TESTS LOGICIELS ET SYSTEMES

**Qualité d’un testeur**

* Être testeur, c’est un état d’esprit.
* Faites preuve de curiosité, rigueur, autonomie, proactivité et polyvalence.
* Ayez de bonnes capacités de communication.
* Vous n’avez peut-être pas toutes ces compétences actuellement, mais vous les développerez au fur et à mesure de vos expériences !

Les qualités d’un testeur est :

* La rigueur et la curiosité.

- Les étapes durant le test logiciel

 **Lisez-le attentivement** et assurez-vous de comprendre les étapes décrites pour vous assurer de savoir quoi faire et comment le faire.

 **Vérifiez que tout est clair** car ce n’est pas forcément la même personne qui écrit le cahier de recette et qui réalise les tests. N’hésitez pas à poser vos questions pour soulever toute incompréhension.

 **Tenez compte des résultats attendus** : le cahier de recette devrait décrire les résultats attendus du test. Assurez-vous de comprendre ces résultats attendus et de savoir comment les vérifier.

 **Prenez connaissance des tests** à effectuer pour savoir jusqu’à quel niveau descendre. Par exemple, est-ce que ce sont des tests simples de l’Interface Homme-Machine (IHM, Front Web ou Front-End) ou est-ce qu’il y a des tests de cas aux limites avec un jeu de données ou une configuration particulière à mettre en place.

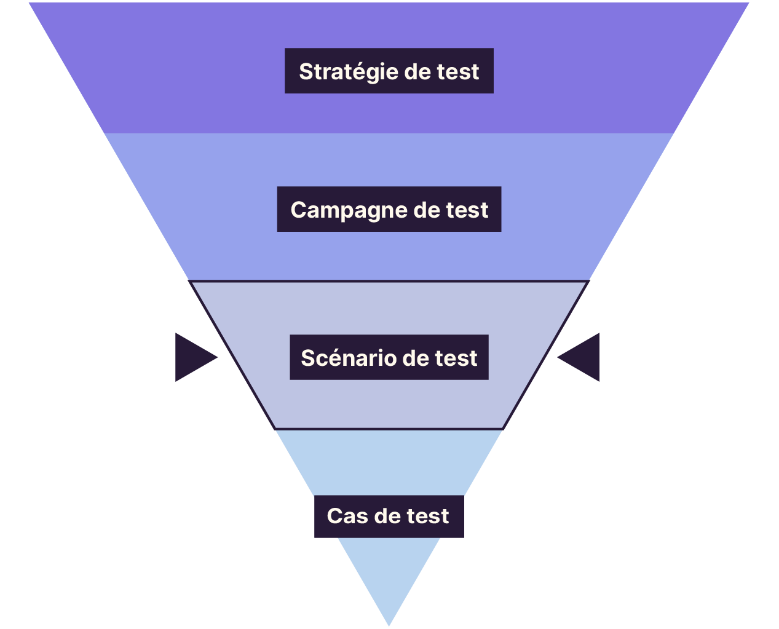
Les cas aux limites sont des tests qui vont pousser le logiciel dans ses retranchements. Il peut s’agir de saisir de caractères spéciaux dans les champs de saisie ou de vérifier ce qui se passe quand on saisit 1 000 caractères.

