Qualité & tests

Partie 1

Exercice 1 : Découverte des tests unitaires avec Jest

Objectif : Installer et configurer Jest, puis écrire des tests unitaires sur des fonctions simples.

1. Installe Jest dans un projet Node.js.
2. Crée un fichier `utils.js` contenant trois fonctions :
   - `somme(a, b)` : retourne la somme de `a` et `b`.
   - `estPair(n)` : retourne `true` si `n` est pair, sinon `false`.
   - `factorielle(n)` : retourne `n!` ($n! = n \times (n-1) \times ... \times 1$).
3. Écris des **tests unitaires** avec Jest pour vérifier ces fonctions.

Exercice 2 : Détection des erreurs et mocks

Objectif : Manipuler les **mocks** et tester les cas extrêmes.

1. Ajoute une gestion d'erreur pour `factorielle()` (interdiction des nombres négatifs).
2. Ajoute un **mock** pour vérifier qu'une fonction est appelée.

Exercice 3 : Mise en place du TDD

Objectif : Écrire les **tests avant d'écrire le code** (TDD).

1. Implémente une fonction `inverse(chaine)` qui inverse une chaîne.
2. Suis la méthodologie TDD :
   - **Écris les tests en premier**.
   - Implémente la fonction pour que les tests passent.

Exercice 4 : Premier test e2e avec Playwright

Objectif : Tester un **formulaire de connexion**.

1. Installe Playwright et configure-le.
2. Teste la présence des champs `email` et `mot de passe`.
3. Simule la saisie et la connexion.

Exercice 5 : Mocks avancés avec Jest

Objectif : Simuler des appels API avec des **mocks** et vérifier leur comportement.

1. Implémente une fonction `getUserData(id)` qui récupère des données utilisateur via une API (`fetch`).
2. Utilise un **mock** pour simuler une réponse API.
3. Vérifie que la fonction **appelle bien l'API** et renvoie les bonnes données.

Exercice 6 : Tests de performance avec Playwright

Objectif : Mesurer les temps de chargement et optimiser une page.

1. Charge une page web et mesure ses **performances** (temps de réponse, métriques LCP/FCP).
2. Vérifie qu'elle se charge en moins de **3 secondes**.

Exercice 7 : Reproductibilité avec Docker

Objectif : Utiliser Docker pour garantir la **même configuration de tests sur toutes les machines**.

1. Crée un **fichier Dockerfile** pour exécuter Jest et Playwright.
2. Assure-toi que l'environnement est **identique pour tous les développeurs**.

Exercice 8: CI/CD avancé - Exécuter les tests sur plusieurs navigateurs

Objectif : Tester avec **Chrome, Firefox et WebKit** en CI/CD.

Modifie **GitHub Actions** pour exécuter les tests Playwright sur plusieurs navigateurs.