## Tests Fonctionnels

Objectif : Vérifier la conformité d’un logiciel avec ses spécifications (cahier des charges et analyse formelle).

Ils mettent en œuvre des scénarios qui permettent de tester les tâches-métiers pour lesquelles le logiciel a été développé (Permettent de valider l’exécution de la tâche métier). Ils mettent l’utilisateur en situation de travail et son souvent développés en technique de « boite noire ».

Chaque test valide une partie du logiciel (soit une action isolée, soit le scénario d’un cas d’utilisation, soit un enchainement de cas (tel un scénario d’usage)).

## Couverture fonctionnelle

Chaque fonctionnalité métier de l’application doit être vérifiée par au moins un cas de test (Il s’agit d’un ensemble d’entrées, de conditions d’exécutions et de résultats attendus, ces derniers sont développés dans un objectif particulier tel que le cheminement particulier d’un programme ou la vérification de la conformité à une exigence spécifique, on essaiera de prendre en premier lieu les chemins pris par un utilisateur).

Chaque fonctionnalité métier se verra vérifiée par un cas de test.

## Approche

On distingue deux types d’approches :

* Une approche manuelle, centrée sur l’IHM : il s’agit d’une approche intuitive permettant de reproduire le comportement de l’utilisateur final. Elle permet donc de tester les cas d’utilisation d’une application.
* Une approche automatisée par programmation : celle-ci demande des capacités que n’ont souvent pas les clients finaux, elle est surtout utilisée par les développeurs, l’exemple le plus fréquent d’utilisation est lors de l’utilisation de la méthode Agile afin d’effectuer des tests à chaque fin d’itération.

## Prévoir les tests

Il faut obligatoirement prévoir les tests avant de les exécuter. A cette fin, toute description d’un composant est accompagnée de tests qui permettront de s’assurer qu’il correspond à sa description.

## Faire évoluer les tests

Les tests fonctionnels doivent évoluer en fonction des modifications apportées au projet. Mais il faut à tout prix garder les données qui ont permis de reproduire un bug ainsi que son scénario de test et les intégrer dans le scénario global de tests de l’application. Ceci permet de tester la non-réapparition du bug.

## Rédiger les tests

Étant donné que les tests pourraient être exécutés de nombreuses fois durant la durée de vie du logiciel, il faut prévoir que le document puisse être rempli, commenté, daté et signé à chaque exécution, par ailleurs son remplissage se passe à chaque exécution de tests. Il y aura également plusieurs documents de tests différents.

## Document de tests

Ils permettent de décrire l’environnement, de faire référence aux cas d’utilisation, de lister les prérequis, de décrire des données en input, de décrire le scénario du test et de décrire les données d’output.

## Scénario de test

Un scénario comprend plusieurs étapes :

* Cas de test le plus général dans lequel tout se passe bien.
* Toutes les autres combinaisons possibles dans lequel tout se passe bien.
* Cas d’erreurs (Il existe plusieurs cas d’erreurs possibles :
  + Tests des erreurs isolées : saisie de données erronées, absence de valeurs)
  + Tests des cas extrêmes : fourchette avec un min plus grand que le max, valeur max, valeur max+1, valeurs nulles, formats invalides, date trop ancienne ou dans le futur, nombre trop grand, nombre de caractères plus grand que celui prévu en Db.
  + Tests des combinaisons d’erreur
  + Tests de données erronées dans un input
  + Tests de recherche : aucune donnée ne correspond aux critères, trop de données correspondent aux critères

## Formalisme

On ne fera pas de copie d’écran (cela deviendrait vite non maintenable). On va plutôt faire une description de l’environnement, faire référence aux cas d’utilisation, effectuer une liste des prérequis, décrire les données en input, décrire le scénario de test et décrire les données en output.