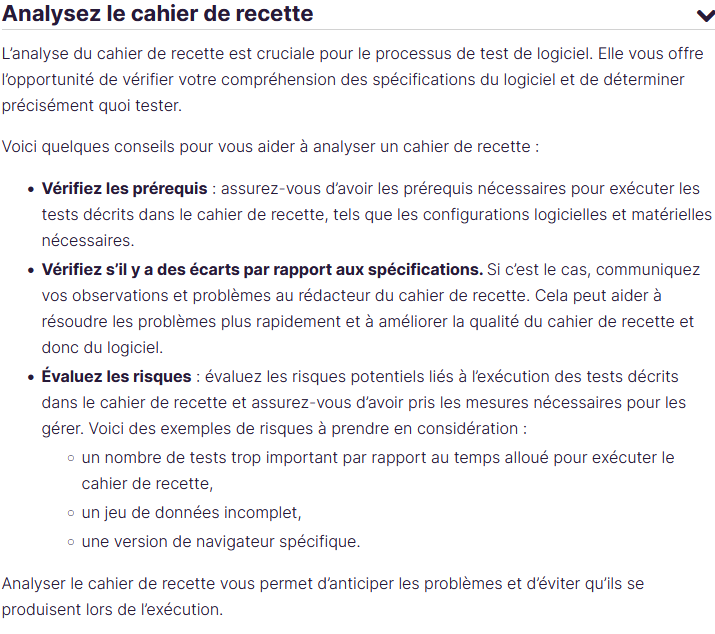
Les cas aux limites sont des tests qui vont pousser le logiciel dans ses retranchements. Il peut s’agir de saisir de caractères spéciaux dans les champs de saisie ou de vérifier ce qui se passe quand on saisit 1 000 caractères.

**Les scénarios de test** sont des étapes détaillées décrivant les actions à effectuer pour tester une fonctionnalité particulière d’un logiciel. Ils sont souvent écrits sous forme de cas d’utilisation et décrivent les étapes nécessaires pour vérifier que le logiciel fonctionne comme prévu ou vérifier le fonctionnement correct d’une fonctionnalité.

Dans le processus de test logiciel, **les cas de test** sont les éléments de base utilisés pour tester les **exigences spécifiques** du logiciel, tandis que **les scénarios de test** sont créés pour tester des **parcours utilisateur complets** dans le système.

**Les scénarios de test** sont conçus à partir d’une campagne de test, elle-même élaborée en fonction d’une stratégie de test. Mais, pour cette partie, ne vous inquiétez pas, vous verrez ça dans la suite du cours.





Préparation aux tests logiciel

* vérifiez que vous avez accès au logiciel à tester ;
* vérifiez les navigateurs web sur lesquels le logiciel doit être fonctionnel ;
* vérifiez sur quels OS (Windows, Linux, macOS) le logiciel doit fonctionner ;
* vérifiez sur quel appareil le site web doit être accessible et utilisable (smartphones, tablettes, PC, Mac, etc.) ;
* assurez-vous d’avoir les données de test : un jeu de test doit pouvoir être rejoué dans les mêmes conditions (pour vérifier qu’on obtient bien les mêmes résultats) ou dans d’autres (pour pouvoir analyser le delta entre deux runs de test).

Les jeux de données sont les données qui seront nécessaires pour le processus de test. Ce sont des données représentatives de la réalité pour une application donnée. Par exemple, ce peut être un identifiant client sur un site e-commerce ou une référence d’article.

**Une régression** est un bug découvert sur une fonctionnalité qui fonctionnait correctement auparavant.

Les test de non-régression sont des tests permettant de vérifier qu’une modification ou la correction d’une fonctionnalité n’a pas apporté de **régression** dans le logiciel.

**Le cahier de recette** a un rôle déterminant car il va **lever toute ambiguïté** sur votre phase de test, de sa réalisation à ses résultats attendus.

**Les tests exploratoires** permettent d’aller au-delà des tests du cahier de recette. Ils correspondent à une approche de test de logiciel qui consiste à explorer le système de manière informelle et à identifier les problèmes et les anomalies.

Les tests exploratoires permettent de :

