

Les Niveaux de test

Table des matières

I. Importance du test	3
II. Exercice : Quiz	4
III. Niveaux et techniques de test	5
IV. Exercice : Quiz	9
V. Outils	10
VI. Exercice : Quiz	14
VII. Essentiel	15
VIII. Auto-évaluation	15
A. Exercice	15
B. Test	17
Solutions des exercices	18

I. Importance du test

Durée : 1 h 30

Prérequis : avoir suivi le parcours jusqu'ici

Environnement de travail : un ordinateur connecté à Internet

Contexte

Vous avez passé du temps à développer un logiciel ou une application, le porteur de projet l'attend depuis longtemps, toute l'équipe est fébrile à l'idée de boucler le projet.

Comment être sûr de faire bonne impression pendant la démonstration au client ?

Nous allons nous intéresser aux différents niveaux de test logiciel afin de livrer une application solide.

Définition Le test logiciel

D'après *L'institute of Electrical and Electronics Engineers* ou IEEE qui publie le *Standard Glossary of Software Engineering Terminology* (qu'on pourrait traduire par Glossaire standard de la terminologie du génie logiciel), un test se définit comme suit : « Le test est l'exécution ou l'évaluation d'un système ou d'un composant, par des moyens automatiques ou manuels, pour vérifier qu'il réponde à ses spécifications ou identifier les différences entre les résultats attendus et les résultats obtenus »

«Le test est l'exécution ou l'évaluation d'un système ou d'un composant, par des moyens automatiques ou manuels, pour vérifier qu'il répond à ses spécifications ou identifier les différences entre les résultats attendus et les résultats obtenus.»

Les bénéfices du test

Vous pensez sans doute que le principal intérêt de tester son code est d'assurer qu'aucune erreur ne va être mise en production ; c'est vrai. Prenez l'exemple de Nissan, qui en 2014 a rappelé plus d'1 million de véhicules, car ils présentaient un défaut logiciel au niveau de l'activation d'un des airbags. Ou encore Starbucks, qui a fermé pendant 1 jour 60 % de ses boutiques au Canada à cause d'un bug dans le système de vente, entraînant des centaines de milliers de dollars de perte. Au-delà de la perte financière brute, vous perdez aussi du temps.

- **Éviter les coûts supplémentaires**

Faire passer des tests à une application fait appel à tout un protocole qui peut rebuter par le temps à y investir. Mais qu'est-ce que ce temps à côté de celui nécessaire pour mobiliser à nouveau toute une équipe afin de corriger une faille ? Ne pas faire ou faire trop peu de tests est le meilleur moyen de perdre du temps et vous le savez, le temps c'est aussi de l'argent.

- **Gagner du temps**

Prendre le temps de tester et corriger de façon proactive plutôt que réactive vous permettra d'éviter les surprises, d'éviter les retards et satisfera votre Product owner. Il existe même une méthode de développement logiciel nommée le Test Driven Development (TDD) qui comme son nom l'indique est un processus qui consiste à penser au testing d'abord, en développant un logiciel de manière itérative et incrémentale tout en réfléchissant aux jeux de test avant d'écrire le code.

- **Entretenir son image**

Veiller à ne laisser passer aucune malfaçon, c'est aussi entretener son image. Imaginez l'impact sur l'image de la marque quand les utilisateurs de véhicules Nissan ont dû entreprendre des démarches pour faire rappeler leur véhicule ou pire ceux qui ont été victimes du défaut de programmation sur la sensibilité du déclenchement des airbags. Ce cas est évidemment extrême, mais il met en lumière les dégâts potentiels que peuvent avoir un bug sur toute une marque ou une entreprise. Plus terre à terre, si un utilisateur veut consulter la FAQ de votre site et qu'il tombe sur une erreur 404 (page introuvable), ou qu'il se confronte à une erreur 500 (erreur interne du serveur) en voulant se connecter afin de finaliser une commande, il y a de fortes chances pour qu'il se tourne vers une solution concurrente.

- **S'assurer de la sécurité**

Un intérêt supplémentaire des tests est de s'assurer que le logiciel ou application est sécurisé. Les failles de sécurité sont légion sur Internet et s'il est complexe de s'imaginer tous les types d'exploitation dont votre application pourrait être victime, c'est bien à vous de s'assurer qu'elle compliquera la vie des utilisateurs malintentionnés. Vous avez déjà dû voir ce conseil, mais vous ne pouvez pas faire confiance à l'utilisateur. Injections SQL, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), attaques par bruteforce et plus de diverses menaces encore. Vous savez coder pour les éviter ? Mettez votre code à l'épreuve malgré tout. Les effets d'un piratage sont catastrophiques pour une entreprise, la remise à flot est souvent longue et complexe et dans le processus c'est du temps, de l'argent et la confiance des utilisateurs envers vos solutions qui s'envolent avec les données.

- **Déterminer la performance**

Tester son application c'est aussi s'assurer de sa performance. De la même manière qu'un utilisateur changera très vite de site Internet s'il tombe sur une erreur, il quittera la page ou cessera d'utiliser un logiciel si le temps de réponse est trop long. Vous pourriez tester un tchat avec une dizaine d'utilisateurs fictifs et une centaine de messages, mais ce module sera-t-il aussi performant si on multiplie les nombres ? Le test c'est aussi éviter les surprises et envisager tous les cas de figure. Mieux vaut anticiper et tester tous les cas d'utilisation plutôt que de mettre en production une solution fonctionnelle mais peu pérenne, qui enrichira ou créera une dette technique.

