Nicolas Godard Y2

Compte-rendu individuel DEVOPS 4

Sommaire

I - TD 1

1. Qu'est-ce qu'une API ?
2. What is REST ?

II - TD 2

1. Tests unitaires vs Tests fonctionnels
2. Lien Tests - Mock
3. Métrique

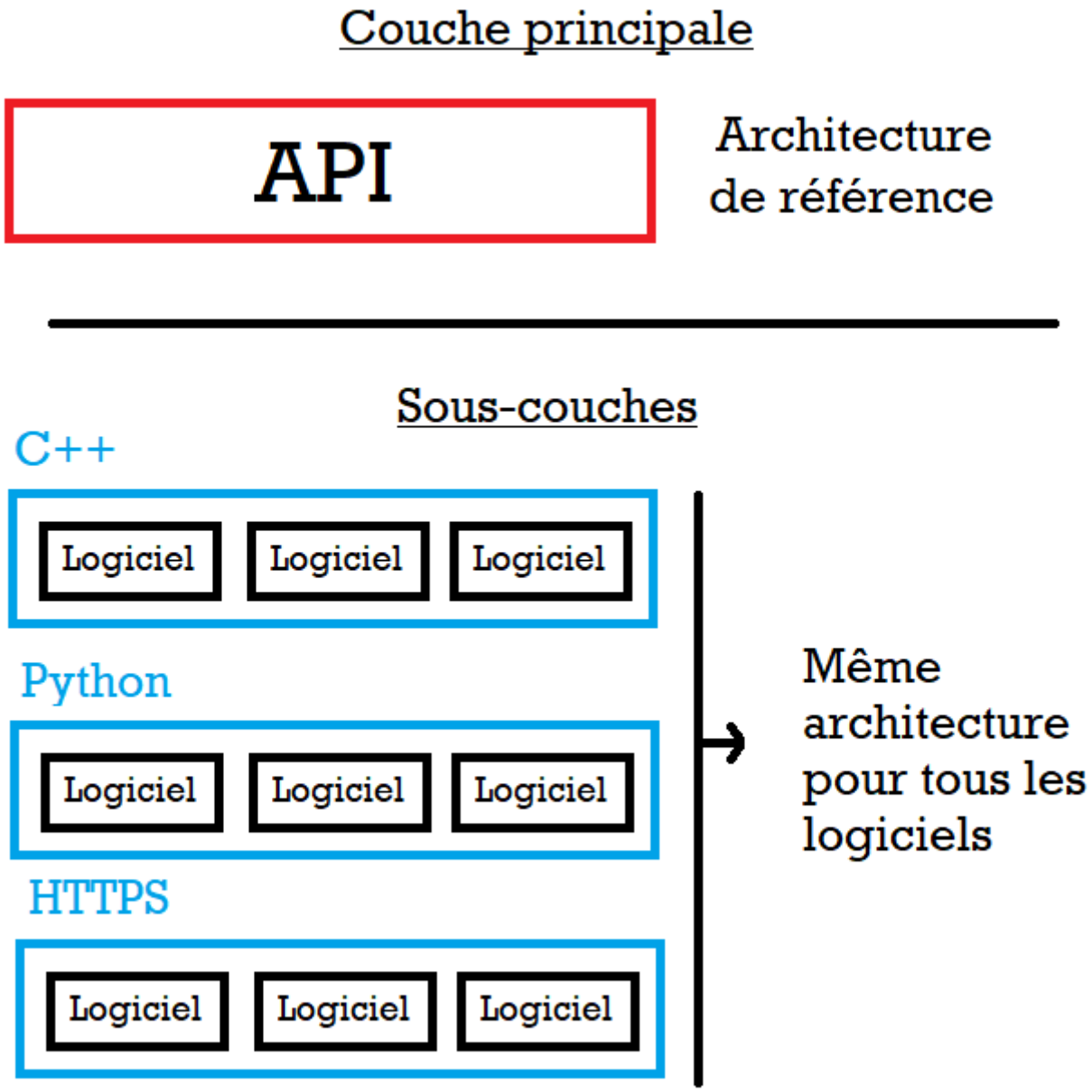
III - TD 3

1. Les Patchs

TD 1

Qu'est-ce qu'une API ?

Une API est une interface de programmation d'applications. C'est un ensemble de classes, de méthodes, de fonctions et de constantes qui servent de façade à un logiciel qui va offrir des services à d'autres logiciels.



WHAT IS REST ?!

REST (REpresentational State Transfer) est une architecture type pour des API. Le but est de faire communiquer des applications entre elles. Par exemple, si un client fait une requête http à un serveur, il aura une réponse (timeout, accept, ...).
Requête -> Réponse[en-tête;body]

6 principes :

1. Une interface uniformisée
2. Séparer les interfaces client-serveur (pour simplifier les pages)
3. Une compréhension claire (les requêtes du clients doivent être complétes)
4. Une réponse doit indiquer si elle est dans le cache (si c'est le cas, le client garde cette information plus tard pour réduire la latencec/chargements/débit du serveur si il refait la même requête au serveur)

5. Un système hiérarchisé (chaque du serveur couche n'a accès qu'à ses données, pas celles des autres couches)
6. Un code facilement accessible

Des Ressources
Toute information sur le serveur peut être une ressource (exemple : une image, un service temporaire, ...).

Ces ressources vont respecter un format : media type (toute ressource contient une adresse). L'API REST va pouvoir utiliser ces ressources pour créer des liens entre le client et le serveur.

