**Behavior Driven Development (BDD) : Comprendre et Pratiquer**

**(Ref audio, BDD comprendre et pratiquer.mp4)**

|  |  |
| --- | --- |
| **Contexte** | **Définition, Exemple, Explication** |
| **Behavior Driven Development (BDD)** | Le BDD est une méthode de développement axée sur la collaboration et la compréhension claire du comportement des fonctionnalités. Il utilise des scénarios en langage naturel. |
| **Exemple** | Un chef de projet demande une fonctionnalité de commentaires, et le BDD implique la définition de scénarios tels que la gestion de la taille des commentaires, des gros mots, et des conditions d'accès. |
| **Explication** | La méthodologie commence par la collaboration et utilise une syntaxe standardisée avec des mots clés (Given, When, Then, And) pour décrire clairement les comportements attendus. |
| **Pratique du BDD** | Appliqué à divers scénarios, par exemple, pour la localisation d'un site où le BDD spécifie que les utilisateurs espagnols voient la version espagnole de la page d'accueil. Il rend les spécifications plus compréhensibles et facilite l'écriture des tests. |
| **Avantages** | Favorise un développement axé sur le comportement, améliore la clarté et la collaboration au sein de l'équipe de développement. |

**Apprenez le test drivent development (TDD)**

|  |  |
| --- | --- |
| **Contexte** | **Définition, Exemple, Explication** |
| **Test-Driven Development (TDD)** | Méthode de développement où les tests sont écrits avant le code. |
| **Cycle Rouge-Vert-Refactor** | Processus itératif de TDD comprenant :  **Rouge :** Écrire un test qui échoue.  **Vert :** Écrire le code minimal pour faire passer le test.  **Refactor :** Améliorer le code sans changer son comportement. |
| **Règles de BDD selon Robert Martin** | 1. Écrire un test qui échoue avant le code. 2. Ne pas écrire un test plus compliqué que nécessaire. 3. Ne pas écrire plus de code que nécessaire pour faire passer le test. |
| **Exemple de Test** | ```javascript  describe("Saving a comment", function() {  it("should throw an error if the comment is over 1000 characters", function() {  comment.length = 1200;  expect(function() {  comment.save();  }).toThrowError("comment is too long");  });  }); |
|  |  |
| **Cycle Rouge-Vert-Refactor (suite)** | Le test est écrit, il échoue initialement (Rouge).  Le code minimal est ajouté pour faire passer le test (Vert).  Refactor : Amélioration du code sans altérer le comportement attendu par le test. |
| **Évolution des Tests** | Les tests deviennent plus spécifiques pour des scénarios plus détaillés.  Le code devient plus général pour accommoder différents scénarios. |
| **Échec Bénéfique** | Dans le TDD, l'échec initial des tests est bénéfique, guidant le développement itératif. |

|  |  |
| --- | --- |
| **En résumé BDD et TDD ensemble :** | |
| **BDD** | * Etapes initiale de testing ou comportement fonctionnalité sont attendu * Utilisation mot clé Etant donné, quand, Alors, Et (Given, when, Then, And) * Comportement attendu guident processus d’écriture du code |
| **TDD** | * Etapes suivantes ou tests sont écrit pour comportement définis précédemment * Test échouent car code de la fonctionnalité est pas écrit * Choix syntaxe et structure des test dépend du langage de prog * Processus itératif « Cycle Rouge Vert Refactor »   Ecrire test qui echoue (rouge), ajout code minimal pour la faire passer (vert) puis modif code sans altérer comportement (refactor) |
| **Application au test Front** | |
| * Utilisation méthodologie BDD et TDD pour le dev frontend * Echec initial des tests pour considérer comme benef, guidant le développement itératif des fonctionnalités * Processus de détailler les comportements, écrire des tests, et dev le code selon cycle Rouge-Vert-Refactor | |

**Définissez des suites et des spécifications**