* **trouver des défauts** qui ne pourraient pas être découverts par des tests formels et améliorer l’efficacité des tests planifiés : en tant que testeur, avec l’expérience, vous arriverez à utiliser votre temps de manière plus efficace en vous concentrant sur les domaines du logiciel qui sont les plus susceptibles de poser des problèmes ;
* **identifier des problèmes inattendus** : en explorant le logiciel de manière flexible, vous pouvez trouver des problèmes qui n’ont pas été identifiés par les tests planifiés. Les tests exploratoires peuvent être particulièrement utiles pour identifier des problèmes de qualité qui pourraient être difficiles à identifier par des méthodes de test plus conventionnelles ;
* **améliorer l’utilisabilité et la stabilité** du site ou de l’application ;
* **trouver de nouvelles idées**, fonctionnalités et les ajouter au backlog ;
* **développer votre créativité** dans les tests : les tests exploratoires peuvent vous encourager à développer votre créativité et à essayer des approches différentes pour tester le logiciel. Cela peut conduire à une plus grande variété de tests, qui peuvent identifier des problèmes que des tests plus conventionnels n’auraient pas identifiés ;
* **trouver des anomalies sur les exigences et la conception**, car ces tests ne sont pas basés sur les documents (cahier de recette ou spécifications) ;
* **s’adapter à tous les cycles de vie**. Ils ne sont pas réservés seulement à l’Agile ;
* **éviter le paradoxe du pesticide** (car deux campagnes de test exploratoires sont toujours différentes) ;
* **diminuer fortement le temps alloué à l’écriture** et à la conception des tests ;
* exécuter un plus grand nombre de tests.

**Les chartes de test** sont des documents qui définissent l’objectif, la portée et les limites d’un test exploratoire. Elles sont utilisées pour structurer et orienter les sessions de tests exploratoires et pour s’assurer que les testeurs se concentrent sur des aspects particuliers du système à tester.

**Une heuristique de test** est une méthode de résolution de problèmes qui utilise une liste de principes, de règles ou de directives pour guider les tests. Les heuristiques de test sont des outils utiles pour les testeurs car elles aident à identifier les problèmes qui pourraient être manqués par une approche plus traditionnelle. Elles sont souvent utilisées dans les tests exploratoires pour guider les sessions de test.

* **Heuristique de comparaison** : comparer les résultats obtenus avec les résultats attendus pour s’assurer que les fonctions du logiciel sont correctes.
* **Heuristique de configuration** : tester différentes configurations système pour s’assurer que le logiciel fonctionne correctement dans toutes les situations.
* **Heuristique de détection** : chercher des signes d’erreurs ou de défauts dans le comportement du logiciel pour trouver les bugs.
* **Heuristique de diversité** : tester une grande variété de scénarios pour s’assurer que le logiciel fonctionne correctement dans toutes les situations possibles.
* **Heuristique de l’expérience utilisateur** : tester le logiciel du point de vue de l’utilisateur pour s’assurer qu’il est convivial et facile à utiliser.
* **Heuristique de la fiabilité** : tester le logiciel pour s’assurer qu’il est fiable et qu’il ne tombe pas en panne.
* **Heuristique de la performance** : tester les performances du logiciel pour s’assurer qu’il est réactif et qu’il ne ralentit pas avec l’utilisation.
* **Heuristique de la sécurité** : tester le logiciel pour s’assurer qu’il est sécurisé et qu’il ne présente pas de risques pour les utilisateurs.

**Les limites de test exploratoires**

 ils sont très fortement **dépendants du testeur** : deux testeurs sur une même session et avec la même charte de test ne testeront pas de la même façon et auront des résultats différents ;

 de même, deux campagnes de tests exploratoires ne sont **jamais identiques** (même si c’est la même personne qui exécute les campagnes) ;

 ils ne sont **pas aussi rigoureux et structurés** que les tests fonctionnels ;

 ils sont souvent exécutés de manière informelle et **ne couvrent pas toutes les fonctionnalités** du système de manière équilibrée ;

 ils ne sont pas automatisables ;

 **ils ne sont pas suffisants**. Ils viennent **en complément** de l’ensemble des outils déjà à disposition des testeurs.

**Jira** est un logiciel de gestion de projet très populaire et polyvalent utilisé dans le développement de logiciels. Il est largement utilisé pour suivre et gérer les tâches, les problèmes et les projets.

La criticité d’une anomalie permet de prioriser les tickets pour que les développeurs s’occupent des anomalies les plus critiques en premier.

* les criticités les plus communes :
* **Mineure** : correspond à un bug peu gênant pour l’application et ses utilisateurs et facilement contournable.
* **Majeure** : correspond à un bug plus important, détériorant significativement l’utilisation de l’application sans la bloquer complètement.
* **Critique** : correspond à un bug bloquant totalement l’utilisation de l’application ou d’une de ses fonctionnalités importantes.