- **Obtenir la satisfaction de l'utilisateur**

Finalement, quand on regroupe tous les avantages susmentionnés, tester son application ou logiciel, c'est s'assurer d'offrir la meilleure expérience aux utilisateurs. Un utilisateur conquis par une expérience sans défaut est un utilisateur qui va revenir, en parler autour de lui et vous faire gagner de la réputation. Vous avez sans doute vous-même été confronté à des logiciels, applications ou sites Internet qui vous ont donné envie de fuir et de chercher un équivalent. Vous ne voulez pas devenir la raison même de cette fuite, alors testez !

Complément	La dette technique
-------------------	---------------------------

En règle générale, il y a deux façons d'accomplir une tâche : soit entièrement, soit partiellement, de sorte que le résultat réponde aux besoins à court terme, mais qu'il faille le retravailler ultérieurement. Dans ce dernier cas, une dette technique apparaît, car les retouches nécessaires augmentent le retard de travail. La dette technique survient donc lorsque vous préférez une solution plus simple à court terme plutôt qu'une solution meilleure à long terme, qui peut être plus difficile ou plus coûteuse à mettre en œuvre.

Exercice : Quiz

[solution n°1 p.19]

Question 1

Que signifie IEEE, le sigle pour l'organisation qui produit un Glossaire sur les terminologies du développement logiciel ?

- ☐ Internet Electrical and Electronics Engineers
- ☐ Institute of Electrical and Electronics Engineers
- ☐ Information's Electrical and Electronics Engineers

Question 2

En vous basant sur la définition donnée en début de cours, un test est-il toujours automatisé ?

- ☐ Oui
- ☐ Non

Question 3

Quelle est la méthode de développement citée qui met l'accent sur les tests ?

- ☐ Test Oriented Development
- ☐ Development Driven by Test
- ☐ Test Driven Development

Question 4

Quels sont les avantages des tests sur une application ou un logiciel (plusieurs réponses possibles) ?

- ☐ S'assurer de la sécurité
- ☐ Évaluer la performance
- ☐ Planifier son rendement
- ☐ Entretenir sa réputation

Question 5

Je code afin d'éviter les failles de sécurité types, dois-je quand même tester mon code ?

