Secteur Tertiaire Informatique
Filière « Etude et développement »

Séquence « Développer des pages Web »

Introduction

Apprentissage

Mise en pratique

Evaluation

Les Tests en Développement Web

TABLE DES MATIERES

1. LES DIFFERENTS TESTS RELATIFS A LA REALISATION D'UNE APPLICATION WEB. 3

1.1 LES TESTS UNITAIRES 3
1.2 LES TESTS D'INTEGRATION 4
1.3 LES TESTS DE VALIDATION 5
1.4 LES TESTS DE PERFORMANCE 5
1.5 LES TESTS D'ACCEPTATION (RECETTE) 7

1. LES DIFFERENTS TESTS RELATIFS A LA REALISATION D'UNE APPLICATION WEB.

1.1 LES TESTS UNITAIRES

En programmation informatique, le **test unitaire** (ou « T.U. », ou « U.T. » en anglais) ou **test de composants** est une procédure permettant de vérifier le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme (appelée « unité » ou « module »).

Dans les applications non critiques, l'écriture des **tests unitaires** a longtemps été considérée comme une tâche secondaire. Cependant, les méthodes Extreme programming (XP) ou **Test Driven Development** (**TDD**) ont remis les **tests unitaires**, appelés « *tests du programmeur* », au centre de l'activité de programmation.

Les tests unitaires sont avant tout écrits comme étant une bonne pratique destinée à aider les développeurs à identifier et **corriger les bugs**, à **refactoriser** le code et à servir de documentation pour une unité du logiciel testé.

Les divers Framework Front ou Back proposent tous des outils intégrés (généralement des bibliothèques de classes) qui permettent la mise en œuvre de **Tests unitaires**.

- Pour **Angular** par exemple on utilisera conjointement **Karma** et **Jasmine**.
- Pour **Symfony** ce sera **PHPUnit** qui est intégré dans le célèbre Framework.

https://phpunit.readthedocs.io/fr/latest/writing-tests-for-phpunit.html

1.2 LES TESTS D'INTEGRATION

Dans le monde du développement informatique, le **test d'intégration** est une phase de tests, **précédée** par les **tests unitaires** et généralement suivie par les **tests de validation**, vérifiant le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme (appelée « unité » ou « module ») ; dans le **test d'intégration**, chacun des modules indépendants du logiciel est assemblé et testé dans l'ensemble.

L'objectif de **chaque phase** de **test** est de détecter les erreurs qui n'ont pas pu être détectées lors de la **précédente phase**.

Avant de tester que l'assemblage des quatre pieds et du plateau d'une table est possible et que l'ensemble est solide, un menuisier commence par s'assurer que chaque pied est solide et à la bonne hauteur, et que le plateau est à la bonne dimension et lui aussi bien solide. **Ensuite**, seulement ensuite, il commence à **tester l'ensemble**.

Vous l'aurez compris : **avant** de réaliser des **tests d'intégration**, il faut s'assurer que des **tests unitaires** pertinents ont été réalisés sur **les différents composants à intégrer**.

Pour cela, le **test d'intégration** a pour cible de détecter les erreurs non détectables par le **test unitaire**.

Le test d'intégration permet également de vérifier l'aspect fonctionnel, les performances et la fiabilité du logiciel.

L'intégration fait appel en général à un **système de gestion de versions**, et éventuellement à des programmes d'installation. Cela permet d'établir une nouvelle version, fondée soit sur une version de maintenance, soit sur une version de développement.

Le concept **d'intégration continue** fait référence à la notion **d'intégration**. Le principe consiste à rendre systématique et plus ou moins automatique l'intégration des différents composants d'un système dès que ces composants sont modifiés, pour faire en sorte que les effets produits par ces modifications soient rapidement mesurables.

Ceci permet notamment d'éviter les gros bugs difficiles à déceler, et favorise **l'agilité** dans les projets.

Afin de prendre la mesure et de comprendre comment mettre en place ces séquences de Tests dans un projet de développement Web je vous propose de suivre le lien :

https://openclassrooms.com/fr/courses/4087056-testez-et-suivez-letat-de-votre-application-php

1.3 Les Tests de Validation

Un **test de validation** est un type de test informatique qui permet de vérifier si toutes les **exigences client**, décrites dans le document de **spécification** du logiciel, sont respectées.

