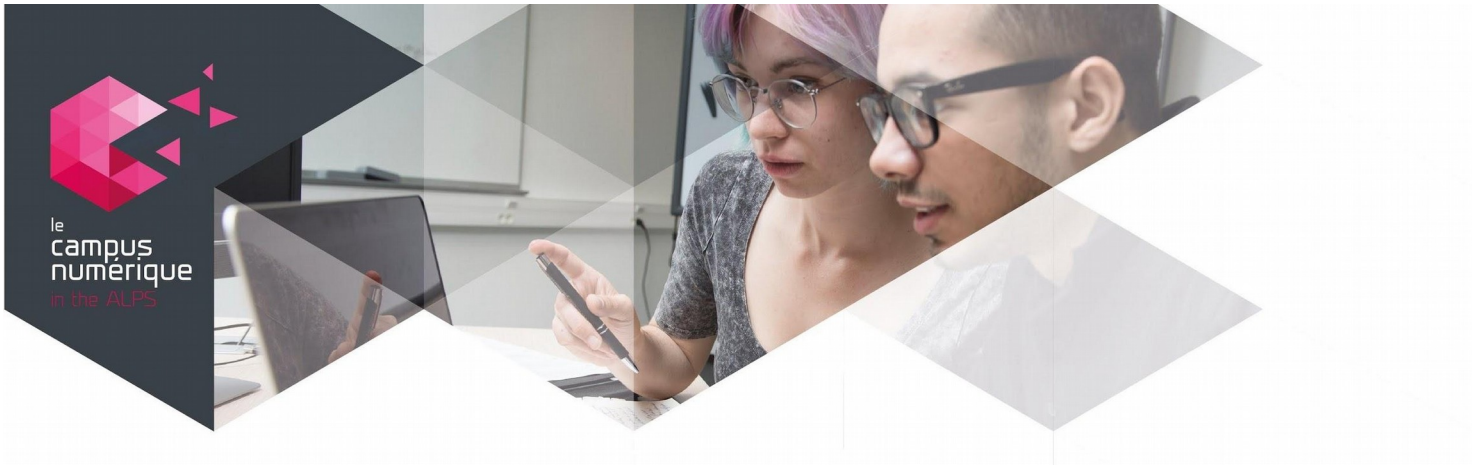


Métiers du test

Rappel sur les tests

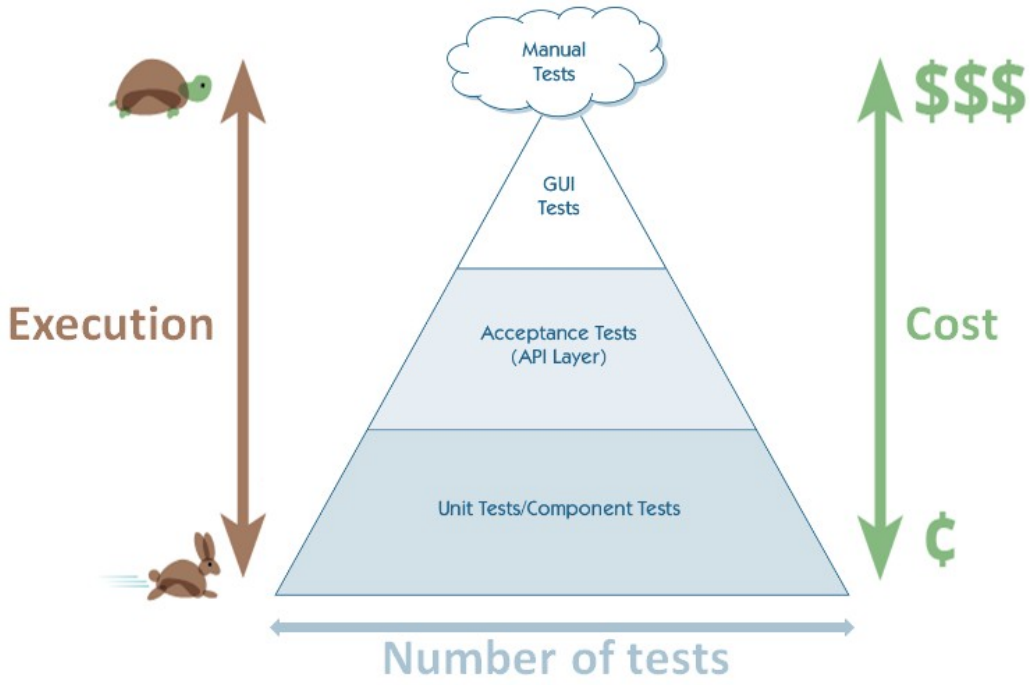
Qu’est-ce qu’un test ?