- ☐ Oui
- ☐ Pas systématiquement
- ☐ C'est inutile

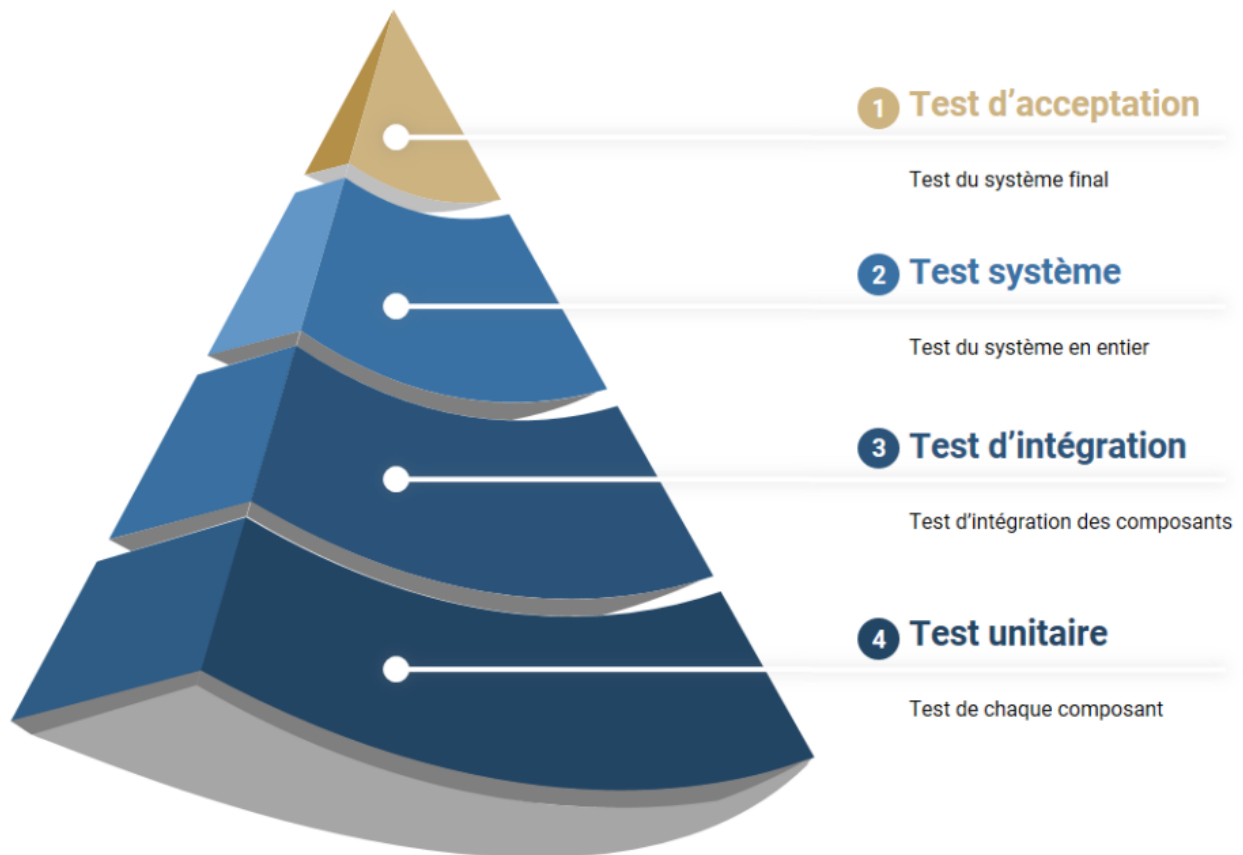
III. Niveaux et techniques de test

Introduction des différents niveaux

Dans le développement logiciel, il existe plusieurs niveaux de test, c'est tout l'intérêt de ce cours. Chaque niveau de test intervient à une étape différente du cycle de développement, ils peuvent eux-mêmes intégrer différents tests et tous les niveaux sont complémentaires.

En règle générale et d'après l'**ISTQB** (« *International Software Testing Qualifications Board* » ou Comité international de qualification du test logiciel en français), il existe quatre niveaux de tests : les tests **unitaires**, les tests **d'intégration**, les tests **système** et les tests **d'acceptation**. Leur objectif est de systématiser les tests logiciels et de reconnaître aisément tous les cas de test possibles à un niveau du développement donné.

Nous allons passer en revue ces quatre niveaux de test, en commençant par les fondations, la base de la pyramide. Gardez à l'esprit que cette unité pédagogique est une introduction aux différents niveaux de tests et que des cours détaillés sur chaque niveau vous seront proposés dans le parcours.



Remarque ISTQB

L'ISTQB est un organisme à but non lucratif qui délivre des certifications de testeurs logiciel. Elle n'est affiliée à aucun acteur majeur des technologies de l'information, la certification n'est pas propre à un système, à un environnement ou à un logiciel.

Test unitaire

Une unité est la plus petite portion testable d'un système ou d'une application qui peut être compilée, chargée et exécutée. Ce type de test, base de la pyramide, permet de tester chaque composant séparément. L'objectif est de tester chaque partie du logiciel en la séparant.

L'intérêt principal de tester ses composants séparément est de ne pas arriver au niveau de test suivant en se demandant « *Mais quelle est la partie qui donne une erreur ?* ». En s'assurant que chaque composant est exempt d'erreur, vous pouvez conclure que l'échec du test vient plutôt de l'alchimie entre les composants plutôt que des composants eux-mêmes.

Vous pourrez par exemple faire des tests unitaires à l'échelle d'une fonction, de façon automatisée ou non. Cependant faire des tests unitaires sans le soutien d'un outil, peut s'avérer coûteux en termes de temps et fastidieux. Nous reviendrons sur une liste d'outils pour différents langages dans la dernière partie de ce cours.

```
1  import org.junit.*;
2  public class WriteAUnitTest {
3      // JUnit calls this method one time before all tests
4      @BeforeClass
5      public static void setUp() {
6          // Place code here for any set up required prior to tests
7      }
8      @AfterClass
9      public static void finished() {
10         // Place code here for any clean up that should be done after tests are finished
11     }
12     @Test
13     public void testFirstName() {
14         Person p=new Person();
15         p.setFirstName("Stephen");
16         Assert.assertEquals("Stephen", p.getFirstName());
17     }
18     @Test
19     public void testLastName() {
20         Person p=new Person();
21         Assert.assertNotNull(p.getLastName());
22     }
23 }
```

Légende : test unitaire avec JUnit, outil de test Java présenté en dernière partie

