Les différents types de test

Les tests logiciels sont un élément essentiel lors du développement logiciel qui garantit la qualité, la fiabilité et la fonctionnalité des applications logicielles. Et avec des ingénieurs de test de logiciels gagnant des salaires supérieurs à 95 000 $, il est indéniable que c'est une compétence précieuse à posséder.

Tout de même, si vous vous posez la question, quels sont les différents types de tests de logiciels, nous avons ce qu'il vous faut. Dans cet article, nous allons plonger dans les différents types de tests de logiciels que vous devez connaître, y compris les définitions et l'utilisation.

**Les principes de base des tests de logiciels**

* Les tests dépendent du contexte : différentes applications ont des exigences et des contraintes différentes, votre test doit donc être adapté au contexte.
* Sophisme d'absence d'erreurs : L'absence d'erreurs ne garantit pas l'absence de défauts dans une application logicielle. Les tests logiciels doivent être conçus pour couvrir les bogues identifiés et potentiels.
* Tests précoces : cela aide à détecter les erreurs tôt, réduisant ainsi le coût et les efforts nécessaires pour les corriger plus tard.
* Paradoxe des pesticides : la répétition des mêmes tests plusieurs fois peut entraîner la découverte des mêmes erreurs tout en passant à côté de nouveaux problèmes. Les tests doivent être révisés régulièrement pour inclure de nouveaux cas.
* Regroupement de défauts : un petit nombre de modules dans une application logicielle sont souvent responsables de la majorité des défauts, les tests doivent donc se concentrer sur leur identification et leur test approfondi.
* Des tests exhaustifs sont impossibles : en raison du nombre souvent illimité de scénarios de test, il est impossible de tester chaque combinaison d'entrées et de conditions.

**Tests manuels vs automatisés**

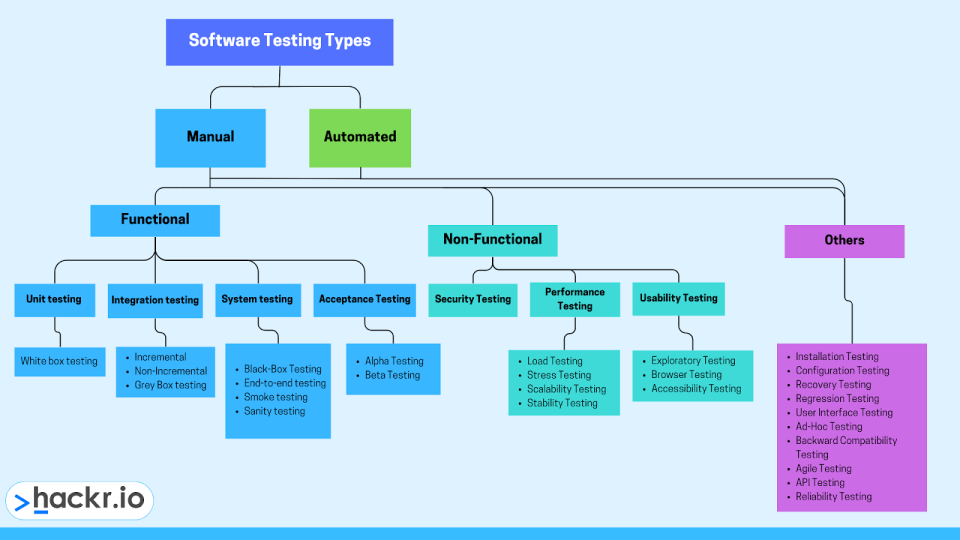
En ce qui concerne les tests de logiciels et les types de tests, la distinction la plus fondamentale est entre les tests manuels et automatisés.

* Tests manuels : les testeurs humains exécutent des tests sur une application logicielle pour identifier les erreurs. Le testeur suit un ensemble prédéterminé de cas de test pour s'assurer que l'application fonctionne comme prévu. Il s'agit d'un processus à forte intensité de main-d'œuvre et peut être lent.
* Tests automatisés : utilise des outils de test de logiciels pour exécuter des tests sur des applications logicielles. Avec cela, les tests sont préprogrammés, ce qui permet aux outils de test automatisés de comparer les résultats réels avec les résultats attendus.
* De manière générale, les tests automatisés sont plus rapides, plus efficaces et plus fiables que les tests manuels, mais ils nécessitent une maîtrise du langage de programmation et des outils logiciels utilisés.