**Document à analyser**

**rassembler des informations provenant de différentes sources**, telles que :

* les tests du cahier de recette ;
* les tests exploratoires ;
* les résultats des tests ;
* les anomalies créées dans Jira ;
* les fonctionnalités testées.

**vous devez vous assurer que les données sont précises et complètes**.

es KPI peuvent inclure le taux de couverture des tests, le taux d’anomalies détectées, le temps de test, etc. Enfin, il est essentiel de se concentrer sur la qualité des données en vérifiant leur exactitude et leur exhaustivité avant de commencer l’analyse.

Voici quelques exemples d’objectifs pour l’analyse de données de test :

* Identifier les domaines qui ont besoin d’amélioration.
* Trouver les causes profondes d’un problème courant.
* Évaluer l’efficacité d’une nouvelle fonctionnalité.
* Optimiser le temps de test.

De plus, voici des exemples d’indicateurs-clés de performance (KPI) que l’on peut utiliser pour l’analyse des données de test :

* Le nombre de bugs par module ou fonctionnalité : cela permet d’identifier les parties du logiciel qui sont les plus sujettes aux erreurs.
* Le taux de résolution des anomalies : cela permet de mesurer l’efficacité des équipes à résoudre les problèmes détectés. Cela concerne les équipes de test et de développement.
* Le temps moyen de détection et de résolution des anomalies : cela permet de mesurer l’efficacité des processus de test et de développement, ainsi que la qualité générale du code.
* Le taux de régression : cela permet de mesurer la capacité du logiciel à conserver son bon fonctionnement au fil des mises à jour et des modifications.
* Le taux de couverture des tests automatisés : cela permet de mesurer l’efficacité des tests automatisés et leur capacité à détecter les erreurs.
* Le taux d’utilisation de l’application : cela permet de mesurer l’adoption et la satisfaction des utilisateurs et de déterminer les fonctionnalités les plus utilisées ou les plus ignorées.
* Le taux d’abandon de l’application : cela permet de mesurer le niveau de frustration des utilisateurs et d’identifier les problèmes d’ergonomie ou de performance.

L’analyse des résultats des tests est une étape cruciale dans le processus de test logiciel, car elle permet de déterminer la **qualité et la stabilité** du logiciel testé.

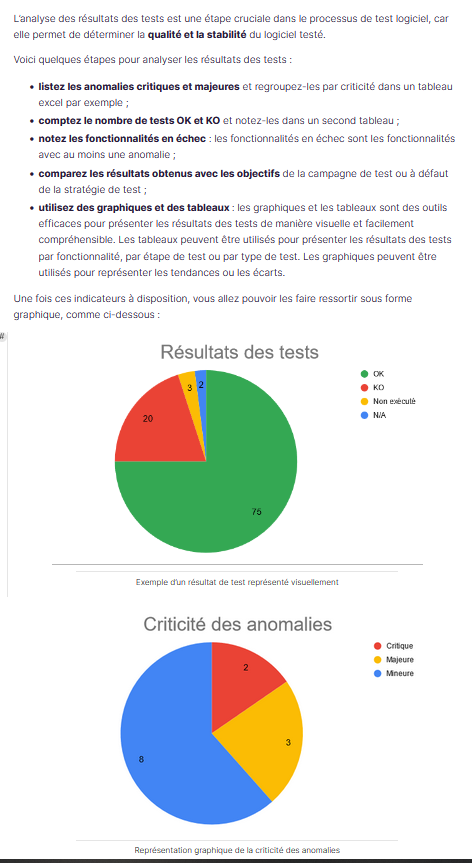
* Etape de rassemblement

Maintenant que vous avez déroulé votre mini campagne de test, vous allez analyser les données.

Dans un premier temps, vous allez rassembler les données. Voici une liste sur laquelle vous baser :

* les tests du cahier de recette ;
* les résultats des tests ;
* les anomalies créées dans Jira ;
* les fonctionnalités testées.

