

# Rapport Tests Samuel CHARTON

## Modèle de Rapport de Validation et Tests d'une Application de Gestion de Bibliothèque

### Instructions Générales

Ce modèle de rapport doit être complété par chaque étudiant et soumis en respectant les consignes suivantes :

- Chaque section doit être remplie avec des informations détaillées.
  - Des captures d'écran des tests doivent être incluses.
  - Les explications doivent être claires et précises.
  - Tout problème rencontré doit être décrit avec les solutions envisagées.
- 

## 1. Introduction

À compléter par l'étudiant :

- Présenter brièvement l'application testée et son objectif.
- Décrire les outils utilisés pour les tests (PHPUnit, Cypress, Selenium, JMeter/k6).
- Expliquer l'objectif du rapport et les différents types de tests réalisés.

## 1. L'application

L'application est très simpliste, elle nous permet principalement de faire une liste de tâches (en ajouter et les supprimer)

Gestion de tâches



## 2. Les outils

- **PHPUnit** : pour effectuer les test unitaire sur le backend, le code directement
- **Selenium** : pour faire des tests sur l'interface avec une extension navigateur qui exécute des actions sous forme de scénarios de tests.
- JMeter : pour les tests de performance, savoir si notre serveur peut supporter beaucoup d'utilisateurs, analyser les temps de réponse

## 3. Objectif du rapport

- L'objectif de ce rapport est de rendre compte des différent tests sur l'application. Il permet aussi de montrer si on a eu des régressions sur le code après l'implémentation d'une nouvelle fonctionnalité.

---

# 2. Résultats des Tests

## 2.1 Tests Fonctionnels (PHPUnit)

À compléter par l'étudiant :

- Coller le code des tests
- Présenter les résultats sous forme de tableau.
- Inclure des captures d'écran des résultats des tests.

**Code des tests :**

## 1. addTaskTest.php

```
<?php
```

```
namespace FlowUp\Test;
```

```
use FlowUp\Unitaire\TaskManager;
```

```
use FlowUp\Unitaire\OutOfBoundsException;
```

```
use PHPUnit\Framework\TestCase;
```

```
class AddTaskTest extends TestCase
```

```
{
```

```
    public function testAddTask()
```

```
    {
```

```
        print("Test AddTask\n");
```

```
        $taskManager = new TaskManager();
```

```
        $taskManager->addTask('Task 1');
```

```
        $this->assertCount(1, $taskManager->getTasks());
```

```
        $this->assertEquals('Task 1', $taskManager->getTask(0));
```

```
}  
  
}
```

## 2. removeTaskTest.php

```
<?php
```

```
namespace FlowUp\Test;
```

```
use FlowUp\Unitaire\TaskManager;
```

```
use FlowUp\Unitaire\OutOfBoundsException;
```

```
use PHPUnit\Framework\TestCase;
```

```
class RemoveTaskTest extends TestCase
```

```
{
```

```
    public function testRemoveTask()
```

```
    {
```

```
        print("Test RemoveTask\n");
```

```
        $taskManager = new TaskManager();
```

```
        $taskManager->addTask('Lalala');
```

```
        $taskManager->removeTask(0);
```

```
        $this->assertCount(0, $taskManager->getTasks());  
    }  
}
```

### 3. getTasksTest.php

```
<?php  
  
namespace FlowUp\Test;  
  
use FlowUp\Unitaire\TaskManager;  
use FlowUp\Unitaire\OutOfBoundsException;  
use PHPUnit\Framework\TestCase;  
  
class GetTasksTest extends TestCase  
{  
  
    public function testGetTasks()  
    {  
  
        $taskManager = new TaskManager();  
  
        $taskManager->addTask('Task 1');
```

```

        $taskManager->addTask('Task 2');

        $tasks = $taskManager->getTasks();

        $this->assertCount(2, $tasks);

        $this->assertEquals('Task 1', $tasks[0]);

        $this->assertEquals('Task 2', $tasks[1]);

    }

}

