UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Développement en BDD

Les paradigmes du BDD

Le **Behavior Driven Development (BDD)** est une pratique de développement de logiciel qui met l'accent sur la **collaboration entre les différentes parties prenantes d'un projet** (comme les développeurs, les testeurs, les responsables produit, etc.) et l'**explication du comportement du système en termes compréhensibles par tous**. Il est souvent utilisé dans le développement Agile

- **Communication et collaboration** : Le BDD insiste sur la nécessité d'une collaboration étroite entre toutes les parties prenantes du projet. Il met l'accent sur le "partage des connaissances", en veillant à ce que tout le monde comprenne le comportement désiré du système.
- **Développement basé sur le comportement** : Comme son nom l'indique, le BDD met l'accent sur **le comportement du système** plutôt que sur les détails techniques de son implémentation. Il s'agit de décrire **ce que le système doit faire de manière compréhensible par tous**, et non de se concentrer sur la manière dont ces fonctionnalités seront mises en œuvre.
- **Spécification exécutable** : Le BDD utilise un **langage naturel** pour **décrire le comportement du système**. Ces descriptions sont ensuite utilisées comme spécifications exécutables, qui peuvent être exécutées comme tests. Cela signifie que les spécifications servent à la fois de documentation et de vérification du système.

Les paradigmes du BDD

- **Tests orientés comportement** : Le BDD utilise un format spécifique pour les tests, connu sous le nom de "**Given-When-Then**" (Étant donné - Quand - Alors). Cela décrit le contexte (Given), l'action qui est effectuée (When), et le résultat attendu (Then). Cela aide à structurer les tests de manière à ce qu'ils reflètent le comportement désiré du système.
- **Développement itératif** : Comme d'autres pratiques Agile, le BDD suit **une approche itérative du développement**. Il s'agit de construire progressivement le système, en ajoutant un comportement à la fois, et en vérifiant constamment que le système se comporte comme prévu.

Les bonnes pratiques du BDD

- **Définissez clairement les comportements** : Vos spécifications **devraient décrire clairement le comportement attendu du système**. Elles ne devraient pas se concentrer sur les détails techniques de l'implémentation, mais plutôt sur **ce que l'utilisateur peut faire et ce qu'il peut s'attendre à voir**.
- **Utilisez le format Given-When-Then** : Ce format est un excellent **moyen de structurer vos scénarios de manière claire et compréhensible**. Il vous aide à vous concentrer sur le contexte (Given), l'action (When) et le résultat (Then). Il est compatible avec le français (**Étant donné-Quand-Alors**)
- **Gardez vos scénarios courts et concentrés** : Chaque scénario doit **tester une seule fonctionnalité ou comportement**. S'il y a trop de choses dans un seul scénario, il devient difficile à comprendre et à maintenir.
- **Automatisez vos scénarios** : Les scénarios BDD doivent **être automatisés pour pouvoir les exécuter régulièrement**. Cela vous permet de vérifier rapidement que votre système se comporte toujours comme prévu, même après des modifications.
- **Revoyez et affinez vos scénarios régulièrement** : Comme tout autre aspect de votre système, vos scénarios BDD devraient être revus et affinés régulièrement. Cela vous aide à maintenir leur pertinence et leur utilité.

Cucumber

Cucumber : Cucumber est l'un des frameworks les plus populaires pour le BDD. Il vous permet d'**écrire des scénarios de tests en langage naturel** qui peuvent être **exécutés comme des tests automatisés**. Cucumber dispose d'une **intégration étroite avec Spring**, ce qui vous permet d'utiliser des fonctionnalités comme l'injection de dépendances dans vos tests.

Gherkin

Gherkin est un langage de spécification utilisé dans le BDD pour écrire des scénarios de tests de manière lisible par les humains.
Il utilise une syntaxe simple et naturelle pour décrire les comportements attendus sous forme de "Features" (fonctionnalités) et de "Scenarios" (scénarios).

Mot Anglais	Mot Français	Description
Feature	**Fonctionnalité**	Description de la fonctionnalité à tester.
Scenario	**Scénario**	Exemple concret illustrant une fonctionnalité.
Given	**Étant donné que**	Contexte initial du scénario.
When	**Quand**	Action ou événement déclencheur.
Then	**Alors**	Résultat attendu après l'action.
And/But	**Et/Mais**	Étapes supplémentaires dans le scénario.

Gherkin

```
Feature: Jeu du Pendu

  Scenario: Proposition correcte d'une lettre
    Given le mot à deviner est "chat"
    And l'état actuel du mot est "_ _ _ _"
    When le joueur propose la lettre "a"
    Then l'état actuel du mot doit être "_ _ a _"
    And le nombre de tentatives restantes est inchangé

  Scenario: Proposition incorrecte d'une lettre
    Given le mot à deviner est "chat"
    And l'état actuel du mot est "_ _ _ _"
    When le joueur propose la lettre "z"
    Then l'état actuel du mot doit rester "_ _ _ _"
    And le nombre de tentatives restantes doit être diminué de 1

  Scenario: Le joueur gagne en devinant toutes les lettres
    Given le mot à deviner est "chat"
    And l'état actuel du mot est "c h a _"
    When le joueur propose la lettre "t"
    Then l'état actuel du mot doit être "c h a t"
    And le joueur doit voir un message de victoire

  Scenario: Le joueur perd après avoir épuisé toutes les tentatives
    Given le mot à deviner est "chat"
    And le nombre de tentatives restantes est 1
    When le joueur propose la lettre "z"
    Then le joueur doit voir un message de défaite
    And le mot complet "chat" doit être révélé
```

```
Fonctionnalité: Jeu du Pendu

  Scénario: Proposition correcte d'une lettre
    Étant donné que le mot à deviner est "chat"
    Et que l'état actuel du mot est "_ _ _ _"
    Quand le joueur propose la lettre "a"
    Alors l'état actuel du mot doit être "_ _ a _"
    Et le nombre de tentatives restantes est inchangé

  Scénario: Proposition incorrecte d'une lettre
    Étant donné que le mot à deviner est "chat"
    Et que l'état actuel du mot est "_ _ _ _"
    Quand le joueur propose la lettre "z"
    Alors l'état actuel du mot doit rester "_ _ _ _"
    Et le nombre de tentatives restantes doit être diminué de 1

  Scénario: Le joueur gagne en devinant toutes les lettres
    Étant donné que le mot à deviner est "chat"
    Et que l'état actuel du mot est "c h a _"
    Quand le joueur propose la lettre "t"
    Alors l'état actuel du mot doit être "c h a t"
    Et le joueur doit voir un message de victoire

  Scénario: Le joueur perd après avoir épuisé toutes les tentatives
    Étant donné que le mot à deviner est "chat"
    Et que le nombre de tentatives restantes est 1
    Quand le joueur propose la lettre "z"
    Alors le joueur doit voir un message de défaite
    Et le mot complet "chat" doit être révélé
```