**Examinons quelques autres différences entre les tests manuels et automatisés.**

|  |  |  |
| --- | --- | --- |
|  | **Test manuel** | **Tests automatisés** |
| **Définition** | Tests logiciels effectués manuellement par des testeurs humains | Tests de logiciels effectués à l'aide d'outils/scripts spécialisés |
| **Quand utiliser** | Pour les tests exploratoires et ad hoc, les tests d'utilisabilité, etc. | Pour les tests de régression, les tests de charge, les tests de performance, etc. |
| **Les ressources requises** | Testeurs humains, cas de test, données de test | Outils de test automatisés, scripts de test, données de test |
| **Génération de rapports** | Les rapports peuvent être générés manuellement | Les rapports peuvent être générés automatiquement |
| **Connaissances en programmation** | Peuvent être requises | Requises pour écrire des scripts de test et utiliser des outils de test. |

**Les différents types de tests de logiciels**

* Les tests de logiciels peuvent être largement divisés en deux types en fonction des techniques utilisées et du niveau de connaissance de l'application logicielle testée.
* Ceux-ci sont connus sous le nom de tests fonctionnels et non fonctionnels, et dans chaque catégorie, il existe plusieurs sous-catégories de tests logiciels.

**Tests fonctionnels**

* Ce type de test évalue la fonctionnalité ou le comportement de l'application logicielle en fonction des exigences et des spécifications de l'application.
* Des tests fonctionnels peuvent être effectués à différentes étapes du développement logiciel pour s'assurer que l'application répond aux exigences fonctionnelles et fonctionne comme prévu. Passons en revue les différents types de tests fonctionnels.

**Tests unitaires**

* Ce type de test fonctionnel est effectué au niveau du module ou du composant. L'objectif des tests unitaires est de tester des modules de code individuels (ou unités) pour s'assurer qu'ils fonctionnent correctement.
* Souvent, cela est fait par les développeurs de logiciels pendant la phase de développement, et cela peut être effectué manuellement ou avec des outils de test automatisés.
* Certains cas d'utilisation des tests unitaires incluent la vérification de l'exactitude des algorithmes, le test des conditions aux limites et la garantie que le code fonctionne comme prévu.
* L'extrait de code suivant est un exemple de test unitaire pour une fonction JavaScript asynchrone qui récupère les données de l'API JSONPlaceholder populaire.
* Ce test utilise MochaJS et vérifie si la fonction renvoie correctement le nom "Leanne Graham" lorsqu'elle reçoit une entrée de 1. C'est le résultat attendu puisqu'il s'agit du premier nom dans la liste des utilisateurs de l'API JSONPlaceholder.

Exemple de code de test unitaire en JavaScript :

* **async** **function** **fetchUserData**(userId) {
* **const** url = `https://jsonplaceholder.typicode.com/users/${userId}`;
* **const** response = **await** fetch(url);
* **const** data = **await** response.json();
* **return** data;
* }
* describe("fetchUserData function", () => {
* test("should return the correct user data", **async** () => {
* **const** userData = **await** fetchUserData(1);
* expect(userData.id).toBe(1);
* expect(userData.name).toBe("Leanne Graham");
* });
* });

**Test de la boîte blanche**

Cela implique de tester le fonctionnement interne de l'application logicielle, y compris le code, la structure et la conception.

Des testeurs ayant des connaissances en programmation intermédiaires/avancées l'exécutent. Généralement, cela se fait pendant la phase de développement. L'objectif des tests en boîte blanche est de s'assurer que le code fonctionne correctement et d'identifier toute erreur de code.

**Tests d'intégration**

Ce type de test fonctionnel examine l'interaction entre différents modules ou composants de l'application logicielle. L'objectif ici est de s'assurer que les composants fonctionnent correctement lorsqu'ils sont intégrés et que cela peut être effectué à différents niveaux.

**Tests incrémentiels**

Cela implique d'intégrer les composants un par un et de les tester au fur et à mesure de leur intégration. Cela permet d'identifier les bogues au début du processus de développement, car vous pouvez savoir si un composant fonctionne correctement avant de l'intégrer à d'autres composants. Il y a deux façons d'aborder cela.