```

#### 4. GetTaskTest.php

```

<?php

namespace FlowUp\Test;

use FlowUp\Unitaire\TaskManager;
use FlowUp\Unitaire\OutOfBoundsException;
use PHPUnit\Framework\TestCase;

class GetTaskTest extends TestCase
{

```

```

public function testGetTask()
{
    print("Test GetTask\n");

    $taskManager = new TaskManager();

    $taskManager->addTask('Task 1');

    $this->assertEquals('Task 1', $taskManager->getTask(0));
}
}

```

5. invalidIndexThrowsExceptionTest.php (tests pour les 2 exeptions dans le même fichier)

```
<?php
```

```
namespace FlowUp\Test;
```

```
use FlowUp\Unitaire\TaskManager;
```

```
use FlowUp\Unitaire\OutOfBoundsException;
```

```
use PHPUnit\Framework\TestCase;
```

```
class InvalidIndexThrowsExceptionTest extends TestCase

{

    public function
testRemoveInvalidIndexThrowsExceptionOnRemove()

    {

        print("Test RemoveInvalidIndexThrowsException on
remove\n");

        $this->expectException(\OutOfBoundsException::class);

        $this->expectExceptionMessage("Index de tâche
invalide: 0");

        $taskManager = new TaskManager();

        $taskManager->removeTask(0);

    }

    public function
testRemoveInvalidIndexThrowsExceptionOnGetTask()

    {

        print("Test RemoveInvalidIndexThrowsException on
getTask\n");

        $this->expectException(\OutOfBoundsException::class);

        $this->expectExceptionMessage("Index de tâche
```



```
invalid: 0");

    $taskManager = new TaskManager();

    $taskManager->getTask(0);

}

}
```

## 6. taskOrderAfterRemovalTest.php

```
<?php

namespace FlowUp\Test;

use FlowUp\Unitaire\TaskManager;
use FlowUp\Unitaire\OutOfBoundsException;
use PHPUnit\Framework\TestCase;

class TaskOrderAfterRemovalTest extends TestCase
{

    public function testTaskOrderAfterRemoval()
    {

        print("Test TaskOrderAfterRemoval\n");
    }
}
```

```

        $taskManager = new TaskManager();

        $taskManager->addTask('Task 1');

        $taskManager->addTask('Task 2');

        $taskManager->addTask('Task 3');

        $taskManager->removeTask(1);

        $tasks = $taskManager->getTasks();

        $this->assertCount(2, $tasks);

        $this->assertEquals('Task 1', $tasks[0]);

        $this->assertEquals('Task 3', $tasks[1]);

    }

}

```

## Tableau :

Test	Résultat
testAddTask()	Succès
removeAddTask()	Succès
testGetTasks()	Succès
testGetTask()	Succès
testRemoveInvalidIndexThrowsExceptionOnGetTask()	Succès
testRemoveInvalidIndexThrowsExceptionOnRemove()	Succès
testTaskOrderAfterRemoval()	Succès

## Capture d'écran :

```
PS C:\Users\FlowUp\Desktop\unitaire> ./vendor/bin/phpunit tests
PHPUnit 12.0.7 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.4.5

Test AddTask
.Test GetTask
.Test GetTasks
.Test RemoveInvalidIndexThrowsException on remove
.Test RemoveInvalidIndexThrowsException on getTask
.Test RemoveTask
.Test TaskOrderAfterRemoval
.                                                    7 / 7 (100%)

Time: 00:00.008, Memory: 8.00 MB

OK (7 tests, 14 assertions)
```

---

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

### À compléter par l'étudiant :

- Décrire le scénario utilisateur testé (ajout, modification, suppression d'un livre).
- Fournir les résultats des tests sous forme de tableau.
- Ajouter des captures d'écran des tests exécutés.