Un test est une activité qui consiste à vérifier qu’un système informatique fonctionne correctement. Il peut s’agir d’une pièce de code, d’un module, d’une application toute entière. Il existe de nombreuses motivations amenant à la mise en place de solutions de test logiciels : validation fonctionnelle, non-régression, performances, stabilité.

On distingue les tests logiciels selon deux grandes catégories : manuels ou automatisés.

La pyramide des tests

La pyramide des tests, illustrée ci-dessous, représente la répartition “idéale” des différents types de tests pour un projet informatique. On retrouve à la base de la pyramide les tests unitaires qui sont rapides à exécuter et peu coûteux en terme de mise en place au moment du développement. Étant donné leur vitesse d’exécution, ils permettent d’avoir un retour rapide après une modification de code.



À l'étage supérieur, on retrouve un ensemble comprenant les tests d'acceptation (tests d'intégration et fonctionnels) qui vont avoir pour but de tester une application au niveau fonctionnel en réalisant des parcours utilisateurs. Ces tests sont idéalement automatisés.

Puis nous retrouvons les tests d'interface graphique et les tests manuels. Ces deux dernières catégories doivent idéalement contenir un faible nombre de tests par rapport aux autres catégories. En effet, les tests de GUI peuvent être fastidieux à mettre en place et à maintenir tout au long du cycle de développement.

Les tests manuels quant à eux sont la plupart du temps exécutés uniquement lors de la finalisation d'une nouvelle version. En effet, leur exécution nécessite de nombreuses ressources humaines et il serait impossible de dérouler ces tests au quotidien.

Activité 1

Dans le cadre de ces activités, nous utiliserons les langages HTML et Javascript pour concevoir les tests.
L'objectif de cette première activité est de se familiariser avec les outils utilisés par la suite.

Consignes :

- Créer un projet JS basique utilisant NPM comme outil de gestion de dépendances
- Importer le projet dans votre dossier Github
- Implémenter les fonctions d'une calculatrice en JS (addition, soustraction, multiplication, division)
- Mettre en place des tests unitaires sur chacune des fonctions en utilisant le framework Jest
- Implémenter une fonction qui calcule le nombre de secondes écoulées depuis l'an 2000
- Mettre en place un test unitaire sur la fonction ci-dessus, toujours via Jest
- Afficher le rapport de couverture des tests pour l'ensemble des fonctions implémentées

Ressources :
https://jestjs.io/docs/en/getting-started

Compétences à valider :

- Se rafraîchir la mémoire sur les tests unitaires
- Utilisation de mock
- Affichage de couverture de test

Activité 2

Introduction

Vous venez d'arriver dans la société "**Url en short**" en tant que testeur.
Cet éditeur de logiciel est en train de finaliser son nouveau produit de raccourcisseur d'URL : Polr.
A l'image de site comme http://bitly.com, Polr permet de générer des URL raccourcies et de réaliser des actions de suivi sur ces URL, comme par exemple, comptabiliser le nombre de clics.
Une version de test a été mise en production à l'adresse suivante : http://polr.web-74.com/

En tant que nouvel arrivant, vous devez vous familiariser avec le métier de testeur, ainsi qu'avec le fonctionnement de l'application Polr. Sous la houlette du chef de projet, votre rôle dans l'entreprise sera de tester automatiquement les parcours clients de ce nouveau produit.