* Descendant : les composants de niveau supérieur sont testés en premier tout en intégrant progressivement les modules ou composants de niveau inférieur. Cela permet d'identifier les défauts majeurs au début du processus de développement et de se protéger contre les fonctionnalités critiques/majeures avant de passer aux composants de niveau inférieur.
* De bas en haut : testez d'abord les composants de niveau inférieur, puis procédez à l'intégration des composants de niveau supérieur. De cette façon, vous identifiez les défauts mineurs au début du processus de développement avant de passer aux composants de niveau supérieur.

**Tests non incrémentiels**

Avec cette approche, vous intégrez tous les composants en même temps et les testez ensemble. Ceci est moins efficace que les tests incrémentaux car il devient difficile d'identifier et de déboguer les erreurs lorsque tous les composants sont intégrés en même temps.

L'exemple suivant de test d'intégration non incrémentiel utilise JavaScript, le framework de test Mocha.js et la bibliothèque d'assertions Chai.

Cela teste une application créée avec Express.js en s'assurant que l'URL racine (`/`) renvoie un code de réponse 200. Le deuxième test vérifie si le titre de la page est correct.

Exemple de code de test d'intégration en JavaScript :

**const** assert = require('chai').assert; **const** request = require('supertest'); **const** app = require('../app'); describe('Integration Tests', **function** () { describe('GET /', **function** () { it('should return a 200 response', **function** (done) { request(app) .get('/') .expect(200, done); }); it('should return the correct title', **function** (done) { request(app) .get('/') .end(**function** (err, res) { **if** (err) **return** done(err); assert.strictEqual(res.text, '<h1>Hello, world!</h1>'); done(); }); }); }); });

**Test de la boîte grise**

Cela implique de tester l'application logicielle avec une certaine connaissance des fonctions et processus internes. Les testeurs auront ainsi accès à des informations limitées sur l'application logicielle, telles que le schéma, l'architecture ou l'algorithme de la base de données.

Souvent, vous effectuerez des tests en boîte grise pendant la phase de test.

**Test du système**

Ce type de test fonctionnel évalue l'application logicielle complète dans son ensemble. Avec les tests du système, l'objectif est de s'assurer que l'application logicielle répond aux exigences fonctionnelles.

Elle peut être effectuée manuellement ou à l'aide d'outils automatisés. Certains cas d'utilisation des tests système incluent le test de l'interface utilisateur ou le test des performances et de la sécurité de l'application.

Le code suivant montre un exemple de test du système. Cet exemple JavaScript utilise la bibliothèque Supertest pour envoyer des requêtes HTTP à une application que nous testons.

Le premier cas teste une requête GET à `/users` qui devrait renvoyer une liste d'utilisateurs avec les propriétés correctes. Le deuxième cas teste une requête POST censée ajouter un nouvel utilisateur avec les bonnes propriétés.

Exemple de code de test système en JavaScript :

**const** assert = require('chai').assert;

**const** request = require('supertest');

**const** app = require('../app');

describe('System Tests', **function** () {

describe('GET /users', **function** () {

it('should return a list of users', **function** (done) {

request(app)

.get('/users')

.expect(200)

.end(**function** (err, res) {

**if** (err) **return** done(err);

assert.isArray(res.body, 'response should be an array');

assert.property(res.body[0], 'name', 'user has a name');

assert.property(res.body[0], 'email', 'user has an email');

done();

});

});

});

describe('POST /users', **function** () {

it('should add a new user', **function** (done) {

request(app)

.post('/users')

.send({ name: 'John', email: 'john@example.com' })

.expect(201)

.end(**function** (err, res) {

**if** (err) **return** done(err);

assert.property(res.body, 'id', 'user has an id');

assert.equal(res.body.name, 'John', 'user has the correct name');

assert.equal(res.body.email, 'john@example.com', 'correct email');

done();

});

});

});

**Test de la boîte noire**

Cela ne nécessite aucune connaissance pratique des fonctions ou processus internes d'une application. Les testeurs exécutent des cas de test prédéterminés pour identifier les problèmes de fonctionnalité plutôt que les problèmes de mise en œuvre internes. Ceci est purement axé sur les entrées et les sorties.

Les tests de boîte noire permettent aux testeurs de garantir la qualité et la fiabilité du logiciel en simulant la manière dont les utilisateurs interagiront avec lui.

**Test de bout en bout**

Cela détermine l'ensemble du flux d'une application, à partir de l'interaction initiale de l'utilisateur et en passant par toutes les étapes jusqu'à ce que la sortie finale soit générée.

