WorkShop Innovation n°2

WorkShop Innovation n°2

Olivier AUVERLOT

NOTE

- Présentation de SQL Maestro PHP Generator
- PHP Generator est un générateur d'applications web basé sur l'exploitation d'un modèle relationnel. Il permet la réalisation rapide d'applications de type CRUD dotées de fonctionnalités avancées (formulaires dynamiques, contrôles dans le navigateur, applications responsive, exportation de données, impression, graphiques, gestion de l'authentification et des privilèges, etc.).
- Grâce à un mécanisme d'événements, le développeur peut implémenter du code métier dans la partie serveur (en PHP) et dans le code client (en Javascript). PHP Generator propose une API facilitant l'implémentation de l'application et se base également sur de nombreuses librairies du monde libre (Smarty, JQuery, PHP Spreadsheet, etc.).

Farid AIT KARRA

Un peu d'histoire...

> Les 5% de bogues découverts après release représentent 95% des coûts de correction.

WTF pour World Taekwondo Federation, fédération mondiale de taekwondo ;-)

- https://fr.wikipedia.org/wiki/WTF

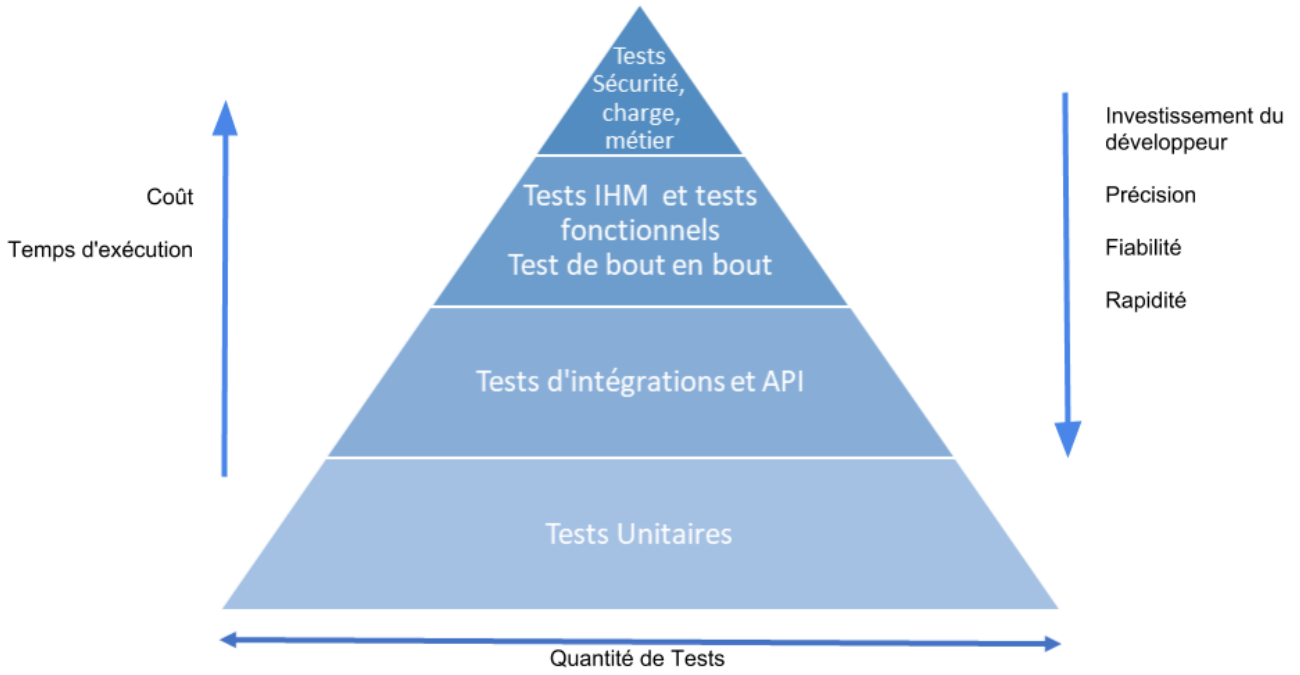
L'Intégration Continue est une méthode agile.

Plus précisément, elle fait partie des 12 méthodes d' eXtreme Programming.

Elle se base sur les principes agiles suivants :

- Fabriquer souvent ("build") ;
- Tester souvent ("test") ;
- Intégrer souvent ("integrate").

