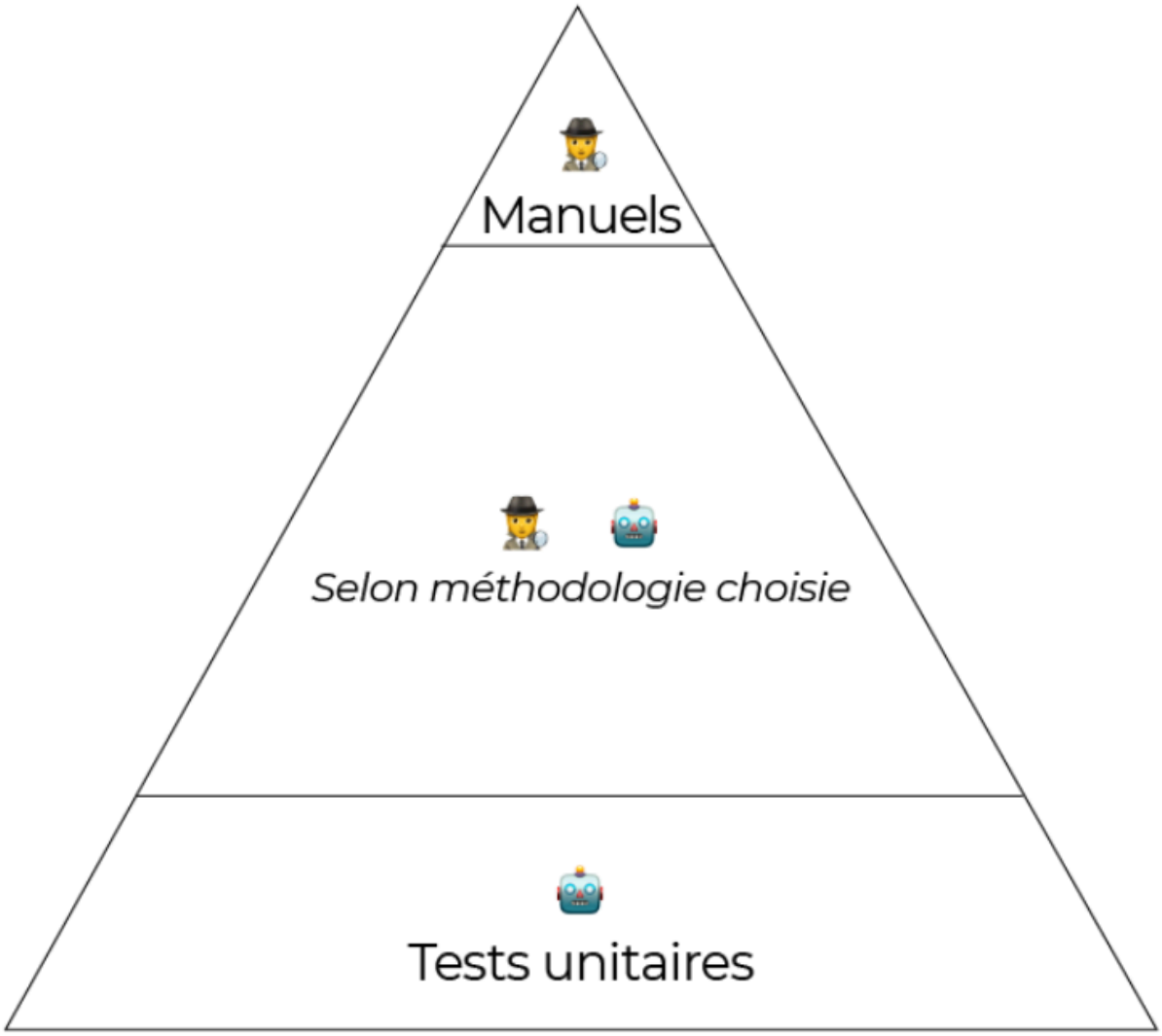
2_La pyramide des tests

Une pyramide de tests représente la somme des tests effectués sur un projet, ainsi que leur type.



Chaque étage de cette pyramide est constitué d'un type spécifique de test. Car, oui, il y a différents types de tests.

Les tests unitaires

Les tests unitaires sont la base de la pyramide de tests. Comme leur nom l'indique, les tests unitaires ont pour objectif de tester **une unité de code**. Il

peut s’agir d’une méthode ou d’une fonction, par exemple. Afin qu’un test unitaire soit considéré comme tel, il doit respecter une règle bien précise :

Un test unitaire ne peut pas être dépendant de fonctionnalités extérieures à l'unité qui est testée.

Par exemple, un test faisant appel à une base de données ou une API ou encore une autre méthode ne sera pas considéré comme un test unitaire.

Un test unitaire vérifie qu'**une fonction réalise l'action** souhaitée. Il ne s'agit pas ici de tester l'interaction entre les différentes fonctionnalités, mais plutôt une partie d'une fonctionnalité.