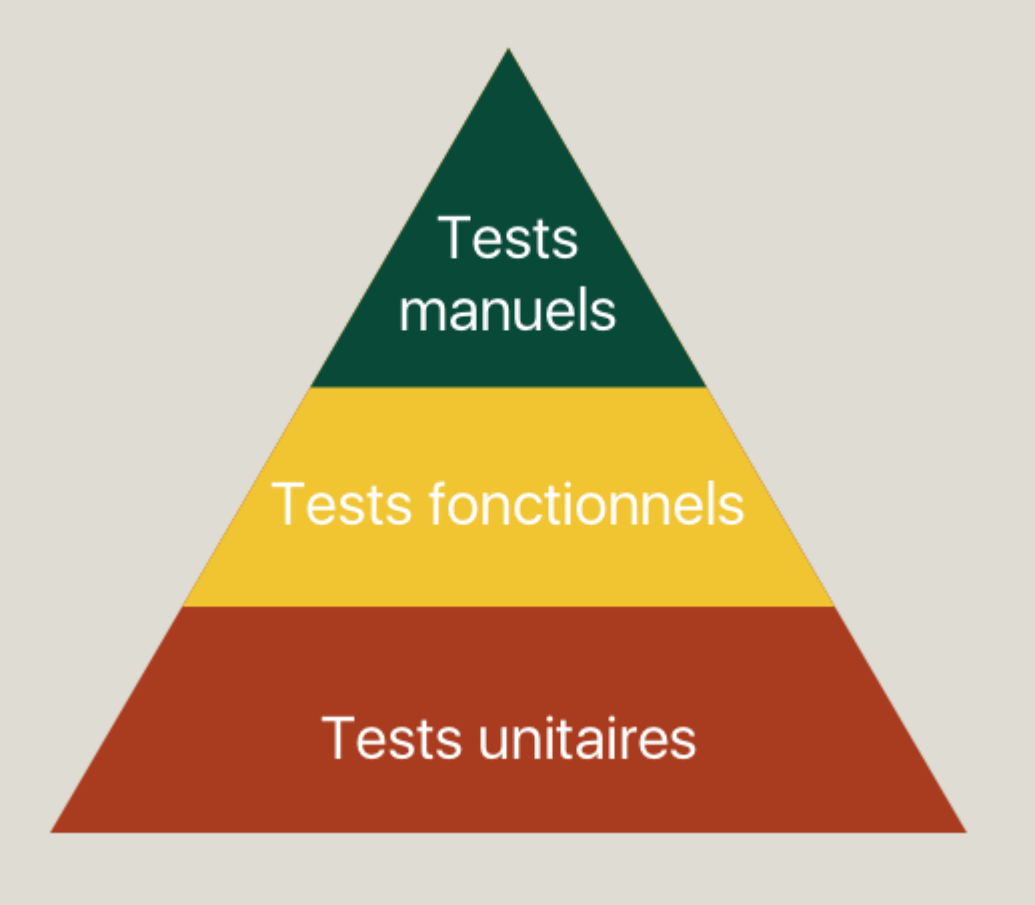
Il existe des "Resource Methods" (dans l'idée comme en HTTP à la différence que dans REST, n'importe quelle méthode peut être utilisée pour un service - /!\ REST != HTTP).

Conclusion TD 1

REST est une architecture. Les ressources sont découpées pour qu'elles soient accessible via plusieurs formats. Le serveur et le client communiques via une interface standardisée et des méthodes/protocols.

TD 2

Tests unitaires vs Tests fonctionnels



Pour faire simple, un test unitaire vérifie le bon fonctionnement d'une portion d'un programme alors qu'un test fonctionnel vérifie toutes les fonctionnalités de l'application.

Ces 2 types de tests ne sont pas utilisées dans un même ojectif. Le but des tests unitaires est de décomposer le code afin de voir si chaque partie du code fonctionne individuellement. Les tests fonctionnels quant à eux ont pour but de vérifier des scénarios (Un utilisateur se rend sur le logiciel, clique sur ...). Le code testé n'est donc pas le même.

Les tests unitaires doivent être faits avant la mise en production du logiciel alors que les tests fonctionnels sont lancés une fois que tout le code a été écrit et que tous ses tests unitaires ont été lancés.

Lien Tests - Mock

Comme nous l'avions vu en CPO, les mocks sont des simulations d'objets ayant le même comportement qu'un objet réel de manière contrôlée.

Lors d'un test unitaire, un mock va permettre de simuler un problème ou une situation difficile à reproduire manuellement (comme une erreur du réseau).

Métrique

Une métrique est un pourcentage sur le taux de couverture en tests unitaires de notre code. Cette valeure permet au programmeur de se faire un avis sur sa couverture de tests. Elle peut lui donner une forme de satisfaction/motivation en fonction du résultat.

Cependant, cette valeure n'est qu'un indicateur. Il n'y a pas de "bon pourcentage de code à couvrir". Évidemment, il ne faut pas qu'elle soit nulle mais l'objectif final doit être étudié à l'avance (une couverture de 100% n'est pas forcément utile et surtout ne veut pas dire que tout le code est couvert, certaines séquences peuvent être oubliées).

Toutefois, il semble qu'en moyenne la métrique visée soit entre 80% et 100%.

Conclusion TD 2

Les tests fonctionnels et unitaires sont différents. Cependant, ils sont tous les deux nécessaires afin d'avoir un code valide. Calculer la métrique est intéressant si le résultat souhaité a été établi au préalable.

TD 3

Les Patchs

Un patch (fichier .patch) est un fichier contenant du code formaté. Il permet de modifier du code existant en ajoutant ou modifiant des fonctionnalités. De plus, un patch va permettre d'ajouter le code en question dans la copie d'un fichier déjà existant plutôt que de directement remplacer ce fichier par un nouveau.

Nous nous sommes arrêtés ici