Points sur les tests



Tests unitaires

Les tests unitaires (TU)

NOTE

Initiés par le développeur lui-même dans l'optique est de vérifier son code au niveau du composant qu'il doit réaliser. Ces tests doivent être automatisés rapidement pour permettre de valider la non régression du fonctionnement du composant lors des multiples livraisons des différentes version du logiciel, surtout en process agile.

Les tests de couverture de code

NOTE

La notion de « couverture de code » va de pair avec les tests unitaires. Il s'agit de la mesure du taux de code source d'une application exécuté lorsque les tests unitaires sont lancés.

Peut-on dire alors qu'une couverture de 100% du code source équivaut à dire que l'application est parfaitement testée ? Absolument pas en fait. La couverture de code, calculée par le framework de tests unitaires utilisé, n'est qu'une mesure **quantitative**, et non **qualitative**.

En effet, la couverture de code ne contrôle pas que des assertions soient présentes dans les tests, que les bons comportements soient vérifiés ni que les tests soient réellement unitaires, lisibles et bien écrits.

Les tests par mutation

NOTE

Il n'est plus à prouver l'utilité des tests unitaires. Ils sont essentiels dans la conception d'une application de qualité. Mais, savons-nous quantifier leur pertinence, leur qualité ?

Un indicateur de couverture du code par les tests à 100%, ne signifie pas du code 100% testé. Cet indicateur ne détermine que grossièrement le pourcentage de code exécuté lors du passage des tests unitaires, pas plus.

Voici une technique qui vous permettra d'accorder plus de confiance à vos tests.

Le processus de cette technique se déroule en deux grandes étapes : la génération de mutants, puis le carnage de ceux-ci. WTF ?

Bilan / analyse

Pour un mutant donné, il y a deux résultats possibles, soit les tests sont toujours au vert, soit au moins l'un d'eux est passé au rouge.

Habituellement on souhaite que nos tests soient au vert, mais dans ce contexte, on veut du rouge. Rouge sang qui attestera de la mort de notre mutant.

En effet si, au minimum, un des tests est en échec cela prouve qu'ils sont capables de détecter les modifications du code métier source. En revanche, si tous les tests sont toujours au vert, le mutant survit, il est donc invisible aux yeux de nos tests.

Un mutant qui survit est potentiellement un test manquant !

- https://kaibee.fr/les-tests-mutants-avec-pitest/

Les tests de qualité de code

NOTE

Les défaillances logicielles augmentent considérablement le coût du développement d'applications. Trouver et corriger les erreurs de production est souvent 100 fois plus coûteux que de les trouver et de les corriger pendant les phases de conception et de codage. Il est essentiel que les équipes intègrent la qualité à toutes les phases du développement logiciel et automatisent autant que possible la vérification de la qualité pour déceler les défauts tôt dans le processus et éviter les efforts répétés. C'est ce que l'on entend par « Qualité Continue » ou CQ, qui constitue aujourd'hui une garantie essentielle – ou une garantie de qualité – dans les cycles de livraison rapides des workflows DevOps et CI/CD.

- https://www.sonarlint.org/
- http://sonarqube.urba.univ-lille.fr/

Tests d'intégration et API

Les tests d intégration (Continuous intégration : CI)

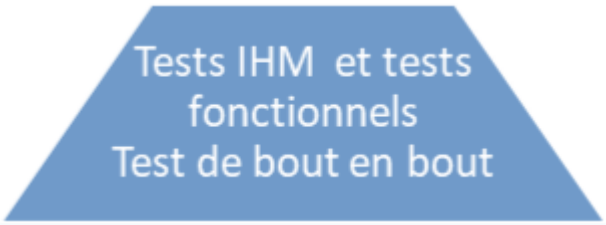
NOTE

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

Pour cela, le test d'intégration a pour cible de détecter les erreurs non détectables par le test unitaire1.

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.

Tests IHM et tests fonctionnels de bout en bout



Les tests d acceptation (Behavior-Driven Development : BDD)

NOTE

il s'agit ici de v51alider l'adéquation du logiciel aux spécifications du client. En fait, toute solution informatique est motivée par un besoin au niveau des utilisateurs. Alors, sur la base d'un cahier des charges arrêté et établi en amont avec le client, ce test rassure sur la conformité du logiciel aux critères d'acceptation et aux besoins des cibles. Ils sont donc généralement réalisé par le client final ou les utilisateurs, ont peut aussi appeler cela la "recette" du logiciel.

