Strategie de test

# Introduction

Dans le cadre de notre stratégie, nous visons à atteindre une fiabilité sans faille, assurant que l'application fonctionne de manière fluide et sans erreurs, pour que rien ne vienne perturber l'expérience utilisateur. Enfin, notre objectif est de garantir une performance exceptionnelle, permettant à l'application de gérer un grand nombre d'utilisateurs et de transactions sans ralentissement, évitant ainsi de se transformer en un dispositif lent et frustrant.

En développant des cas de test détaillés axés sur les fonctionnalités de l'application, nous allons mettre l'accent sur les scénarios d'utilisation réelle ainsi que sur les parcours moins fréquentés. De plus, l'intégration du Behavior-Driven Development (BDD) et du Test-Driven Development (TDD) dans notre processus de conception jouera un rôle clé. Cette approche nous permettra de mieux comprendre les exigences à travers le prisme des comportements utilisateurs et de garantir que chaque aspect de l'application est rigoureusement testé dès les premières étapes du développement, assurant ainsi une qualité et une fiabilité accrues.

# Analyse AMDEC

L'analyse AMDEC sera utilisée pour identifier et évaluer les risques liés à chaque fonctionnalité de l'application. Cette méthode nous permettra de prioriser les tests en fonction de la sévérité et de la probabilité d'occurrence des défaillances potentielles.

# 1. Tests Unitaires

* **Lecteur de CSV (CSVReader**) **:**
  + Test de lecture de fichier CSV : Vérifier que le lecteur de CSV peut lire un fichier existant et en extraire correctement les données.
  + Gestion des erreurs : Tester le comportement en cas de fichier inexistant ou corrompu.
* **Générateur d’équipe (TeamGenerator) :**
  + Génération des équipes : Confirmer que les équipes sont correctement générées à partir d'une liste de joueurs, avec le nombre de joueurs par équipe spécifié.
  + Répartition aléatoire : Assurer que la répartition des joueurs dans les équipes est aléatoire.
* **Générateur de tournois (TournamentGenerator) :**
  + Génération des poules *:* S'assurer que les poules sont correctement générées avec un nombre approprié d'équipes.
  + Simulation des matchs de poules : Vérifier que la simulation des matchs de poules qualifie correctement les équipes pour les phases finales.
  + Génération des phases finales : Tester la logique de génération des phases finales pour confirmer que le tournoi avance correctement jusqu'à désigner un vainqueur.

# 2. Tests d’Intégration

* Intégration CSVReader et TeamGenerator : Tester le flux de travail depuis la lecture des données d'un fichier CSV jusqu'à la génération des équipes.
* Intégration TeamGenerator et TournamentGenerator : Valider l'intégration en vérifiant que les équipes générées peuvent être utilisées pour générer un tournoi complet.

# 3. Tests Fonctionnels

* Flux complet de génération de tournoi : Exécuter un test couvrant le scénario complet depuis la lecture du fichier CSV, la génération des équipes, jusqu'à la création et l'exécution du tournoi. Vérifier que l'application produit le résultat attendu et gère correctement les différentes étapes.

# 4. Tests de Performance

Gestion des grands nombres : Tester l'application avec un grand nombre de joueurs et d'équipes pour évaluer les performances et identifier d'éventuels goulets d'étranglement.