Les tests d'intégration

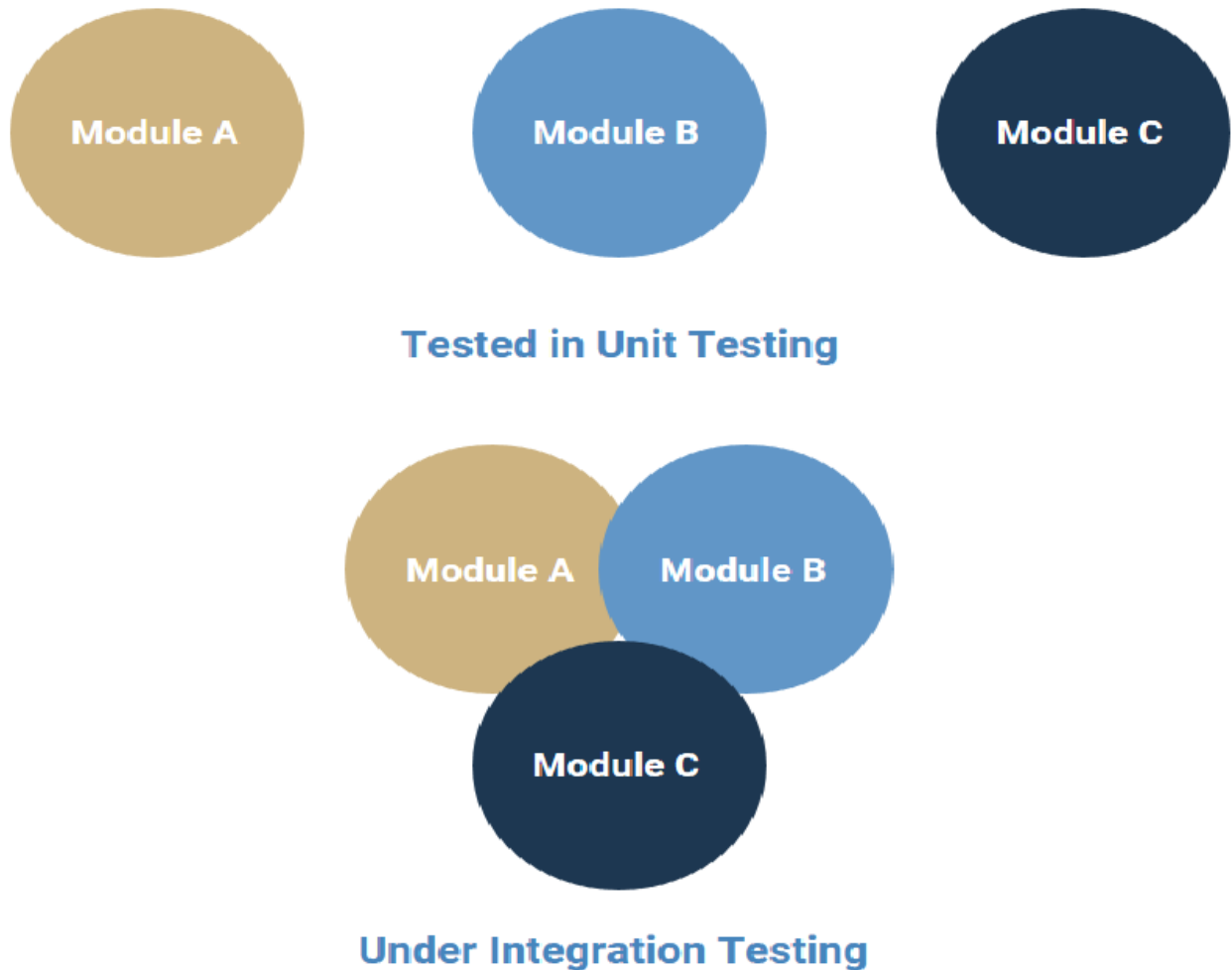
Au deuxième étage de cette pyramide, on retrouve la phase de test où différents modules logiciels sont combinés et testés en tant que groupe pour s'assurer que le système intégré est prêt pour le test du système.

Dans une équipe, les développeurs de chaque module peuvent avoir une logique de programmation différente ; c'est là que le test d'intégration commence à dévoiler son intérêt. Les tests d'intégration vont révéler si les ingrédients mélangés ensemble vont donner une sauce à défaut comestible, au meilleur des cas savoureuse.

Parallèle culinaire mis à part, vous aurez probablement besoin également de tester votre code avec une base de données par exemple.

Il existe différentes approches de tests au sein de ce niveau, comme l'approche Big Bang ou Incremental, elle-même composée d'approches Top Down ou Bottom Up.

Vous en apprendrez plus sur ces dernières dans le cours dédié au test d'intégration.



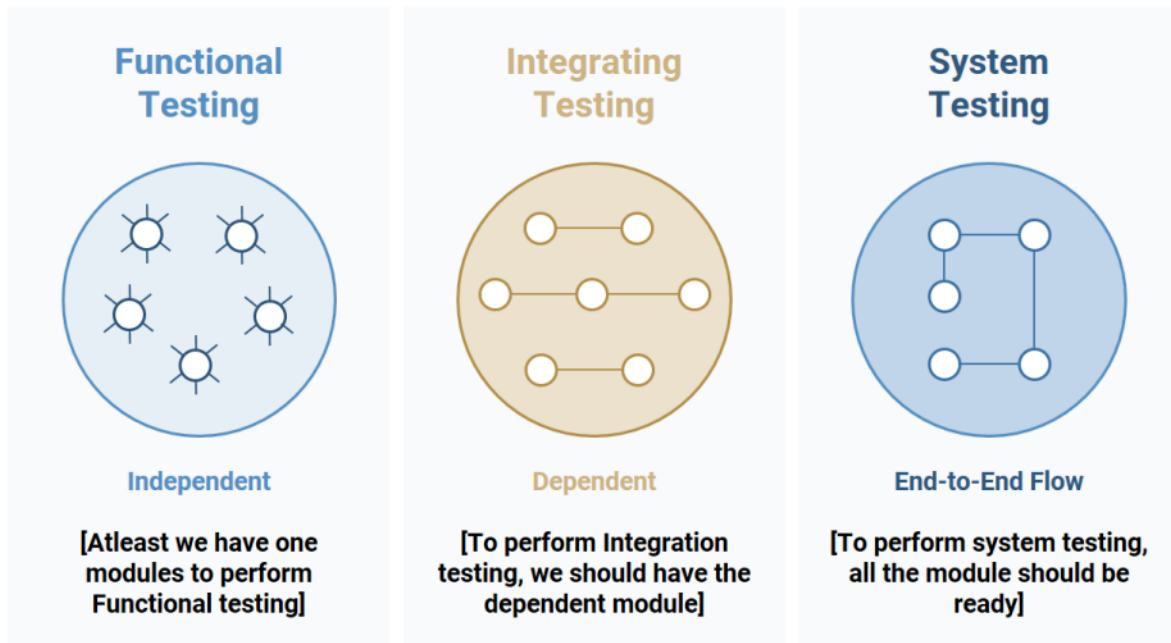
Légende : test unitaire en haut, deuxième étape test d'intégration en bas.

