Engineer

Tests de contrat

tests des accords d'interface entre composants

Ce que c'est

Les tests de contrat vérifient que deux composants s'accordent sur les requêtes, réponses ou messages qu'ils échangent. Un écran de suivi de colis et un service de suivi peuvent tous deux passer leurs tests isolés. Pourtant l'écran échoue s'il attend un champ `status` et que le service envoie `state` . Un test d'interface formule ce que l'écran utilise puis vérifie que le service sait le fournir.

Pact présente une méthode dans laquelle le consommateur échange avec un substitut contrôlé du fournisseur. Le test enregistre les interactions dans un contrat. Le fournisseur vérifie ensuite ce contrat en exécutant sa propre implémentation dans des conditions précises. Lors d'un échange de messages, la vérification porte sur le message produit et consommé. Les équipes peuvent travailler et livrer à des rythmes différents, pourvu que les preuves concernent les versions qu'elles utiliseront ensemble.

Le terme s'emploie aussi pour la vérification d'un fournisseur par rapport à sa documentation publiée, par exemple une description OpenAPI. Cette méthode sert quand la spécification écrite est l'accord partagé. Elle peut définir les champs, les codes d'état et les valeurs permises. Les exemples établis par le consommateur indiquent quelles combinaisons un appelant exploite et sur quelles hypothèses il s'appuie. Les deux méthodes peuvent coexister. Un parcours dans l'intégration réelle permet de vérifier ce qui dépasse l'échange, notamment le routage et la configuration.

Utilisez le concept pour prendre une décision

Utilisez « Que sont les tests de contrat ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. La partie invisible pour les utilisateurs : où résident les données, quelles règles s'appliquent et comment les autres systèmes peuvent y accéder.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 6 pipelines ETL/ELT conçus et exploités — 6 pipelines ETL sur trois moteurs de bases de données

6 pipelines ETL/ELT conçus et exploités Contactez-nous →

Pourquoi c’est important

Une interface dure souvent plus longtemps que le code qui l’a introduite. L’équipe du service peut renommer un champ et adapter ses propres tests sans modifier un appelant maintenu ailleurs. Une application mobile peut rester sur une ancienne version plusieurs mois. Un simple changement de type compte s’il détermine le texte affiché. Un contrat vérifiable rend ces dépendances visibles avant publication.

Lorsque les mises en service ont des rythmes distincts, il faut savoir quelles versions sont compatibles. Un client ancien peut comprendre peu de valeurs d’état et un service récent vouloir ajouter une étape intermédiaire. Les équipes peuvent adapter le consommateur d’abord, accepter les deux formes pendant la transition et retirer l’ancien comportement après avoir vérifié les versions dont elles dépendent. L’ordre sûr dépend des contrats, de l’usage réel et de la politique de diffusion. La réussite d’une combinaison récente ne couvre pas tous les clients encore présents.

Les contrôles doivent exprimer les véritables points d’accord. Si toute référence bien formée convient, vérifions le format sans imposer la valeur exacte de l’exemple. Si un champ détermine un comportement visible, contrôlons son type et les valeurs comprises par le client. Un bon cas protège l’accord tout en laissant le fournisseur faire évoluer les détails internes.

À quoi cela ressemble en pratique

Imaginons une application de suivi de colis. Elle demande un colis par sa référence. Lorsque le colis existe et est en transit, l’écran attend une réponse de succès, la référence et un texte qu’il sait associer à une étiquette. Lorsqu’il ne trouve pas le colis, il attend une autre réponse et propose de revoir la référence. Chaque interaction est associée à une condition préalable, par exemple l’existence d’un colis en transit ou l’absence de toute référence correspondante.

Faisons passer la requête par le vrai code du consommateur en utilisant le substitut contrôlé. Vérifions l’affichage qui résulte de chaque échange. Exerçons ensuite les mêmes interactions sur le chemin réel de réponse du fournisseur.

Créons des données déterministes pour chaque état et réinitialisons-les entre les essais. Si la base de données ou le transporteur est remplacé, gardons dans le test l'analyse de la requête, la logique du service et la construction de la réponse dont dépend le consommateur.

Remplaçons provisoirement le champ `status` par `state` sans modifier le client. La vérification devrait signaler le champ manquant et permettre d'identifier l'échange. Un champ facultatif ignoré par le consommateur reste acceptable si l'accord le permet. Relions chaque preuve à sa version. Mélanger le résultat d'un nouveau fournisseur et une copie non identifiée d'un ancien contrat empêche de savoir ce qui a été mis à l'épreuve.

Notre note sur l'ouverture d'une application dans les portefeuilles mobiles Solana montre des intégrations qui attendent chacune un chemin de lien et un encodage distincts. Elle expose ces différences et propose des essais sur appareil. Un test de la forme des liens générés pourrait préserver ces accords. Un essai sur un véritable appareil permet de voir si le navigateur transmet le lien à l'application installée et si le parcours aboutit. Les deux observations ne couvrent pas la même chose.

Où les problèmes surviennent

Un contrat se rigidifie si chaque octet doit correspondre à un exemple enregistré. Le fournisseur peut être libre d'ajouter un champ ignoré ou de reformater un message auxiliaire. Rendons un champ obligatoire lorsqu'un appelant l'utilise ou lorsque la spécification publique l'exige. Fixons l'ordre, les unités et les mots exacts aux endroits où l'accord en dépend.

Un test perd aussi sa force s'il contourne la limite réelle. Vérifier un objet JSON écrit à la main peut laisser le sérialiseur hors du parcours. Constater que le substitut a produit le résultat préparé ne montre pas que le consommateur sait le lire. Exécutons le vrai code de requête ou de réponse de la partie vérifiée et notons ce que fournissent les dépendances remplacées.

Enfin, un exemple réussi couvre peu de choses sur les données manquantes, les références inconnues ou les anciennes versions. Recensons les erreurs traitées par le client et les compatibilités promises par le fournisseur. Gardons des essais d'intégration qui exercent la configuration et le routage. Le contrat apporte une preuve ciblée et assez rapide pour accompagner chaque évolution pertinente.

Pour aller plus loin

- Pact et la définition des tests de contrat
- Vérification du consommateur et du fournisseur dans Pact
- Documentation de Pact sur les états du fournisseur
- Spécification OpenAPI pour les interfaces HTTP

Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Que sont les tests de contrat ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/contract-testing/