Découverte des tests d'intégration et d'interface avec Cypress

Si les tests unitaires sont relativement simples à écrire et couvrent une bonne partie des bugs qui pourraient se produire dans une application, ils ne peuvent couvrir les problèmes qui résulteraient d'un conflit entre différentes fonctions qui agissent ensemble pour fournir une fonctionnalité à un utilisateur.

Les tests d'intégration ou "fonctionnels", peuvent être très proches des tests unitaires comme plus proches de l'interface utilisateur.

Ce qui va distinguer un test d'intégration d'un test d'interface c'est le focus que l'on va mettre sur le test:

- cherches-t-on à savoir si la soumission d'un formulaire fonctionne côté serveur ("fonctionnel") ou côté client? ("test d'interface")
- vas-t-on soumettre en POST des valeurs sur le serveur ("fonctionnel") ou simuler le comportement réel d'un utilisateur en scriptant un navigateur? ("test d'interface")
- cherches-t-on à tester la charte graphique ou le fonctionnement d'une partie du site?

Aujourd'hui, nous allons utiliser Cypress: un framework de test développé en Javascript avec lequel on peut faire des tests unitaires, d'intégration ou encore d'interface!

Vous développerez dans 2 travaux pratiques les deux types de tests: fonctionnels et d'interface.

Le bon réflexe: le "smoke testing"

Quand on souhaite commencer les tests sur une application déjà développée et en production, on ne sait jamais comment commencer:

- des tests unitaires? Pas si simple
- des tests d'interface? Simple mais long à exécuter

Dans ce cas, vous pouvez développer une suite de tests facile à écrire et à exécuter et qui a l'avantage d'être efficace:

- si la suite passe, pas sûr que votre application fonctionne;
- mais si elle ne passe pas, il ne faut absolument pas déployer la mise à jour!

Concrètement, cette suite de tests vérifie que chaque page du site est accessible, c'est à dire qu'elle retourne un code HTTP 200 quand on y accède.

On peut bien évidemment compléter ce test, vérifier quelques informations disponibles sur la page et au fur et à mesurer compléter la suite de tests avec des scénarios de plus en plus complets.

Vous allez installer Cypress et écrire votre première suite de tests, dans ce TP vous aurez de nombreux indices pour vous y aider, mais il est attendu de votre part un minimum d'autonomie.

Installation de Cypress

> Créez un dossier dans lequel effectuer votre TP, les commandes seront à effectuer à l'intérieur de ce dossier.

Installez Cypress à l'aide de npm:

```
npm install cypress
```

Ensuite, pour vérifier que l'installation s'est bien passée exécutez la commande suivante:

```
./node_modules/.bin/cypress open
```

> Un souci? Jetez un oeil à la documentation.

Une fenêtre devrait s'ouvrir, ce qui vous permet de tester le client Cypress avec des suites de tests d'exemple.

Ecrivez une suite de test d'intégration: smoke testing

Cypress a une documentation extraordinaire! Elle est complète et regorge d'exemples et de vidéos explicatives.

Ensuite, il dispose d'un site d'exemple que l'on va contrôler avec nos tests.

Dans cet exercice, vous devez écrire des tests confirmant que toutes les pages de ce site sont accessibles, et pour vous assurer que vous êtes sur la bonne page, vous devez également contrôler le titre principal (`h1`) de chaque page.

Pour vous aider, lisez la section "Getting started".

> A partir du moment où nous testons des éléments d'interface dans ces tests, nous les qualifierons de tests d'interface

Bonus

Vous avez peut être remarqué que sans même faire d'assertions, la suite de tests passe.

Vous pouvez compléter cette suite de tests à l'aide d'assertions comme nous l'avions vu dans le TP sur les tests unitaires.

Tests "End to End": vérifions l'interface utilisateur

Au plus haut niveau de la pyramide de tests de Mike Cohn se trouvent les tests End to End ou dits d'interface.

Ici, nous allons plutôt vouloir vérifier que l'interface de l'application fonctionne comme attendue:

- est-ce que les champs d'un formulaire sont accessibles?
- est-ce qu'on peut le soumettre?

- est-ce que les polices d'écriture sont bien rendues?
- est-ce qu'il y a une régression visuelle entre la branche master et celle de pré prod?

Il existe toute sorte de tests end to end, et de nombreux outils pour vérifier notamment la présence de bugs de régression visuelle.

Dans cet exercice, et en vous aidant des exemples de la documentation, je souhaite que vous testiez la recherche dans Google (ou Qwant) du terme "ESGI".

Il faudra:

- accéder à l'url du moteur de recherche;
- remplir le champ de recherche avec le terme de la requête;
- récupérer les 10 premiers résultats et les afficher (console.log() suffit);
- accéder à chacun des dix liens et prendre une capture d'écran du site;
- les captures d'écran doivent être dans un dossier nommé `captures`;
- enfin, confirmer que le site en question contient le terme recherché à l'aide d'une assertion.

Bon courage!