Test du système

Nous continuons notre ascension vers le sommet de la pyramide en restant dans la cuisine, qui décidément est une très bonne analogie à la programmation. Au menu du deuxième niveau de test, soit l'intégration, nous espérons réaliser une délicieuse sauce ; mais la sauce n'est pas un plat de résistance. Le test du système entre en jeu pour savoir si tous les accompagnements mis en ensemble vont réaliser un excellent plat.

Les tests système sont effectués sur un système complet et intégré. Ils permettent de vérifier la conformité du système par rapport aux exigences. Il teste l'interaction globale des composants. Il implique des tests de charge, de performance, de fiabilité et de sécurité. Le test de système est le plus souvent le test final permettant de vérifier que le système répond aux spécifications. Il évalue les besoins fonctionnels et non fonctionnels du test.

Dans ce niveau il existe aussi différents tests comme le test de régression ou de migration.



Test d'acceptation

Les tests d'acceptation sont des tests effectués pour déterminer si les exigences d'une spécification ou d'un contrat sont satisfaites à la livraison. Les tests d'acceptation sont essentiellement effectués par l'utilisateur ou le client. C'est le moment de savoir si votre client va trouver le repas à son goût. Il est l'heure de composer son plus beau cahier de recette (pas de blague avec la cuisine, c'est vraiment son nom) qui regroupe les procédures de tests, les sorties attendues des tests et les sorties effectives (traditionnellement avec une notation OK pour un test réussi, KO pour un test échoué), les attentes du client en termes de résultat des tests etc. Se renseigner sur la création et le maintien d'un bon cahier de recette est un sujet de veille instructif.

Remarque L'intégration continue

Ou Continuous Integration en anglais, souvent vu sous le sigle CI de paire avec le sigle CD pour Continuous Delivery. Cette pratique, issue de la mouvance DevOps, tend à se démocratiser. Les développeurs intègrent le code dans un dépôt, de préférence plusieurs fois par jour. Chaque intégration peut ensuite être vérifiée de façon automatisée et des tests conduits automatiquement également. Bien que les tests automatisés ne fassent pas strictement partie de l'intégration continue, ils sont généralement implicites. Vous retrouverez un cours y étant dédié.

Exercice : Quiz

[solution n°2 p.20]

Question 1

Qu'est-ce qu'une unité dans le test unitaire ?

- ☐ Une page de code
- ☐ La plus petite portion de code testable
- ☐ Un module entier

Question 2

Combien de niveaux de test existe-t-il d'après l'ISTQB ?

- ☐ 4
- ☐ 5
- ☐ 6

Question 3

En quoi consiste le test unitaire ?

- ☐ Tester si les composants s'intègrent tous ensemble
- ☐ Tester individuellement chaque composant
- ☐ Tester si la commande est conforme aux attentes du client

Question 4

Quel test est réalisé en présence du client ?

- ☐ Test de régression
- ☐ Test du système
- ☐ Test d'acceptation

Question 5

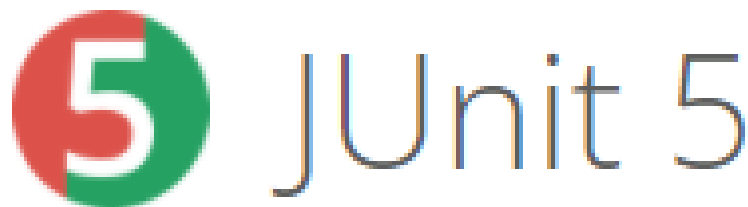
Quelle pratique de test est issue de la mouvance DevOps ?

- ☐ Intégration Continue
- ☐ Top Down Approach
- ☐ Test Driven Development

V. Outils

Les outils de tests unitaires

Pour vous assister dans vos tests unitaires il existe quelques outils populaires :



JUnit¹ est un framework de test unitaire open source pour JAVA. Il est utile aux développeurs Java pour écrire et exécuter des tests reproductibles.

¹ <https://junit.org/junit5/>

PHPUnit

PHPUnit¹ vous permet de réaliser des tests pour PHP. Une documentation en français est proposée sur le site officiel du framework.



Mocha² est un framework de test Javascript basé sur Node.js et votre navigateur. La documentation en anglais est très accessible.

Unittest³ est un module présent dans la librairie Python. Il vous permet également de faire des tests unitaires. Il suffit d'importer unittest et de créer une classe dont le nom commence par test.

1 <https://phpunit.de/>

2 <https://mochajs.org/>

3 <https://docs.python.org/3/library/unittest.html#module-unittest>

Exemple Test Python avec unittest

Dans le premier exemple, on cherche à vérifier que le 3 + 4 font bien 7. À l'exécution du test on obtient un « OK » car en effet mathématiquement l'opération est correcte.

```
1 <code>
2 import unittest
3
4 def add(x,y):
5     return x + y
6 class MyTest(unittest.TestCase):
7     def test(self):
8         self.assertEqual(add(3,4), 7)
9 if __name__ == '__main__':
10     unittest.main()
11 </code>
12
```

Output :

```

.....

Ran 1 test in 0.000s

OK

```

Exemple

Dans le second exemple, on cherche à vérifier si 3 + 4 font 8. Le test échoue en précisant que 7 (le résultat obtenu) n'est pas égal à 8 (le résultat prévu).

```
1 <code>
2 import unittest
3
4 def add(x,y):
5     return x + y
6 class MyTest(unittest.TestCase):
7     def test(self):
8         self.assertEqual(add(3,4), 8)
9 if __name__ == '__main__':
10     unittest.main()
11 </code>
12
```

