Qualité & tests

Partie 2

Exercice 1 : Mise en place de ESLint

Objectif : Utiliser un **linter** pour améliorer la qualité du code.

1. Installe ESLint.
2. Configure-le pour **interdire `var` et forcer les `const` si possible**.

Exercice 2 : Intégration de tests dans GitHub Actions

Objectif : Automatiser les **tests Jest et Playwright** sur GitHub.

Ajoute un workflow GitHub pour exécuter **Jest et Playwright**.

Tests Unitaires Avancés avec Jest

Exercice 3 : Tester une fonction de conversion monétaire

- Implémente une fonction `convertirMontant(montant, taux)` qui convertit un montant en fonction d'un taux de change.
- Vérifie avec Jest que la fonction respecte ces conditions :
    1. `convertirMontant(100, 1.2)` doit retourner **120**.
    2. Si `montant` ou `taux` est négatif, la fonction doit **lever une erreur**.
    3. Tester avec des valeurs extrêmes (ex: taux de **0**, montant très grand, etc.).

Exercice 4 : Tester un module de gestion d'utilisateurs avec des Mocks

Tu dois tester une fonction qui récupère des utilisateurs depuis une base de données simulée.

- La fonction `getUserById(id)` doit renvoyer un utilisateur en fonction de son **ID**.
- **Utiliser un mock** pour simuler une base de données.

Exercice 5 : Test de non-régression sur une fonction de calcul de TVA

- Implémente une fonction `calculerTVA(prix, taux)` et vérifie qu'un **changement accidentel dans le code** n'introduit pas une erreur de calcul.

Exercicce 6: Tester un formulaire de connexion

1. Crée une page avec un formulaire `<input type="email">` et `<input type="password">`.
2. Simule un **utilisateur qui entre un mauvais mot de passe**, puis un bon.
3. Vérifie que l'utilisateur est bien redirigé après connexion réussie.