### Scénarios Présent dans le projet

## 1. Ajout d'une Task

Extension : (Selenium IDE) - Selenium IDE - 2.2 Tests End-to-End (E2E) avec Cypress et Selenium\* — Mozilla Fir...

Project: 2.2 Tests End-to-End (E2E) avec Cypress et Selenium\*

Tests ▾ +

Search tests... 🔍

http://127.0.0.1:5500/src/index.html

	Command	Target	Value
✓ AddTask*			
✓ RemoveTask*	1 ✓ open	/src/index.html	
	2 ✓ set window size	778x693	
	3 ✓ click	id=taskInput	
	4 ✓ type	id=taskInput	Test
	5 ✓ click	css=button	
	6 ✓ assert element present	css=.task-item	
	7 ✓ assert text	css=span:nth-child(1)	Test
	8 ✓ close		

Command open // ↗

Target /src/index.html ↶ 🔍

Value

Description

Log	Reference	
3. click on id=taskInput OK		11:19:12
4. type on id=taskInput with value Test OK		11:19:13
5. click on css=button OK		11:19:13
6. click on css=button:nth-child(2) OK		11:19:13
7. assertElementNotPresent on css=.task-item OK		11:19:13
8. close OK		11:19:13
'RemoveTask' completed successfully		11:19:13

## 2. Ajout puis suppression d'une Task

Extension : (Selenium IDE) - Selenium IDE - 2.2 Tests End-to-End (E2E) avec Cypress et Selenium\* — Mozilla Fir...

Project: 2.2 Tests End-to-End (E2E) avec Cypress et Selenium\*

Tests ▾ +

Search tests... 🔍

http://127.0.0.1:5500/src/index.html

	Command	Target	Value
1	✓ open	http://127.0.0.1:5500/src/index.html	
2	✓ set window size	778x693	
3	✓ click	id=taskInput	
4	✓ type	id=taskInput	Test
5	✓ click	css=button	
6	✓ click	css=button:nth-child(2)	
7	✓ assert element not present	css=.task-item	
8	✓ close		

Command: open // [🔗]

Target: http://127.0.0.1:5500/src/index.html [🔍]

Value: [ ]

Description: [ ]

Log	Reference
3. click on id=taskInput OK	11:19:12
4. type on id=taskInput with value Test OK	11:19:13
5. click on css=button OK	11:19:13
6. click on css=button:nth-child(2) OK	11:19:13
7. assertElementNotPresent on css=.task-item OK	11:19:13
8. close OK	11:19:13
'RemoveTask' completed successfully	11:19:13

Tableau :

Étape	Résultat
AddTask	Succès
Suppression d'un livre	Succès

## 2.3 Tests de Non-Régression

**À compléter par l'étudiant :**

- Expliquer les modifications apportées au code.
- Comparer les résultats des tests avant et après modification.
- Fournir une analyse des éventuelles régressions détectées.

**Exemple de tableau à remplir :**

Fonctionnalité	Avant modification	Après modification
Ajout d'un livre	OK	OK / Échec

---

## 2.4 Tests de Performance avec JMeter/k6

**À compléter par l'étudiant :**

- Décrire le test de charge effectué (nombre d'utilisateurs simulés, durée du test, etc.).
- Présenter les résultats sous forme de tableau et graphiques.
- Analyser les performances et proposer des améliorations.

**Exemple de tableau à remplir :**

Métrique	Valeur
Temps de réponse moyen	X ms
Nombre d'erreurs	X %

---

## 3. Problèmes détectés et solutions proposées

**À compléter par l'étudiant :**

- Lister les problèmes rencontrés lors des tests.
  - Expliquer comment ces problèmes ont été analysés et résolus.
- 

## **4. Conclusion**

**À compléter par l'étudiant :**

- Faire un bilan des tests effectués.
  - Proposer des améliorations pour l'application.
- 

Réalisé par : CHARTON Samuel

Collaboration avec : N/A

Date : 19/03/2025