Output :

```
F
=====
=====
FAIL: test (__main__.MyTest)
-----
Traceback (most recent call last):
  File "sample2.py", line 8, in test
    self.assertEqual(add(3,4), 8)
AssertionError: 7 != 8
-----

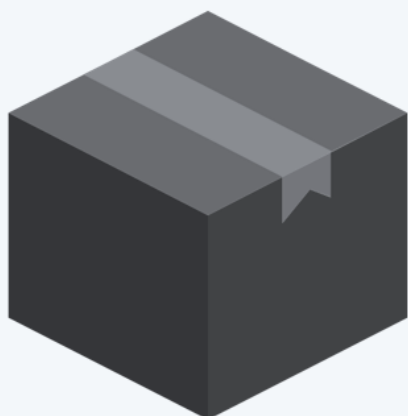
Ran 1 test in 0.001s

FAILED (failures=1)
```

Complément Les approches de test

Après avoir découvert les niveaux de test, il est intéressant de savoir qu'il existe trois grandes techniques de test s'appliquant à l'informatique (programmation comme test de pénétration), qui sont les tests de la boîte noire, grise et blanche.

QA testers



Black box - we do not know anything

Developers



White box - we know everything

- **La boîte noire, opaque, ou black box**, est une technique qui permet d'examiner le fonctionnement d'un logiciel d'un point de vue externe sans approfondir l'examen du code. L'avantage des tests en boîte noire est qu'ils peuvent être appliqués à tous les niveaux de test.

Dans une boîte noire, il est impossible de voir à travers la boîte ; nous ne pouvons que constater que la boîte est noire. De la même manière, dans le domaine des tests de logiciels, le test de la boîte noire est une approche de test permettant d'étudier l'application testée sans miser sur sa conception et sa structure interne. Il est mis en place pour examiner le comportement de l'application d'un point de vue extérieur ; c'est pourquoi on le nomme aussi test comportemental. Il s'agit d'observer comment le logiciel se comporte dans plusieurs scénarios sans regarder les détails du code.

Nous avons tous effectué des tests en black box dans notre vie en appuyant sur le bouton « *power* » d'un appareil électrique en espérant que celui-ci s'allume, sans pour autant comprendre tous les mécanismes que cela implique.

Pour résumer, cette technique se focalise sur la fonctionnalité du logiciel. Ce dernier intervient aux dernières étapes du processus de test pour examiner les aspects fonctionnels du logiciel. C'est typiquement le type de test dédié aux QA testers, des professionnels dont le métier est de tester des logiciels.

- **Le test en boîte blanche ou white box**, à contrario, est une méthodologie de test purement axée sur le fonctionnement de la structure interne du logiciel. Il nécessite une connaissance approfondie du codage pour comprendre la manière dont le logiciel est articulé. Le test boîte noire n'est pas une alternative au test boîte blanche. Ils sont complémentaires. Quand vous naviguez sur un site Internet et que vous effectuez une opération comme valider une commande, vous attendez une validation ou une erreur, c'est le test black box. Si en revanche vous scrutez le code source de ce site pour vérifier le fonctionnement de la page, ses variables et que vous employez vos connaissances du code pour comprendre le comportement, il s'agit d'un test en boîte blanche. Finalement, c'est l'étendue de vos connaissances en programmation qui sont mises à contribution pour vérifier la conception et la structure interne de l'application. Vous, développeur, pratiquez des tests en boîte blanche quand ils sont effectués sur votre travail.
- **Le test en boîte grise, gray box ou grey box**, est le savant mélange des deux techniques précédentes. Le testeur a accès partiellement à des informations sur la structure interne du code. Il combine les bénéfices des tests et la connaissance des développeurs afin d'améliorer la qualité générale du produit. Le test est effectué du point utilisateur plutôt que de celui du concepteur.

Exercice : Quiz

[solution n°3 p.21]

Question 1

Quel outil listé vise à tester du code en Javascript ?

- ☐ Mocha
- ☐ Latte
- ☐ Cappuccino

Question 2

Il est nécessaire de télécharger unittest :

- ☐ Vrai
- ☐ Faux

Question 3

Quelle approche de test est celle de testeur QA ?

- ☐ Boîte Noire
- ☐ Boîte Grise
- ☐ Boîte Blanche

Question 4

Quelle est la technique de test mise en place en prenant connaissance de la structure interne du code ?

- ☐ Boîte Noire
- ☐ Boîte Grise
- ☐ Boîte Blanche

Question 5

De quel point de vue est réalisé un test en boîte grise ?

- ☐ Celui du client
- ☐ Celui du développeur
- ☐ Celui de l'utilisateur

VII. Essentiel

Le test est indispensable à la construction d'un logiciel ou d'une application. Il existe diverses techniques de test qui sont les tests en boîte noire, grise ou blanche. Les tests des logiciels interviennent à des étapes différentes du cycle de développement et sont divisés en quatre niveaux, qui sont les tests unitaires, d'intégration, système et d'acceptation. Vous disposez de divers outils, logiciels ou bibliothèques pour vous assister lors des tests, qui sont consignés dans un cahier de recette.

