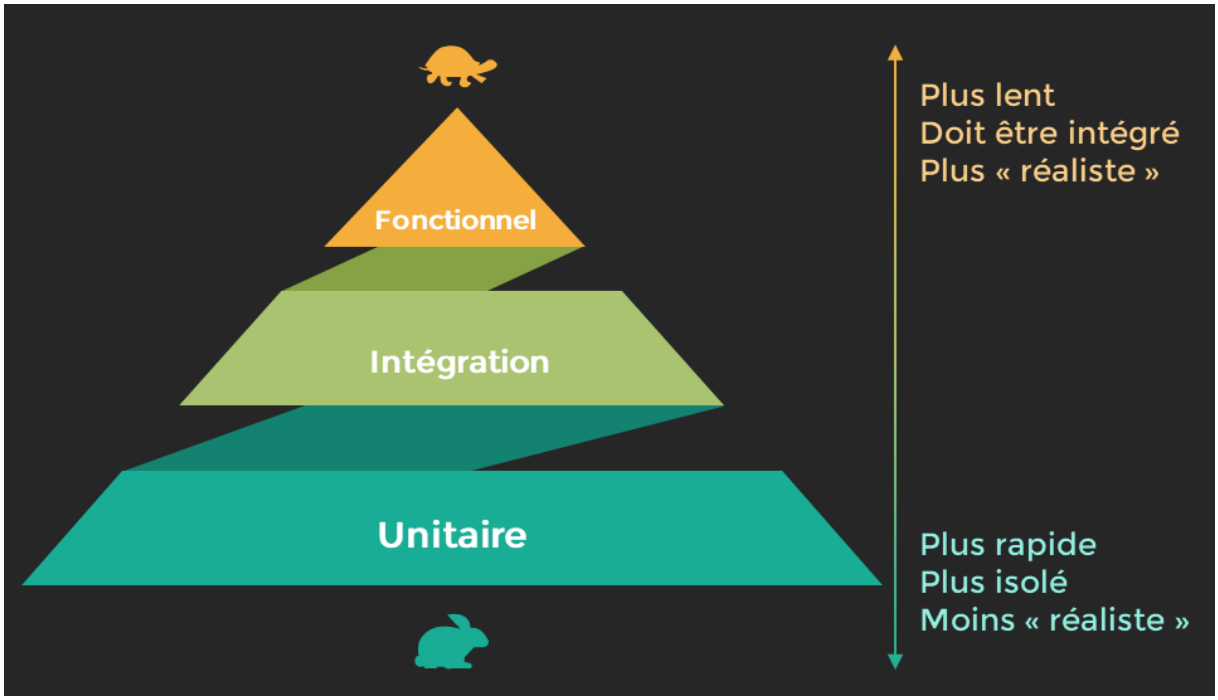
Stratégie de testing

Clément Jourdain

1. Description de la stratégie de testing.



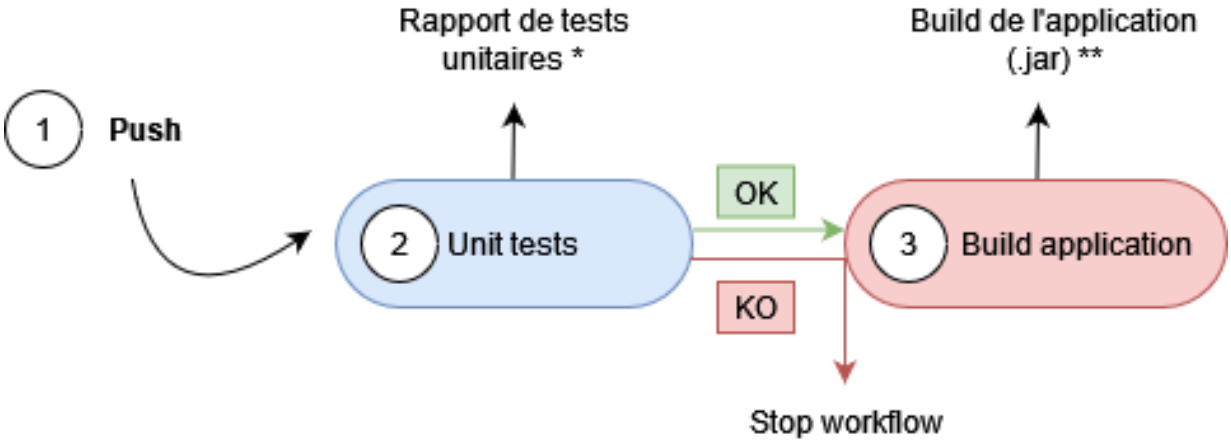
1- Les tests unitaires qui testent de "petites" unités de code. Plus précisément, ils testent que chaque fonctionnalité extraite de manière isolée se comporte comme attendu. De bons tests unitaires sont stables, c'est-à-dire que le code de ces tests n'a pas besoin d'être modifié, même si le code de l'application change pour des raisons purement techniques. Ils deviennent donc rentables, car ils ont été écrits une seule fois, mais exécutés de nombreuses fois.

