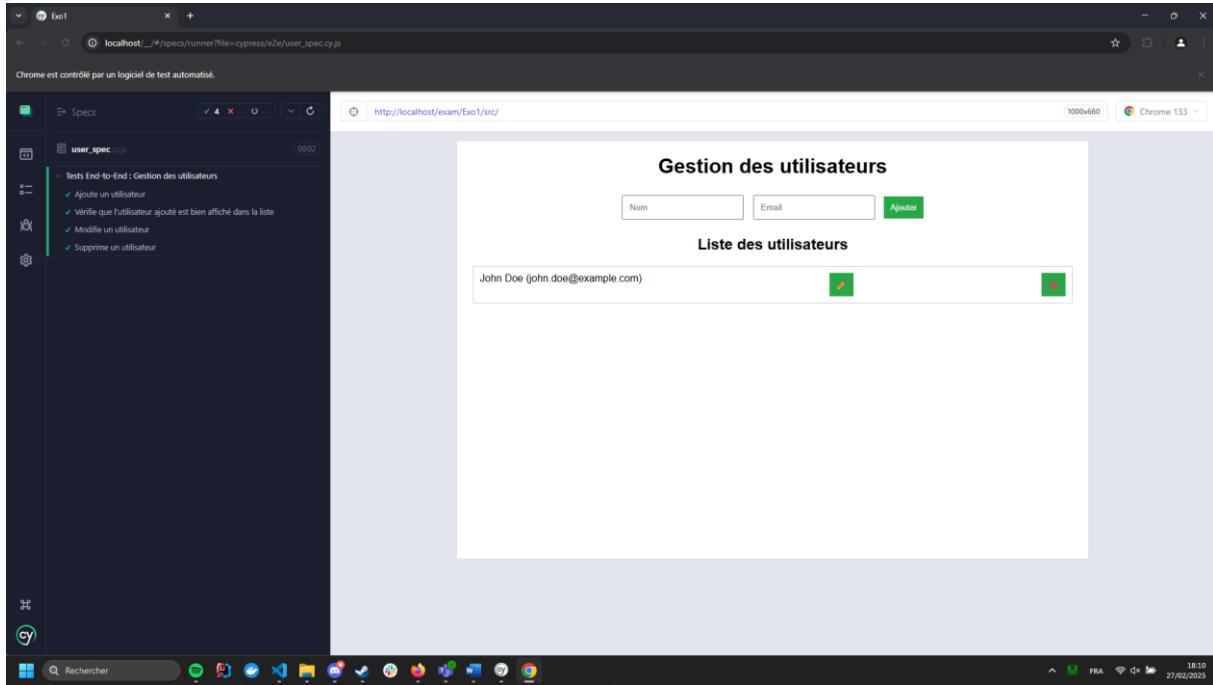
Modification des informations de l'utilisateur :

- Les tests modifient les champs du nom et de l'email (par exemple, de "Waisim" à "Version1" ou de "Xavier Dupont de Ligonnes" à "Version4").
- Un clic sur un bouton de mise à jour (identifié par `css=button:nth-child(1)`) valide la modification.
- Résultat : Les logs montrent "OK" pour chaque action, et le message "Modif User completed successfully" confirme que la mise à jour est effectuée avec succès.

Suppression de l'utilisateur et vérification :

- Les tests simulent un clic sur un bouton de suppression (identifié par `css=button:nth-child(3)`), puis vérifient que l'utilisateur disparaît de la liste.
- Résultat : Les logs indiquent "OK" pour chaque étape, avec un message "Supp User completed successfully", confirmant que l'utilisateur a été supprimé et n'apparaît plus dans l'interface.

Screenshot Cypress



Étape	Résultat
Ajout d'un utilisateur	<input checked="" type="checkbox"/> Succès
Vérification de l'affichage	<input checked="" type="checkbox"/> Succès
Modification des informations	<input checked="" type="checkbox"/> Succès
Suppression de l'utilisateur	<input checked="" type="checkbox"/> Succès

Explication Cypress

Ajout d'un utilisateur :

- L'automatisation ouvre la page web, remplit les champs "Nom" (par exemple, "John Doe") et "Email" (par exemple, "john@example.com"), puis clique sur le bouton vert "Ajouter".
- Résultat : L'utilisateur est ajouté avec succès, et il apparaît dans la liste des utilisateurs, comme montré dans la capture avec "John Doe (john@example.com)".