VIII. Auto-évaluation

A. Exercice

Vous disposez de plusieurs aquariums et souhaitez utiliser un compteur pour vous aider à transférer vos poissons d'un environnement à un autre. À cette fin vous avez développé un module nommé Aqua>Transfer, construit avec plusieurs composants dont la splendide interface prototype est visible ci-dessous. Vos aquariums sont identifiés par des ID à 4 chiffres. Vous souhaitez mettre en place des tests unitaires pour ce module.

L'idée pour cette introduction au testing n'est pas d'être exhaustif, mais de s'entraîner à imaginer tous les cas de figure. Dans des formulaires comme celui illustré ci-dessous, vous devrez vous mettre à la place de nombre d'utilisateurs qui entreront des données invalides.

AQUA > Transfer

Depuis l'aquarium n° :

Vers l'aquarium n° :

Nombre de poissons à transférer :

Transférer

Annuler

Question 1

[solution n°4 p.22]

Identifiez tous les composants. Combien sont-ils ?

Question 2

[solution n°5 p.22]

Écrivez un jeu de tests avec les résultats possibles pour chaque composant. Vous devez fournir une donnée ou une action en entrée (ex : clic), le résultat du test (OK / KO) et y associer un message d'erreur ou commentaire.

Indice :

Quelques exemples de test avec un composant input mail sur un module formulaire de connexion :

Champ adresse mail		
Données en entrée	Résultat	Commentaire de sortie
adresse@example.fr	OK	Validation
adresse.example.fr	KO	Erreur → L'adresse ne contient pas d'arobase
adre sse@example.fr	KO	Erreur → L'adresse mail ne peut contenir d'espace

B. Test**Exercice 1 : Quiz**

[solution n°6 p.25]

Question 1

En partant de la base de la pyramide des niveaux de test, quel est le test qui se trouve au 3^{ème} niveau ?

- ☐ Système
- ☐ Intégration
- ☐ Acceptation

Question 2

Le client de la solution développée ne participe à aucun test :

- ☐ Vrai
- ☐ Faux

Question 3

Quel document regroupe les procédures de test effectuées ?

- ☐ Livre de cuisine
- ☐ Cahier de recette
- ☐ Fiche d'intervention

Question 4

Qu'est-ce que la dette technique ?

- ☐ Une dette créée à cause de trop nombreuses commandes matérielles
- ☐ Une dette créée car des solutions peu pérennes ont été mises en place

Exercice 6

[solution n°7 p.26]

En partant cette fois-ci du sommet de la pyramide, quel est le bon ordre des niveaux de test ?

1.
2.
3.
4.


Réponse : ____

Solutions des exercices

Exercice p. 4 Solution n°1**Question 1**

Que signifie IEEE, le sigle pour l'organisation qui produit un Glossaire sur les terminologies du développement logiciel ?

- ☐ Internet Electrical and Electronics Engineers
- ☒ Institute of Electrical and Electronics Engineers
- ☐ Information's Electrical and Electronics Engineers

 IEEE est le sigle de l'*Institute of Electrical and Electronics Engineers* qui publie le *Standard Glossary of Software Engineering Terminology*.

Question 2

En vous basant sur la définition donnée en début de cours, un test est-il toujours automatisé ?


- ☐ Oui
- ☒ Non

 Un test n'est pas toujours automatisé, loin de là, cependant le gain de temps est indéniable.

Question 3

Quelle est la méthode de développement citée qui met l'accent sur les tests ?


- ☐ Test Oriented Development
- ☐ Development Driven by Test
- ☒ Test Driven Development

 C'est le Test Driven Development qui pense aux tests d'abord et au code ensuite.

Question 4


Quels sont les avantages des tests sur une application ou un logiciel (plusieurs réponses possibles) ?

- ☒ S'assurer de la sécurité
- ☒ Évaluer la performance
- ☐ Planifier son rendement
- ☒ Entretenir sa réputation

 Les tests ont beaucoup de bénéfices, mais planifier le rendement financier de l'application n'en fait pas partie.

Question 5


Je code afin d'éviter les failles de sécurité types, dois-je quand même tester mon code ?

- ☒ Oui
- ☐ Pas systématiquement
- ☐ C'est inutile
-  L'humain fait parfois des erreurs malgré ses grandes connaissances : testez toujours votre code, surtout quand il s'agit de sécurité. Pas jamais, pas parfois : systématiquement.

Exercice p. 9 Solution n°2


Question 1

Qu'est-ce qu'une unité dans le test unitaire ?

- ☐ Une page de code
- ☒ La plus petite portion de code testable
- ☐ Un module entier
-  Une unité est la plus petite portion de code testable. Un module, voire toute une page, serait testé dans le test d'intégration.


Question 2

Combien de niveaux de test existe-t-il d'après l'ISTQB ?

- ☒ 4
- ☐ 5
- ☐ 6
-  D'après l'ISTQB il existe 4 niveaux de test soit : les tests unitaires, d'intégration, du système et d'acceptation.


Question 3

En quoi consiste le test unitaire ?

- ☐ Tester si les composants s'intègrent tous ensemble
- ☒ Tester individuellement chaque composant
- ☐ Tester si la commande est conforme aux attentes du client
-  Le test unitaire est un test individuel de chaque composant.

Question 4

Quel test est réalisé en présence du client ?

- ☐ Test de régression
- ☐ Test du système
- ☒ Test d'acceptation
-  Le test d'acceptation est réalisé en présence de la personne à qui la solution développée est destinée, le client. C'est lui qui validera ou non le pourcentage d'erreurs acceptables du logiciel avant mise en production.