Les tests de non-regression (TNR)

NOTE

Un test de non régression permet de valider que la mise en ligne d'une nouvelle fonctionnalité sur un logiciel n'impactera pas les fonctions déjà existantes. Les tests fonctionnels auront bien validé que la nouvelle fonction est opérationnelle mais c'est le test de non régression qui validera que cette dernière n'impacte pas les autres fonctionnalités du logiciel.

Le test de non régression est déployé sur un environnement de recette et doit vérifier au minimum que les fonctionnalités principales ou « critiques » du logiciel sont toujours disponibles après la livraison de nouveaux développements.

- http://selenium.urba.univ-lille.fr/

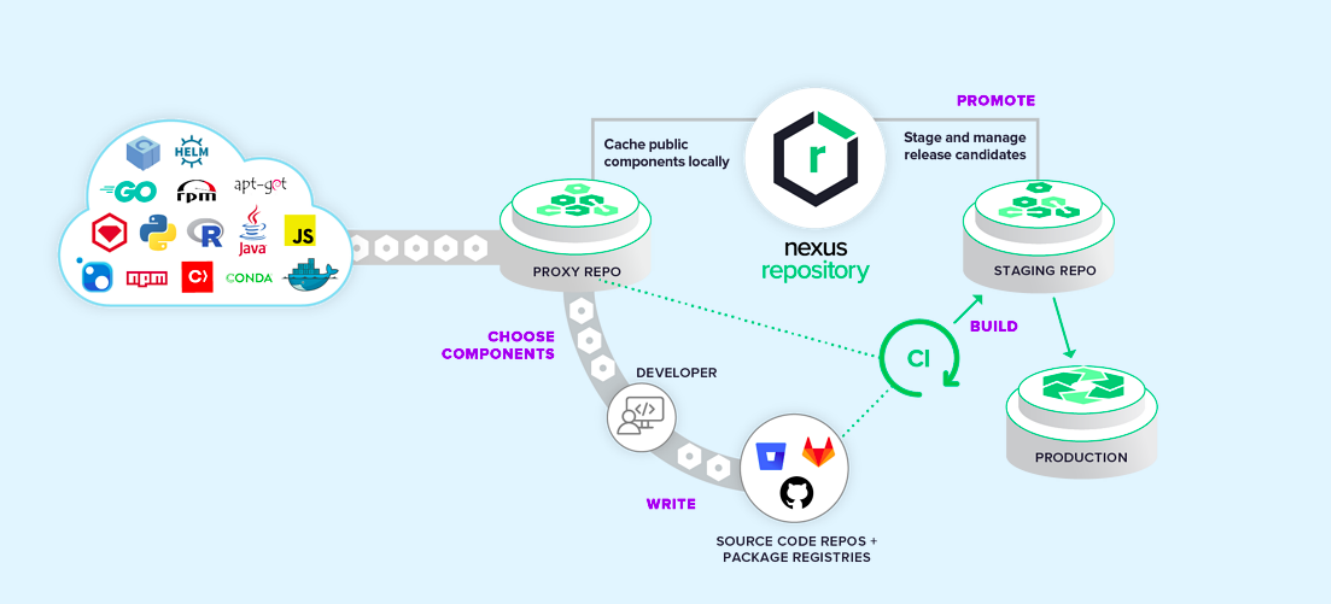
Tests sécurité, charge, métier

`Les tests de montée en charge`

NOTE

Un test de montée en charge des applications WEB consiste à solliciter un site Internet en lui appliquant un certain nombre de requêtes automatisées appelés scénarios de test de charge . Cela permet d'observer le comportement de l'application WEB au delà de l'audience normale.