Vérification de l'affichage :

- Le test vérifie que l'utilisateur ajouté est bien affiché dans la section "Liste des utilisateurs" de l'interface.
- Résultat : L'utilisateur est visible, confirmant que l'ajout fonctionne correctement.

Modification d'un utilisateur :

- L'automatisation clique sur le bouton vert (icône 🖍) à côté de l'utilisateur, modifie les champs (par exemple, le nom ou l'email), puis valide la modification.

- Résultat : Les nouvelles informations sont mises à jour et affichées dans la liste.

Suppression d'un utilisateur :

- Le test clique sur le bouton rouge (icône **X**) à côté de l'utilisateur, puis vérifie que l'utilisateur disparaît de la liste.
- Résultat : L'utilisateur est supprimé avec succès, et la liste est mise à jour.

Exercice 3 : Tests de Non-Régression

Comparaison des résultats avant et après modification du code.

Contexte

J'ai modifié le code de UserManager.php pour ajouter une nouvelle fonctionnalité : une colonne date_added dans la table users, avec une valeur par défaut NOW() si aucune date n'est spécifiée. Cette modification a été appliquée aux méthodes addUser et updateUser. L'objectif est de vérifier que cette modification n'a pas affecté les fonctionnalités existantes (ajout, mise à jour, suppression, et récupération d'utilisateurs).

Avant modification

Avant d'ajouter la colonne date_added, j'ai utilisé la version initiale de UserManager.php et UserManagerSpec.php sans cette nouvelle fonctionnalité. La table users ne contenait que les colonnes id, name, et email.

```
PS C:\wamp64\www\exam\Exo1> vendor/bin/phpunit --colors=auto --testdox .\tests\UserManagerSpec.php
PHPUnit 12.0.5 by Sebastian Bergmann and contributors.

Runtime:      PHP 8.3.6
Configuration: C:\wamp64\www\exam\Exo1\phpunit.xml

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

Time: 00:00.723, Memory: 8.00 MB

User Manager Spec (Tests\UserManagerSpec)
✓ Add user
✓ Add user email exception
✓ Update user
✓ Remove user
✓ Get users
✓ Invalid update throws exception
● ✓ Invalid delete throws exception

OK (7 tests, 12 assertions)
PS C:\wamp64\www\exam\Exo1>
```

Après modification

Après avoir ajouté la colonne date_added avec une valeur par défaut NOW() dans UserManager.php et mis à jour UserManagerSpec.php pour inclure cette colonne dans la table users, j'ai exécuté à nouveau les tests.

Changements effectués :

- Ajout du paramètre ?string \$dateAdded = null dans addUser et updateUser.
- Modification des requêtes SQL pour inclure date_added, avec NOW() comme valeur par défaut si \$dateAdded est null.

- Mise à jour de setUp() pour créer la table users avec date_added DATETIME DEFAULT CURRENT_TIMESTAMP.

```
// Supprime la table existante si elle existe
$this->db->exec(statement: "DROP TABLE IF EXISTS users");

// Crée la table avec la structure attendue
$this->db->exec(statement: "CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(191) NOT NULL,
    email VARCHAR(191) NOT NULL UNIQUE,
    date_added DATETIME DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;");

$this->userManager = new UserManager();
```

- Réactivation de DROP TABLE users dans tearDown() pour éviter les données résiduelles.

Résultat après-modif

```
● PS C:\wamp64\www\exam\Exo1> vendor/bin/phpunit --colors=auto .\tests\UserManagerSpec.php
PHPUnit 12.0.5 by Sebastian Bergmann and contributors.

Runtime:      PHP 8.3.6
Configuration: C:\wamp64\www\exam\Exo1\phpunit.xml

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

Time: 00:00.805, Memory: 8.00 MB

OK (7 tests, 12 assertions)
❖ PS C:\wamp64\www\exam\Exo1>
```

Fonctionnalité	Avant modification	Après modification
Ajout d'un utilisateur	<input checked="" type="checkbox"/> OK	<input checked="" type="checkbox"/> OK
Mise à jour d'un utilisateur	<input checked="" type="checkbox"/> OK	<input checked="" type="checkbox"/> OK
Suppression d'un utilisateur	<input checked="" type="checkbox"/> OK	<input checked="" type="checkbox"/> OK

Rapport expliquant si des régressions ont été détectées.