Cela implique de tester l'application sur différents systèmes, serveurs et interfaces pour s'assurer que tous les composants fonctionnent ensemble de manière transparente et répondent aux exigences des parties prenantes ou des clients.

Des tests de bout en bout sont souvent effectués pour identifier les problèmes d'intégration, les goulots d'étranglement des performances ou les défauts fonctionnels qui surviennent en raison de la complexité de l'application et de ses différents composants.

**Test de fumée**

Ce type de test logiciel évalue si les fonctionnalités principales ou critiques d'une application logicielle fonctionnent comme prévu et si l'application est suffisamment stable pour des tests supplémentaires. Cela évite de perdre du temps ou des ressources dans une situation instable ou défectueuse.

Ces tests peuvent impliquer l'exécution d'une série de tests de base sur l'application, tels que le lancement de l'application, la connexion et l'exécution de tâches simples, pour vérifier que l'application fonctionne correctement.

**Test de santé (Sanity)**

Ce type de test logiciel évalue les nouvelles fonctionnalités ou modifications pour vérifier si elles fonctionnent comme prévu. Les tests peuvent impliquer l'exécution d'une série de tests rapides sur l'application, tels que des tests de fonctionnalités de base ou des tests d'intégration pour vérifier que les nouvelles fonctionnalités ou les modifications n'ont pas introduit de problèmes ou de bogues majeurs.

Tests d'acceptation

**Tests d’acceptation**

Ce type de test fonctionnel est utilisé pour évaluer si une application ou un produit répond aux critères d'acceptation et aux exigences des parties prenantes ou des clients.

Il s'agit de tester l'application sur différents cas d'utilisation et de vérifier qu'elle répond aux exigences fonctionnelles et non fonctionnelles ainsi qu'aux attentes et aux besoins des utilisateurs.

**Test alpha**

Il s'agit de l'un des types de tests logiciels en génie logiciel menés en interne par des développeurs ou des testeurs pour évaluer une application avant qu'elle ne soit rendue publique.

Cela comprend souvent le test de l'application pour identifier les problèmes ou les bogues et la prise de mesures correctives avant la publication.

**Tests bêta**

Il s'agit de l'un des types de tests menés par un groupe d'utilisateurs finaux ou de clients, et il est terminé avant la publication d'une application, mais toujours dans un environnement réel.

Il permet de rassembler des informations sur l'expérience utilisateur, les fonctionnalités et les performances des applications. Les commentaires peuvent ensuite être utilisés pour identifier les problèmes ou les bogues afin d'améliorer l'expérience utilisateur.

**Tests non fonctionnels**

Cette approche de test évalue les aspects non fonctionnels d'une application logicielle en évaluant des éléments tels que les performances, la convivialité globale, la compatibilité, la sécurité et la fiabilité. Cela garantit que les applications logicielles répondent aux normes de qualité et fonctionnent comme prévu.

**Tests de sécurité**

Les tests de sécurité identifient les vulnérabilités et les menaces d'une application logicielle. Cela peut inclure des tests de vulnérabilités telles que l'injection SQL, le script intersite (XSS) et d'autres menaces de sécurité.

Les tests de sécurité sont effectués via des techniques de test manuelles et automatisées, qui incluent des tests de pénétration, des analyses de vulnérabilité et des revues de code.

**Test de performance**

Ce type de test non fonctionnel évalue les performances des applications logicielles dans différentes conditions et peut être effectué à l'aide de diverses techniques, comme nous allons maintenant l'examiner.

**Test de charge**

Cela évalue les performances de l'application sous des charges normales et maximales. L'objectif est de s'assurer que l'application peut gérer la charge attendue sans problèmes de performances.

**Tests de résistance**

Ceci est utilisé pour vérifier les performances de l'application sous des charges ou des conditions extrêmes. Cela permet d'identifier les points de rupture de l'application et de vérifier si elle peut gérer des charges inattendues ou anormales.

Tests d'évolutivité

Cela évalue la capacité d'une application à gérer des charges de travail ou des utilisateurs accrus. Cela permet de déterminer si l'application peut évoluer à la hausse ou à la baisse selon les besoins pour gérer des charges de travail variables.

**Test de stabilité**

Ceci est utilisé pour déterminer la fiabilité (ou la stabilité) d'une application sur une période prolongée. L'objectif est de s'assurer que l'application peut fonctionner de manière cohérente dans le temps sans dégradation des performances ni pannes.