1.4 Les Tests de Performance

Un **test de performance** est un test dont l'objectif est de déterminer la performance d'un **système informatique**.

L'acception la plus courante de ce terme est celle dans laquelle ces tests logiciels vont avoir pour objectif de mesurer les temps de réponse d'un système applicatif en fonction de sa sollicitation.

Cette définition est donc très proche de celle de **test de charge** où l'on mesure le comportement d'un système en fonction de la charge d'utilisateurs simultanés. Seuls les tests de charge permettent de valider correctement une application ou un système **avant déploiement**, tant en qualité de service qu'en consommation de ressources.

Ces tests peuvent être de plusieurs types, notamment :

Test de charge : il s'agit d'un test au cours duquel on va **simuler** un nombre **d'utilisateurs virtuels** prédéfinis, afin de valider l'application pour une charge attendue d'utilisateurs. Ce type de test permet de mettre en évidence les points sensibles et critiques de l'architecture technique. Il permet en outre de mesurer le dimensionnement des serveurs, de la bande passante nécessaire sur le réseau, etc.

Test de performance : proche du Test de charge, il s'agit d'un test au cours duquel on va mesurer les performances de l'application soumise à une charge d'utilisateurs. Les informations recueillies concernent les temps de réponse utilisateurs, les temps de réponse réseau et les temps de traitement d'une requête sur le(s) serveur(s).

La nuance avec le type précédent réside dans le fait qu'on ne cherche pas ici à valider les performances pour la charge attendue en production, mais plutôt vérifier les performances intrinsèques à différents niveaux de charge d'utilisateurs.

Test de dégradations des transactions : il s'agit d'un **test technique** primordial au cours duquel on ne va simuler que l'activité transactionnelle **d'un seul scénario fonctionnel** parmi tous les scénarios du périmètre des tests, de manière à déterminer quelle charge le système est capable de supporter pour chaque scénario fonctionnel et d'isoler éventuellement les transactions qui dégradent le plus l'ensemble du système.

La démarche utilisée est d'effectuer une montée en charge linéaire jusqu'au premier point de blocage ou d'inflexion. Pour dépasser celui-ci, il faut paramétrer méthodiquement les composants systèmes ou applicatifs afin d'identifier les paramètres pertinents, ce jusqu'à obtention des résultats satisfaisants. Méthodologiquement, ce test est effectué avant les autres types de tests tels que performance, robustesse, etc. où tous les scénarios fonctionnels sont impliqués.

Test de stress : il s'agit d'un test au cours duquel on va simuler l'activité maximale attendue **tous scénarios fonctionnels** confondus en heure de pointe de l'application, pour voir comment le système réagit au maximum de l'activité attendue des utilisateurs.

La durée du palier en pleine charge, en général de 2 heures, doit tenir compte du remplissage des différents caches applicatifs et clients, ainsi que de la stabilisation de la plateforme de test suite à l'éventuel effet de pic-rebond consécutif à la montée en charge.

Dans le cadre de ces tests, il est possible de *pousser le stress* jusqu'à simuler des défaillances systèmes ou applicatives afin d'effectuer des tests de récupération sur incident (Fail-over) ou pour vérifier le niveau de service en cas de défaillance.

Test de robustesse, d'**endurance**, de **fiabilité** : il s'agit de tests au cours desquels on va **simuler** une **charge importante d'utilisateurs** sur une **durée relativement longue**, pour voir si le système testé est capable de supporter une activité intense sur une longue période sans dégradations des performances et des ressources applicatives ou système.

Le résultat est satisfaisant lorsque l'application a supporté une charge supérieure à la moitié de la capacité maximale du système, ou lorsque l'application a supporté l'activité d'une journée ou plusieurs jours/mois/années, pendant 8 à 10 heures, sans dégradation de performance (temps, erreurs), ni perte de ressources systèmes.

Test de capacité, test de **montée en charge** : il s'agit d'un test au cours duquel on va **simuler un nombre d'utilisateurs** sans cesse **croissant** de manière à déterminer quelle charge limite le système est capable de supporter.

Éventuellement, des paramétrages peuvent être effectués, dans la même logique que lors des tests de dégradation, l'objectif du test étant néanmoins ici de déterminer la capacité maximale de l'ensemble système-applicatif dans une **optique prévisionnelle**.