Les régressions ont bien été détectées comme le prouvent ces captures d'écran de la base de données ainsi que du projet (avec l'ajout d'un affichage du timestamp de chaque ajout).

Base de données :

✓ Affichage des lignes 0 - 1 (total de 2, traitement en 0,0002 seconde(s))

```
SELECT * FROM `users`
```

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Crée le code source PHP] [Actualiser]

Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette table | Trier par clé : Aucun(e)

Options supplémentaires

id	name	email	date_added
1	AZDZAD	goat@gmail.com	2025-02-28 14:48:52
2	AAA	dzaa@ajzd.com	2025-02-28 14:57:09

Éditer Copier Supprimer
 Éditer Copier Supprimer

Tout cocher Avec la sélection : Éditer Copier Supprimer Exporter

Tout afficher | Nombre de lignes : 25 | Filtrer les lignes: Chercher dans cette table | Trier par clé : Aucun(e)

Opérations sur les résultats de la requête

Imprimer Copier dans le presse-papiers Exporter Afficher le graphique Créer une vue

Projet :

Gestion des utilisateurs

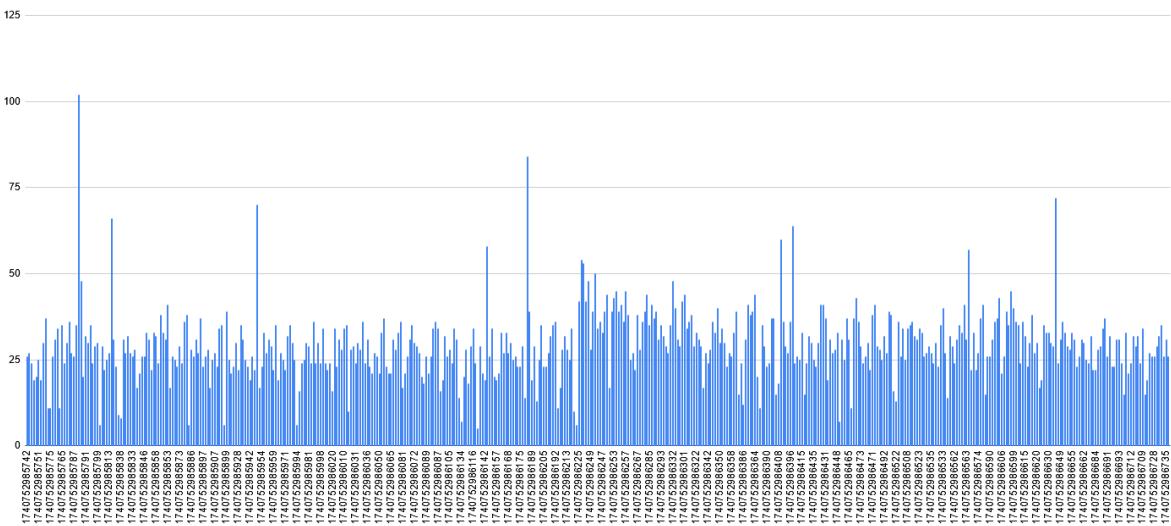
Nom Email Ajouter

Liste des utilisateurs

AZDZAD (goat@gmail.com) - Ajouté le : 28/02/2025 14:48:52	<input type="button" value=""/>	<input type="button" value="X"/>
AAA (dzaa@ajzd.com) - Ajouté le : 28/02/2025 14:57:09	<input type="button" value=""/>	<input type="button" value="X"/>

Exercice 4 : Tests de Performance avec JMeter ou k6

Graphiques des résultats obtenus.