Le chef de projet vous présente le projet (présentation orale des formateurs)

L'objectif de cette première activité est de :
- Comprendre les objectifs du projet (pour l'utilisateur, pour les administrateurs, pour l'éditeur du logiciel)
- Imaginer les parcours clients les plus importants
- Rédiger un cahier de tests fonctionnels contenant les principaux parcours de l'application Polr.

À titre d'exemple, deux tests ont été rédigés par le chef de projet (voir documents ressources).

Afin de faciliter l'écriture de vos tests, un compte admin a été créé:
User : admin
Pwd : campus

Livrable:
- Le cahier de tests au format .doc.

Ressources:
- Deux exemples de tests au format .doc

Compétences à valider:
- Identifier les principaux parcours utilisateur d'une application
- Rédiger un cahier de tests fonctionnels

Activité 3

La première activité vous a amené à écrire des tests fonctionnels que l'on peut jouer à la main.

Le chef de projet souhaiterait désormais que l'exécution des ces tests soit automatisée. Pour ce faire vous allez devoir utiliser un outil pour coder et exécuter ces tests.

Les outils de tests fonctionnels peuvent être parfois un peu compliqués à installer. Pour ne pas perdre de temps sur cette phase, nous avons installé un système et préparé quelques tests d'exemple :

https://github.com/vberthet/module-test-puppeteer

Outils utilisés :

Pour faire les tests, nous utilisons 2 outils :

- Puppeteer : c'est un outil qui permet de piloter un navigateur avec du code javascript
- Jest : c'est un outil qui permet de lancer des tests automatiques (comme phpunit en PHP)

Consigne :

- Faire un fork de ce projet sur votre compte github
- Lire le README.md
- Cloner le projet
- Lancer les tests fonctionnels chez vous
- Ajouter vos propres tests

Livrable:

- Faire un fork du projet module-test-puppeteer
- Le code de vos tests pushés dans votre fork

Ressources:

- https://github.com/vberthet/module-test-puppeteer
- https://jestjs.io/docs/en/getting-started

Compétences à valider:

- Utiliser un outil de tests fonctionnels
- Lancer automatiquement les tests à chaque push dans GIT.

Activité 4

Mise en place d'une intégration continue.

Sur certains projets, lancer des tests peut durer plusieurs heures. Dans ce cas, les développeurs ne lancent pas les tests en entier à chaque modification du code, ça serait beaucoup trop long.

On utilise alors des systèmes qui lancent les tests automatiquement à chaque fois qu'on fait un git push. Si les tests lancés sont en échec, on reçoit un email, sinon on ne reçoit rien.

Ce principe général s'appelle l'intégration continue.

Pour notre projet, on va mettre en place une intégration continue en utilisant travis-ci.

Livrable :

- Vos tests fonctionnels doivent être visibles sur travis-ci

Ressources :
Pas de ressource particulière.

Compétences à valider :

- Lancer les tests à chaque push dans github
- Consulter le résultat des tests dans travis-ci
- Afficher l'indicateur de résultat des tests dans le README.

Activité 5

Réponse à un feedback client et rédaction d'un ticket dans github si nécessaire

Les clients peuvent remonter des problèmes à travers un formulaire de feedback. Nous allons interagir avec le client pour préciser son feedback et avoir toutes les informations nécessaires qui nous permettront de résoudre son problème.

Nous travaillerons par groupe à l'oral pour aider à qualifier les feedbacks des clients

Livrable:

- Les tickets détaillés dans GitHub
- Les réponses (rédigées collectivement) aux feedbacks client

Resources:

- Pas de ressource nécessaire

Compétences à valider:

- Rédiger des tickets dans un outil de bug tracking