2- Les tests d'intégration vérifient les interactions entre les unités. Ils peuvent nécessiter l'exécution de composants extérieurs (base de données, services web, etc.). Le lancement de ces composants et l'interaction avec les unités de code développées rendent ces types de test plus lents et potentiellement moins stables mais simulent des scénarios plus proches de l'utilisation finale de l'application.

3- Les tests fonctionnels (end-to-end), visent à simuler le comportement d'un utilisateur final (ou plusieurs) sur les fonctionnalités proposées par l'application. L'ensemble du code développé est pris comme une boîte noire à tester, sans connaissance des briques et des unités de code qui la compose. Les simulations obtenues sont donc les plus proches de l'application utilisée dans des conditions réelles. Ces tests nécessitent toute l'infrastructure nécessaire à l'application. Ils sont les plus lents à exécuter, et testent une partie beaucoup plus grande du code développé.

2. Description du pipeline.

De manière à implémenter la stratégie de testing énoncée plus haut la **plateforme** d'intégration continue **Github Actions** a été choisie. Cette dernière permet de créer des **workflows** qui sont trigger selon des événements spécifiques (push, merge request…). Un **workflow** est un **processus automatisé** qui exécute des jobs.



* Artifacts mis en ligne

** Artifacts mis en ligne seulement en cas de succès

1 - A chaque fois qu'un commit est effectué sur le repo le workflow est exécuté.

2 - Le premier job du workflow à être joué est celui concernant les tests unitaires. Une fois terminé ce dernier peut être en échec ou en succès. Dans le cas d'un succès un rapport de tests unitaires est mis en ligne et peut être téléchargé pour consulter les résultats et le job suivant est joué. En cas d'échec un rapport de tests unitaires est mis en ligne et peut être téléchargé pour consulter les résultats et le workflow ne se poursuit pas.

3 - Le deuxième job du workflow est celui du build de l'application. Dans le cas d'un succès le jar de l'application est mis en ligne et peut être téléchargé.

3. Tests d’acceptance.

Il est stipulé que la mise en œuvre pour le PoC sera réussie quand les différents points présents dans la partie « Déclaration d’hypothèse » du document « Hypothèse de développement d'une preuve de concept pour le sous-système d'intervention d'urgence en temps réel » seront atteints. Ces derniers ont été reportés et reformulés ci-dessous sous forme de tests d’acceptance grâce à la norme Gherkin.

1 - La première hypothèse est le temps moyen de traitement d’une urgence passe de 18,25 minutes (valeur actuelle) à 12,00 minutes (valeur souhaitée).

GIVEN : Un cas d’urgence

WHEN : Le processus d’urgence (allant de l’utilisation d’Urgency service jusqu’à l’acheminement à l’hôpital) est déclenché

THEN : Le temps nécessaire pour exécuter le processus est de 12 minutes.

Le PoC mis en place ne permet pas de tester le processus dans son entièreté car il ne se concentre que sur l’identification et la communication de l’hôpital le plus proche permettant de prendre en charge la pathologie renseignée.

2 - Plus de 90 % des cas d'urgence sont acheminés vers l'hôpital compétent le plus proche du réseau.

GIVEN : 100 cas d’urgence

WHEN : Le micro service Urgency service est interrogé

THEN : 90 de ces cas sont acheminés vers l’hôpital le plus compétent.

Le PoC ne permet pas de collecter de retour du personnel des urgences et donc ne permet pas de mesurer le pourcentage des cas d’urgence acheminés vers l’hôpital le plus proche.

3 - Que l’hôpital retourné aux personnes des urgences soit le plus proche et traite la pathologie renseignée.

GIVEN : Une position sous forme de latitude, longitude ainsi qu’une pathologie.

WHEN : Le micro service Urgency service est interrogé

GIVEN : l’hôpital le plus proche, capable de traiter la pathologie renseignée, est renvoyé.

Le PoC réponds à cette exigence. Elle peut être vérifié notamment grâce au test d’intégration implémenté sur le pipeline.

4 - La dernière hypothèse est que le temps de réponse de moins de 200 millisecondes avec une charge de travail allant jusqu'à 800 requêtes par seconde, par instance de service

GIVEN : Le micro service Urgency service

WHEN : 800 requêtes par secondes sont effectuées

THEN : le temps de réponse est de 200ms par requête