**Tests d'utilisation**

Ce type de test non fonctionnel évalue la convivialité et la facilité d'utilisation d'une application logicielle.

Cela peut impliquer d'observer vos utilisateurs lorsqu'ils interagissent avec une application, ce qui permet d'identifier les problèmes ou les défis que les utilisateurs peuvent rencontrer, tels que des temps de réponse lents, des difficultés à effectuer des tâches ou des interfaces utilisateur confuses.

**Essais exploratoires**

Cela implique de tester des logiciels sans cas de test ou scripts prédéfinis en explorant et en expérimentant une application. Cela permet d'identifier les erreurs ou les bogues qui n'ont pas été détectés par les tests prédéfinis existants tout en acquérant une meilleure compréhension des fonctionnalités et des performances d'une application.

**Test du navigateur**

Ces tests garantissent que les applications Web ou les sites Web sont compatibles avec différents navigateurs Web, tels que Chrome, Firefox, Internet Explorer, Safari, etc.

Cela comprend le test de divers aspects du site Web, tels que la mise en page, la conception, les fonctionnalités et les performances, sur différents navigateurs et appareils. Ceci est essentiel pour les applications Web ou les sites Web afin de garantir qu'ils sont accessibles à un public plus large avec une expérience utilisateur cohérente.

**Test d'accessibilité**

Ce type de test logiciel évalue l'accessibilité d'une application, en s'assurant que l'application est utilisable par tout le monde, y compris les personnes malvoyantes ou malentendantes.

Cela peut impliquer d'évaluer l'application par rapport aux directives et aux normes d'accessibilité, telles que WCAG (Web Content Accessibility Guidelines), et de la tester avec des technologies d'assistance telles que des lecteurs d'écran ou des loupes.

**Test de compatibilité**

Ce type de test de logiciel non fonctionnel vérifie si une application peut fonctionner comme prévu avec différents matériels, systèmes d'exploitation, navigateurs, bases de données et autres composants logiciels. Cela permet de faire fonctionner l'application de manière transparente sur différentes plates-formes pour garantir une expérience utilisateur cohérente.

**Autres types importants de tests de logiciels**

Au-delà des types de tests qui relèvent des tests fonctionnels ou non fonctionnels, les testeurs de logiciels peuvent également mettre en œuvre plusieurs autres techniques de test de logiciels.

**Test d'installation**

Il s'agit de l'un de ces types de tests en génie logiciel qui évalue si une application logicielle ou un produit peut être installé, mis à niveau ou désinstallé sans aucun problème ni erreur.

Les tests peuvent impliquer l'installation du logiciel sur différentes plates-formes avec différentes configurations pour identifier les problèmes liés à l'installation, tels que les fichiers manquants ou les dépendances, les échecs d'installation ou les conflits avec d'autres applications ou composants logiciels.

**Test de configuration**

Cela vérifie si une application logicielle ou un produit peut fonctionner comme prévu sur différentes configurations et configurations.

Cela inclut la vérification des produits logiciels sur différents matériels, systèmes d'exploitation, navigateurs, configurations réseau ou autres composants logiciels pour identifier tout problème lié à la configuration ou tout comportement ou performance inattendu.

**Test de récupération**

Test si une application peut se remettre d'échecs ou d'erreurs et reprendre ses opérations normales sans perte ni corruption de données.

Pour ce faire, les testeurs simulent différentes pannes, telles que des pannes matérielles ou réseau, des pannes logicielles ou des pannes de courant, et vérifient que l'application peut s'en remettre et continuer à fonctionner comme prévu. Cela peut aider à minimiser les temps d'arrêt et la perte de données en cas de pannes ou d'erreurs.

**Les tests de régression**

Ce type de test détermine si les changements ou modifications apportés à un produit logiciel ont introduit de nouveaux problèmes ou bogues ou ont eu un impact sur la fonctionnalité existante. Cela se fait en réexécutant les tests et les cas de test actuels sur l'application pour repérer les nouveaux bogues qui peuvent être apparus.

**Test de l'interface utilisateur**

Il s'agit de l'un des types de test qui vérifie l'interface utilisateur d'une application, en s'assurant qu'elle est conviviale, intuitive et répond aux besoins du public cible.