Test aux limites : il s'agit d'un test au cours duquel on va **simuler** en général une **activité bien supérieure à l'activité normale**, pour voir comment le système réagit aux limites du modèle d'usage de l'application.

Proche du test de capacité, il ne recouvre pas seulement l'augmentation d'un nombre d'utilisateurs simultanés qui se limite ici à un pourcentage en principe prédéfini, mais aussi les possibilités d'augmentation du nombre de processus métier réalisés dans une plage de temps ou en simultané, en jouant sur les cadences d'exécutions, les temps d'attente.

Test de résilience : il s'agit d'un test au cours duquel on va **simuler** une **activité réelle** tout en **simulant** des pannes en environnement réel et de vérifier que le système informatique continue à fonctionner.

Proche du test de robustesse, il permet de s'assurer de la résilience des applications et de l'infrastructure sous-jacente. Pour provoquer les pannes pendant le test, on peut utiliser des *chaos monkey* qui mettent aléatoirement hors service des instances dans l'environnement de test.

1.5 LES TESTS D'ACCEPTATION (RECETTE)

En informatique, le **test d'acceptation** (ou **recette**) est une phase de développement des projets, visant à assurer formellement que le **produit** est **conforme** aux **spécifications** (réponse donnée à un instant « t » aux attentes formulées). Elle s'inscrit dans les activités plus générales de qualification.

Cette étape implique, en la présence effective des différents acteurs du projet, **maîtrise d'œuvre** et **maîtrise d'ouvrage**, le déroulement rigoureux de procédures de tests préalablement décrits, et l'identification de tout écart fonctionnel ou technique.

La procédure de recette se déroule en deux étapes principales :

- Les tests système.
- Les tests d'acceptation utilisateur.

Si la première étape a lieu chez le **fournisseur**, la deuxième se déroule en revanche généralement dans les locaux et avec les **infrastructures du client**.

La **recette usine** comprend tous les tests réalisés chez le **fournisseur**, **avant** la **livraison**. Elle désigne donc les **tests unitaires**, les **tests de validation** et les **tests d'intégration**.

D'un point de vue **maîtrise d'ouvrage**, la recette usine correspond à une période de quelques jours pour valider que la livraison correspondra à sa commande.

En ce qui concerne les **tests unitaires**, de **validation** et **d'intégration**, ceux-ci sont considérés comme relevant des **tests de qualification en amont**, au niveau du **fournisseur**.

Phase très importante pour celui-ci, car c'est elle qui lui permet de vérifier que son produit est de qualité.

La durée de la phase de **test d'acceptation**, signifie aussi la période (généralement courte) durant laquelle le **client procède lui-même à ses propres tests** dans **ses locaux**, avant **d'accepter les livrables**.

À l'issue de la **recette usine**, le **fournisseur** et le **client** signent un **procès-verbal** de fin de recette usine, qui accompagne la **livraison du produit** et le **cahier de recette**.

Lors de l'étape de vérification d'aptitude (**VA**) ou **vérification d'aptitude au bon fonctionnement** (**VABF**) (aptitude à répondre aux besoins exprimés dans le cahier des charges initial) ou **recette utilisateur**, le client réalise deux catégories de tests différentes.

D'un côté, une **recette technique** est effectuée afin de vérifier que le produit livré est **techniquement conforme** sur toute la chaîne de processus.

De l'autre, la **maîtrise d'ouvrage** contrôle l'aspect **fonctionnel** du produit lors de la **recette fonctionnelle**.

La recette fonctionnelle a pour but la **validation des fonctionnalités** exprimées dans le **cahier des charges** et détaillées dans les **spécifications fonctionnelles**.

Si la **VABF** se déroule correctement et est validée, le client procède alors à la **mise** en **service opérationnelle**.

Une période de vérification de service régulier (VSR) commence donc par un premier déploiement sur un site pilote.
Cette mise en production permet de valider le produit en **conditions réelles**.

À la différence des étapes précédentes, celle-ci se déroule pleinement en environnement de production avec des données réelles.

CREDITS

ŒUVRE COLLECTIVE DE l'AFPA
Sous le pilotage de la DIIP et du centre d'ingénierie sectoriel Tertiaire-Services

Equipe de conception (IF, formateur, mediatiseur)
Formateur : Alexandre RESTOUEIX
Sources Wikipédia et OpenClassRooms

Date de mise à jour : 08/01/2021