Appuyez-vous sur le cours pour vérifier les données et en faire l’analyse.



Maintenant que vous avez déroulé votre mini campagne de test, vous allez analyser les données.

Dans un premier temps, vous allez rassembler les données. Voici une liste sur laquelle vous baser :

* les tests du cahier de recette ;
* les résultats des tests ;
* les anomalies créées dans Jira ;
* les fonctionnalités testées.

Appuyez-vous sur le cours pour vérifier les données et en faire l’analyse.

Bonne pratique d un testeur

 Réunissez vos données de campagne de test.

 Assurez-vous qu’elles soient valides.

 Définissez les objectifs de l’analyse.

 Analysez les données.

**Le rapport de Bilan d’un testeur**

Ce rapport résume et synthétise les résultats de la campagne de test et inclut éventuellement des recommandations pour l’amélioration future :

* **Rédigez un sommaire (optionnel)** : vous pourrez rédiger un sommaire si votre bilan de campagne est assez conséquent sinon ce n’est pas nécessaire.
* **Décrivez les objectifs de la campagne de test** : reprenez les objectifs de la campagne de test ou de la stratégie de test pour les inclure dans votre bilan. Ainsi, il est plus facile de voir si les objectifs sont atteints en ayant tout au même endroit (objectifs et résultats).
* **Présentez les résultats de la campagne de test** : présentez sous forme synthétique et claire les résultats de la campagne de test comme vu au chapitre précédent.
* **Faites des recommandations (optionnel)** : vous pouvez faire des recommandations sur la façon de traiter les anomalies détectées et d’améliorer le processus de test à l’avenir. Ces recommandations peuvent inclure des actions correctives à court terme pour résoudre les problèmes actuels, ainsi que des améliorations à long terme pour prévenir les problèmes futurs.
* **Concluez le rapport** : faites un bref résumé.
* **Vérifiez et revoyez le rapport** : vérifiez que vous n’avez rien oublié et que tout est compréhensible.

**Présentation des resultats aux publiques**

* **Fournissez une vue d’ensemble** : avant de plonger dans les détails, il est important de fournir une vue d’ensemble des résultats des tests. Cela permettra à l’équipe de comprendre rapidement les points-clés de la campagne de test, tels que le taux d’anomalies détectées, le taux de couverture des tests, le temps de test, etc.
* **4. Mettez en évidence les points-clés** : lors de la présentation des résultats, il est important de mettre en évidence les points-clés qui aideront l’équipe à prendre des décisions. Par exemple, si une fonctionnalité particulière a connu un grand nombre d’échecs, cela doit être clairement mis en évidence afin que des mesures correctives puissent être prises.
* **5. Fournissez des détails supplémentaires si nécessaire** : si les membres de l’équipe ont besoin de plus de détails, il est important de les fournir. Cependant, ne vous perdez pas dans les détails et restez concentré sur les points-clés.
* **6. Proposez des actions correctives** : enfin, il est intéressant de proposer des actions correctives pour les anomalies détectées. Cela permettra à l’équipe de prendre des mesures concrètes pour améliorer la qualité du logiciel.
* En résumé, une présentation efficace des résultats d’une campagne de test doit fournir une vue d’ensemble claire et concise, mettre en évidence les points-clés, fournir des détails supplémentaires si nécessaire, et proposer des actions correctives pour les anomalies détectées.

**Mettre à jour les test existant**

* **Pour suivre l’évolution des exigences**
* **Pour améliorer la couverture des tests**

**Créér des nouveau tests**

* **Nouvelles fonctionnalités**
* **Correction de bugs**
* **Changements de conception**
* **Évolution des besoins du client**
* **Découverte de nouveaux comportements à partir des tests exploratoires**

**Supprimez les tests obsolètes**

* **Les tests sont obsolètes**
* **Les tests ne sont pas faisables**
* **Les tests sont redondants**

En test logiciel, **un faux positif** est un résultat de test qui est signalé comme une anomalie ou un bug alors qu’en réalité, le comportement du logiciel est correct. Autrement dit, le test a échoué alors que le logiciel est en fait fonctionnel. Les faux positifs peuvent être causés par plusieurs facteurs tels qu’une erreur dans le script de test, un environnement de test inapproprié, des données de test incorrectes, etc. Les faux positifs sont importants à identifier et à corriger, car ils peuvent entraîner une perte de temps et de ressources dans le processus de développement et de test.