Question 5

Quelle pratique de test est issue de la mouvance DevOps ?

☒ Intégration Continue

☐ Top Down Approach

☐ Test Driven Development

 La Top Down Approach fait partie du niveau de test d'intégration, le Test Driven Development est une méthode de développement, la bonne réponse est donc l'intégration continue.


Exercice p. 14 Solution n°3**Question 1**

Quel outil listé vise à tester du code en Javascript ?

☒ Mocha

☐ Latte

☐ Cappuccino


 Si tous vous réchauffent en hiver, c'est Mocha qui vous permet de tester du JS.

Question 2

Il est nécessaire de télécharger unittest :

☐ Vrai

☒ Faux

 Nul besoin de télécharger unittest, le module est présent nativement avec Python.


Question 3

Quelle approche de test est celle de testeur QA ?

☒ Boîte Noire

☐ Boîte Grise

☐ Boîte Blanche

 Le testeur QA n'a pas développé le logiciel ou l'application, il ignore comment elle est conçue, il utilise donc la technique de la boîte noire.

Question 4

Quelle est la technique de test mise en place en prenant connaissance de la structure interne du code ?

☐ Boîte Noire

☐ Boîte Grise

☒ Boîte Blanche

- Q Si vous vous appuyez sur votre connaissance de la structure interne du code pendant les tests, vous utilisez donc l'approche de la boîte blanche.

Question 5

De quel point de vue est réalisé un test en boîte grise ?

- ☐ Celui du client
- ☐ Celui du développeur
- ☒ Celui de l'utilisateur

- Q Afin de créer la meilleure solution possible d'un point de vue expérience utilisateur, il est nécessaire de se mettre à la place de celui-ci. La bonne réponse est donc : le point de vue utilisateur.

p. 16 Solution n°4

Le module Aqua>Transfer est constitué de 5 composants qui sont :

- Le champ « *Depuis l'aquarium n°* »
- Le champ « *Vers l'aquarium n°* »
- Le champ « *Nombre de poisson à transférer* »
- Le bouton « *Transférer* »
- Le bouton « *Annuler* »

p. 17 Solution n°5

Champ « Depuis »		
Données en entrée	Résultats	Commentaire de sortie
1 234	OK	Validation
123	KO	Erreur → Nombre entier à 4 chiffres seulement
12 345	KO	Erreur → Nombre entier à 4 chiffres seulement
(champ vide)	KO	Erreur → Entrer une valeur
aquarium 400	KO	Erreur → Entrer seulement des chiffres
9 999	KO	Erreur → Cet aquarium n'existe pas

Champ « Vers » : Jeu identique au champ « depuis »


Champ « <i>Nombre de poissons à transférer</i> »		
Données en entrée	Résultat	Commentaire de sortie
12	OK	Validation
- 12	KO	Erreur → Nombre entier positif uniquement
(champ vide)	KO	Erreur → Entrer une valeur
douze	KO	Erreur → Entrer seulement des chiffres
99	KO	Erreur → Cet aquarium contient moins de poissons

Bouton « <i>Transférer</i> »		
Données en entrée	Résultat	Commentaire de sortie
Champs remplis	OK	Message → Quantité transférée avec succès
Action : clic		
(champ quantité vide)	KO	Erreur → Entrer une quantité
(champ Depuis vide)	KO	Erreur → Entrer une ID valide dans « <i>Depuis</i> »
(champ « <i>Vers</i> » vide)	KO	Erreur → Entrer une ID valide dans « <i>Vers</i> »

Bouton « Annuler »		
Données en entrée	Résultat	Commentaire de sortie
Champs remplis Action : clic	OK	Toutes les valeurs des champs ont été réinitialisées
Champs remplis	KO	Erreur → Les valeurs n'ont pas été réinitialisées


Exercice p. 17 Solution n°6**Question 1**

En partant de la base de la pyramide des niveaux de test, quel est le test qui se trouve au 3^{ème} niveau ?

- ☒ Système
- ☐ Intégration
- ☐ Acceptation
-  C'est le test système qui intervient au 3^{ème} niveau de la pyramide.


Question 2

Le client de la solution développée ne participe à aucun test :

- ☐ Vrai
- ☒ Faux
-  Le client participe au test d'acceptation, car c'est bien dans le nom, il doit accepter ce dont on lui fait la démonstration.

Question 3

Quel document regroupe les procédures de test effectuées ?

- ☐ Livre de cuisine
- ☒ Cahier de recette
- ☐ Fiche d'intervention
-  Le livre de cuisine n'a pas sa place dans les tests, et la fiche d'intervention concerne davantage un Technicien d'Assistance en Informatique. Reste le très important cahier de recette qui est la bonne réponse.

Question 4

Qu'est-ce que la dette technique ?

- ☐ Une dette créée à cause de trop nombreuses commandes matérielles
- ☒ Une dette créée car des solutions peu pérennes ont été mises en place
- ☐ Bien qu'une dette puisse être accumulée à cause de commandes trop importantes, celle qui nous intéresse et que nous essayons d'éviter, est causée par des solutions mises en place visant l'impact immédiat plutôt que le long terme.

Exercice p. 18 Solution n°7

En partant cette fois-ci du sommet de la pyramide, quel est le bon ordre des niveaux de test ?

Test d'acceptation

Test système

Test d'intégration

Test unitaire