> **Dans le projet Crocoplus, un test unitaire pourrait valider que la fonction `nourrir_crocodile()` a pour effet de mettre à jour la date du dernier repas de notre objet crocodile.**

Les tests d'intégration

Juste au-dessus des tests unitaires dans notre pyramide, nous avons les tests d’intégration. Les tests d'intégration ont pour objectif de vérifier qu’une nouvelle fonctionnalité ne va pas poser de problème lors de son intégration au sein de l’application.

En effet, sur un projet nous allons développer de nombreuses fonctionnalités. Parfois, dans le cadre d’un travail en équipe, d’autres développeurs vont le faire également. Les tests d'intégration vont vérifier que **toutes** ces fonctionnalités arrivent à **travailler ensemble**.

Tout comme les tests unitaires, les tests d'intégration sont réalisés automatiquement par l'ordinateur. Cela les rend plus rapides à exécuter et plus fiables. Ils **correspondent aux spécifications techniques**, et testent par conséquent **l'architecture** du programme.

> **Dans le projet Crocoplus, par exemple, nous pourrions mettre en place un système de repas automatique. Notre méthode nommée `servir_repas()` se chargerait de spécifier que tous les crocodiles ont eu leur repas en faisant appel à la méthode `nourrir_crocodile()`. Nous souhaitons tester que si `servir_repas()` est exécutée, la date de dernier repas des crocodiles sera mise à jour. Nous allons donc tester que le résultat est celui attendu, en faisant appel à la méthode tierce `nourrir crocodile()`. Notre test dépendant d'une autre fonction que celle testée, il s'agit d'un test d'intégration.**

Tests fonctionnels

Les tests fonctionnels vont vérifier qu'une fonctionnalité, dans son ensemble, marche comme nous le souhaitons du point de vue de l'utilisateur. Ils sont, le plus souvent, réalisés par l'ordinateur compte tenu de la rapidité d'exécution, mais ils peuvent également être réalisés par un humain. Ils reprennent un **parcours utilisateur**.

Pour cela, il existe des librairies comme **Selenium** pour les applications web, capables d'ouvrir un navigateur et de se charger de simuler les actions d'un utilisateur.

Généralement, un test fonctionnel est **associé à une User Story** et teste son bon déroulement avant l'étape de vérification manuelle.

> **Dans le projet Crocoplus, un de ces tests pourrait être la validation du processus d'ajout d'un crocodile. Le test va alors vérifier qu'un utilisateur peut se connecter, accéder à la page d'ajout d'un crocodile, l'ajouter, puis voir une page de confirmation.**

Tests de performance

Les tests de performance évaluent la vitesse, la réactivité et la stabilité d'une application soumise à une charge de travail. Ces tests sont effectués afin d'identifier les goulots d’étranglement dans le code, liés aux performances.

> **Dans le projet Crocoplus, le test va simuler des requêtes sur les fonctionnalités de l’application afin de mesurer les temps de réponse. Si les métriques sont inférieures aux seuils de la spécification, la fonctionnalité peut être validée.**

Tests d'acceptation

Un test d'acceptation est réalisé par un humain, généralement le client ou son représentant, et valide que la fonctionnalité développée correspond bien à celle qui était attendue.

Le client donne ses retours ou **valide la fonctionnalité**.

> **Dans le projet Crocoplus, le client va lui-même ajouter un crocodile et faire ses retours. Si la fonctionnalité lui convient, elle est marquée comme acceptée.**

La phase de recette

Elle est réalisée en collaboration étroite avec le client, et valide que le logiciel livré correspond bien à ses attentes. Durant cette phase, l'équipe responsable du projet teste les différentes composantes du projet et décèle les derniers bugs avant la mise en ligne.

La phase de recette est courante dans un projet séquentiel (cycle en V), mais pas dans le cas de méthodologies agiles. En effet, les méthodologies agiles préconisent des livraisons rapides, les tests d'acceptation font office de recette.

> **Dans le projet Crocoplus, un de ces tests pourrait être la validation du processus d'ajout d'un crocodile. Le test va alors vérifier qu'un utilisateur peut se connecter, accéder à la page d'ajout d'un crocodile, l'ajouter, puis voir une page de confirmation.**

Plusieurs formes de tests

Il existe bien d'autres formes de tests !

Plusieurs autres aspects d'un projet peuvent être testés :

- **l'accessibilité** : respect des normes du W3C, accessibilité aux personnes en situation de handicap… ;
- **la sécurité** : certificat SSL, formulaires sécurisés, base de données cryptée...

En résumé

- Il existe deux grandes formes de tests, les tests **automatisés** et les tests **manuels**.
- L'ensemble des tests est découpé en 4 types de tests :
  - les tests **unitaires** testent une unité de code ;
  - les tests **d'intégration** testent une fonctionnalité ;
  - les tests **fonctionnels** testent le parcours utilisateur ;
  - les tests de **performance** testent la réactivité et la stabilité d'une fonctionnalité ;

- les tests **d’acceptation** testent la conformité à la spécification.

Maintenant que nous en savons un peu plus sur les différents tests, il est temps de voir comment les mettre en place, en commençant par les tests unitaires.