* Il est important de maintenir le cahier de recette à jour : nouvelles fonctionnalités, nouveaux tests, tests obsolètes.
* Suivez les bonnes pratiques pour assurer l’efficacité des activités de test.

### Adoptez les bonnes pratiques de maintenance du cahier de recette

* **Mettez à jour régulièrement** : Il est important de mettre à jour le cahier de recette régulièrement pour s’assurer que toutes les nouvelles fonctionnalités et modifications ont été ajoutées.
* **Nettoyez les tests obsolètes** : Supprimer les tests obsolètes peut aider à maintenir la clarté et la pertinence du cahier de recette.
* **Évitez les duplications** : Éviter les duplications de tests similaires peut réduire la complexité du cahier de recette.
* **Gérez le versionning** : Il est important de conserver une trace de l’historique des modifications apportées au cahier de recette, de manière à pouvoir identifier facilement les changements et à revenir à une version antérieure si nécessaire. Cela permet également de mieux gérer les conflits et les erreurs éventuels lors de la collaboration entre différents membres de l’équipe de test. L’utilisation d’un système de contrôle de version, tel que Git, peut faciliter la gestion de version du cahier de recette.
* **Conservez un format identique entre les anciens et nouveaux tests** : Il est important d’avoir une certaine cohérence dans le format d’écriture des tests, même s’ils sont rédigés par des testeurs différents. Cela permet de faciliter la relecture et la compréhension des tests par l’ensemble des parties prenantes impliquées dans le processus de test.

En suivant ces bonnes pratiques, il sera plus facile de maintenir la qualité des tests tout au long du cycle de vie du projet, et de réduire les risques d’erreurs ou d’incohérences dans la documentation de test.

* **Information pratique pour un testeur**

Le code source et l’analyse du code source ne vous serviront pas comme source d’information pour l’analyse de la campagne de test. Ils ne sont pas pertinents.

Par contre les trois éléments suivants sont des éléments importants pour votre analyse :

* le cahier de recette, qui montre les tests effectués et le résultat de ces tests ;
* les spécifications, pour vérifier que les tests ont validé que les spécifications étaient respectées ;
* les anomalies, pour montrer les défauts du produit.

D’après vous, le nom de chaque testeur qui a relevé des anomalies devrait-il faire partie du bilan de campagne de test ?

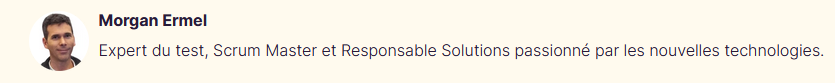
Le nom des testeurs n’est pas pertinent dans un bilan de campagne de test. Peu importe quel testeur a exécuté tel test, ça ne devrait pas influer sur le résultat.

Le cas de test 3 est à revoir car il indiquait de vérifier que la fiche produit contenait :

* la référence du produit ;
* le nom commercial du produit.

Il n’y a pas d’impact sur les autres cas de test qui concernent d’autres fonctionnalités.

Les tableaux et graphiques sont très visuels et permettent de voir rapidement et facilement les résultats.  
Une description textuelle, même agrémentée de listes à puces, prend plus de temps à appréhender.



**Votre professeur pour ce cours : Francis Bock**

Francis est Directeur Adjoint d'un projet européen qui produit des services géospatiaux. Il a enseigné le développement backend à l'Université Gustave Eiffel et il a été DSI dans l'administration.



Ecrire la documentation du projet : certification

Un SI est un environnement bien plus complexe. En effet, il faut le voir comme un ensemble de ressources, à la fois **humaines, matérielles** et **immatérielles**, dont le rôle est de **collecter, stocker, traiter** et **distribuer** de l’information.