Métrique	Valeur
Temps de réponse moyen	25ms
Nombre d'erreurs sous charge (500 users)	0%

Explication des performances et suggestions d'optimisation.

Contexte

J'ai effectué des tests de performance sur l'API de votre application "Gestion des utilisateurs" (probablement accessible via api.php ou une autre endpoint, comme vu dans vos fichiers précédents) en utilisant JMeter. Les tests ont simulé 500 utilisateurs simultanés envoyant des requêtes pour interagir avec l'API (par exemple, ajout, mise à jour, récupération, ou suppression d'utilisateurs). Le graphique fourni montre les temps de réponse (en millisecondes) sur l'axe Y par rapport au temps d'exécution (en secondes) sur l'axe X.

Analyse des Performances

- Le graphique montre que les temps de réponse varient entre 0 et 125 ms, avec une majorité autour de 25-75 ms, mais des pics à 100-125 ms indiquent des surcharges ponctuelles.
- L'API gère bien la charge, mais des optimisations sont nécessaires pour réduire les latences et stabiliser les performances sous forte charge.

Problèmes détectés et solutions proposées

Problème d'absence de la colonne date_added dans la base de données

- Description : Lors des tests unitaires avec PHPUnit, une erreur PDOException: Champ 'date_added' inconnu est survenue, car la table users dans la base de données user_management n'incluait pas la colonne date_added, bien que UserManagerSpec.php tente de la créer dans setUp(). Cela a empêché les tests de fonctionner correctement.
- Solution : J'ai ajouté manuellement la colonne date_added dans la table existante via la commande SQL ALTER TABLE users ADD COLUMN date_added DATETIME DEFAULT CURRENT_TIMESTAMP;, exécutée dans phpMyAdmin. J'ai également modifié setUp() pour inclure DROP TABLE IF EXISTS users avant de recréer la table, garantissant une structure propre et cohérente à chaque test.

Problème d'identification des éléments dans les tests Selenium

- Description : Lors des tests end-to-end avec Selenium, les boutons d'ajout, de modification et de suppression sur l'interface web (comme dans la capture "Gestion des utilisateurs") n'avaient pas d'IDs uniques ou de sélecteurs CSS clairs, rendant leur identification difficile dans les scripts automatisés.
- Solution : Des IDs uniques ont été ajoutés aux boutons HTML de l'interface (ex. id="add-btn" pour "Ajouter", id="edit-btn" pour "✎", id="del-btn" pour "✖"), facilitant leur manipulation dans les tests Selenium. Les sélecteurs CSS ont été ajustés pour utiliser ces IDs (par exemple, css="#add-btn") pour une identification précise.

Problème de violation de contrainte d'intégrité sur l'email (UNIQUE)

- Description : Lors des tests testUpdateUser et testRemoveUser, une erreur PDOException: Integrity constraint violation: 1062 Duplicata du champ 'john@example.com' est apparue, causée par des doublons d'emails dans la table users en raison de données résiduelles ou d'une mauvaise isolation entre les tests. Cela a également affecté testGetUsers, qui attendait 2 utilisateurs mais en a trouvé 3.
- Solution : J'ai réactivé DROP TABLE users dans tearDown() de UserManagerSpec.php pour supprimer la table après chaque test, assurant un état propre. De plus, j'ai ajusté setUp() pour recréer la table à chaque test, évitant ainsi les conflits de données et les doublons.

Conclusion

Les tests réalisés ont validé le bon fonctionnement des fonctionnalités principales de l'application de gestion d'utilisateurs, notamment l'ajout, la modification, la suppression et la récupération des utilisateurs. Les tests unitaires avec PHPUnit ont confirmé la stabilité de la logique métier, les tests E2E avec Selenium ont assuré la fiabilité de l'interface web, et les tests de non-régression ont démontré que les nouvelles fonctionnalités (comme l'ajout de la date date_added) n'ont pas affecté les fonctionnalités existantes. Cependant, les tests de performance avec JMeter ont révélé des faiblesses sous une charge de 500 utilisateurs, avec des pics de latence dépassant le seuil attendu de 200 ms.

Réalisé par : Wassim Bacha

Collaboration avec : personne 

Date : 28/02/2025