Dans ce cas, les testeurs testent l'interface sur différents appareils et plates-formes pour s'assurer que les utilisateurs peuvent facilement naviguer et interagir avec l'application. Cela peut également impliquer de tester la conception visuelle, la mise en page et la réactivité pour garantir une expérience utilisateur cohérente et positive.

**Tests ad hoc**

C'est l'un des types de tests en génie logiciel effectués sans aucun cas de test formel ni documentation. Les testeurs l'exécutent généralement de manière exploratoire ou de manière libre pour identifier les bogues qui n'ont pas été trouvés via les tests traditionnels.

Les tests ad hoc sont utiles dans les situations où les exigences ne sont pas claires et où les testeurs doivent utiliser leur expertise et leurs connaissances pour identifier les problèmes potentiels.

**Test de rétrocompatibilité**

Cela teste si une nouvelle version du logiciel est compatible avec les versions précédentes. Cela garantit que les nouvelles versions n'interrompent pas les fonctionnalités des anciennes versions et que les données et les fichiers créés avec les anciennes versions peuvent être utilisés avec la nouvelle version.

Les tests de compatibilité descendante sont cruciaux pour les logiciels ayant une large base d'utilisateurs, car ils garantissent que les utilisateurs peuvent continuer à utiliser le logiciel sans problème.

**Tests agiles**

Ce type de test de logiciels suit les mêmes principes qu'une approche de développement agile, à savoir qu'il s'agit d'une approche itérative et incrémentale et vise à fournir rapidement des logiciels de haute qualité.

Les tests agiles incluent des tests continus tout au long du cycle de développement, avec des tests fréquents et des boucles de rétroaction pour s'assurer que le logiciel répond aux exigences des clients.

**Test d'API**

Cela implique de tester l'interface de programmation d'application (API) en testant les entrées/sorties (E/S), les fonctionnalités, les performances, la fiabilité et la sécurité. Les tests d'API sont effectués avec des outils de test automatisés qui simulent divers scénarios.

**Test de fiabilité**

Cela détermine la stabilité (ou la fiabilité) d'une application en testant une application dans des conditions normales et extrêmes.

Il comprend des tests de stress, de performance, de charge et d'endurance. Les tests de fiabilité visent à garantir que l'application logicielle peut fonctionner correctement, de manière cohérente et fiable sur une période prolongée.

**Conclusion**

Les tests logiciels sont un processus crucial qui garantit que les applications logicielles répondent aux normes requises et fonctionnent comme prévu. Avec la complexité croissante des applications logicielles et les demandes des clients, les tests sont devenus une partie indispensable du processus de développement.

En tant que professionnel du développement logiciel, une bonne compréhension des différents types de tests logiciels vous permet d'identifier l'approche de test appropriée en fonction de la portée, de la complexité et des exigences spécifiques de votre projet.

Dans cet article, nous avons couvert les différents types de génie logiciel que vous devez connaître en 2023. Avec l'aide de notre guide, vous devriez vous sentir plus en confiance pour choisir les bonnes méthodes de test pour votre projet.

**Questions fréquemment posées**

**1. Quels sont les principaux types de tests logiciels ?**

Au plus haut niveau, les principaux types de tests logiciels sont les tests manuels et automatisés. Les tests manuels consistent à exécuter des tests par des testeurs humains pour identifier les erreurs potentielles, tandis que les tests automatisés impliquent des outils logiciels pour exécuter des tests.

Au-delà de ces deux types, il existe de nombreux autres types de tests de logiciels, comme indiqué dans notre guide.

**2. Combien de types existe-t-il dans les tests de logiciels ?**

Il est difficile de donner un nombre exact pour les différents types de tests de logiciels, car il s'agit d'un domaine en constante évolution avec de nouvelles techniques et approches de test de logiciels constamment développées. Notre guide couvre cependant plus de 30 types de tests de logiciels.

**3. Quels sont les quatre types de tests de systèmes ?**

Les quatre principaux types de tests système sont les tests d'acceptation, les tests système, les tests d'intégration et les tests unitaires. Consultez les sections individuelles ci-dessus pour plus d'informations à ce sujet.

**4. Quelles sont les 5 méthodes de test ?**

Les cinq principales méthodes de test sont les tests unitaires, les tests d'intégration/système, les tests fonctionnels, les tests de régression et les tests d'acceptation. Pour plus d'informations, consultez les sections individuelles de notre guide ci-dessus.

Source web anglais : <https://hackr.io/blog/types-of-software-testing>