Livraison continue (TAG gitlab / release management avec NEXUS)



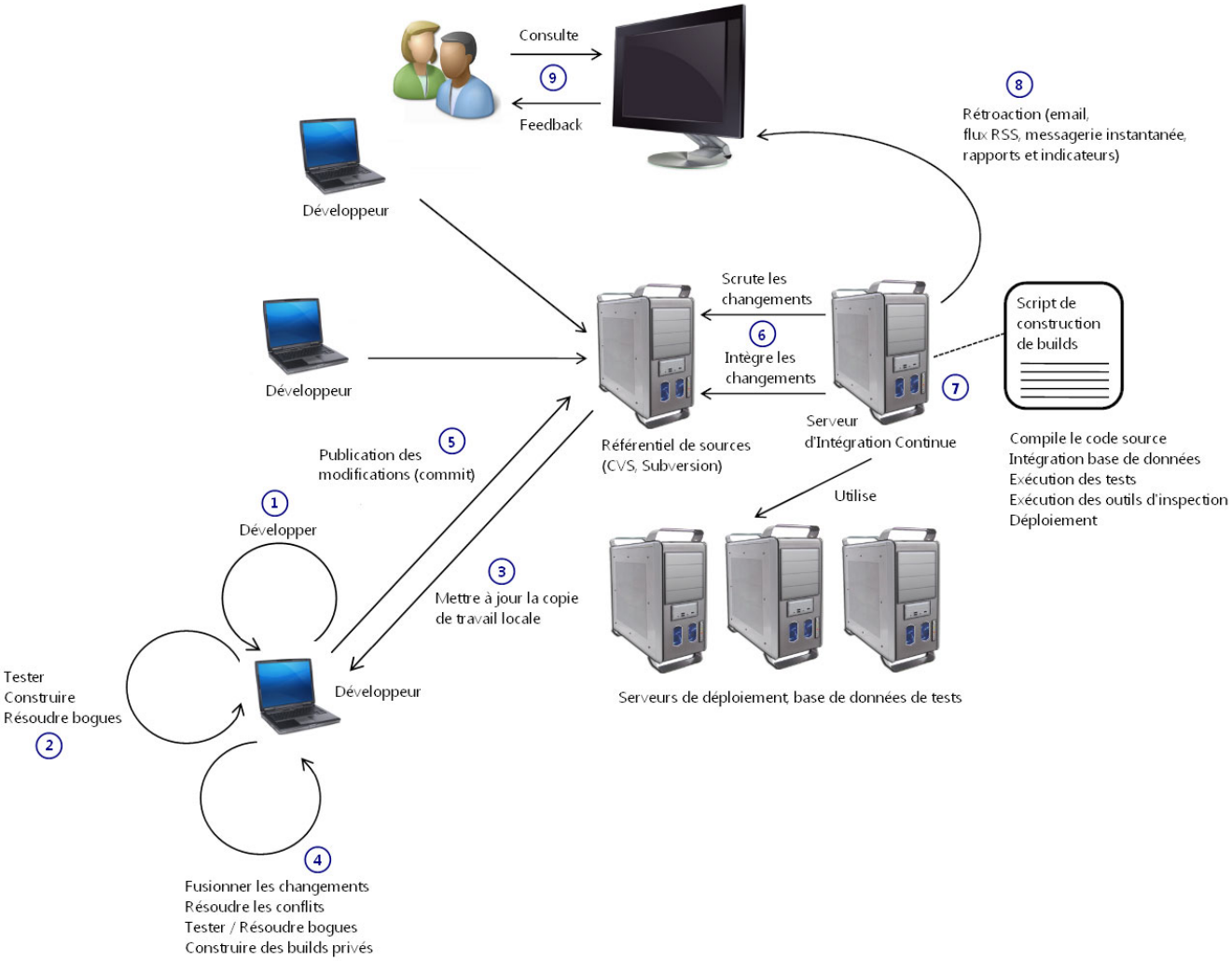
Release management

NOTE

Un Repository : source unique de vérité pour tous vos composants, binaires et artefacts de build Distribution efficace des composants et des conteneurs aux développeurs

- http://nexus.urba.univ-lille.fr/

Résumé



Questions (FeedBack)...

© Copyright :

Tour d’horizon des tests dans nos applications (Yannick grenzinger et Maxime Gellé)

- https://www.youtube.com/watch?v=AeuCcq_bCDw
- https://blog.octo.com/mutation-testing-un-pas-de-plus-vers-la-perfection/
- https://www.all4test.fr/differents-test-logiciel/
- https://fr.wikipedia.org/wiki/Test_d%27int%C3%A9gration
- https://www.nutcache.com/fr/blog/tests-unitaires/
- https://fr.sonatype.com/nexus/repository-pro
- https://fr.wikipedia.org/wiki/SOLID_(informatique)
- https://12factor.net/fr/