

Analyse de la testabilité de la classe QuiEstCeClient

Constructeur QuiEstCeClient(hôte, port)

Tableau de synthèse

Critère	Note	Justification synthétique
Observabilité	★★☆☆☆ (2/5)	Aucun effet visible (aucune sortie ni log, initialisation silencieuse)
Contrôlabilité	★★★☆☆ (3/5)	Paramètres contrôlables (hôte, port), mais connexion interne opaque.
Disponibilité	★★★★☆ (4/5)	Constructeur public accessible dans tout contexte avec hôte/port valides.
Stabilité	★★★★☆ (4/5)	Déterministe : mêmes hôte/port → mêmes résultats

Analyse détaillée

Critère	Limites observées	Pistes d'amélioration
Observabilité	Aucune sortie ni log (observabilité faible).	Ajouter un log ou un champ d'état exposé pour la connexion (politique de log).
Contrôlabilité	Dépend du serveur réseau en arrière-plan ; difficile de simuler tous les cas.	Permettre l'injection d'un serveur simulé ou un mode hors-ligne pour tester sans serveur réel.
Disponibilité	Méthode publique sans restrictions (bonne disponibilité).	-
Stabilité	Comportement identique pour mêmes paramètres (stable).	Utiliser un serveur de test figé pour garantir la reproductibilité.

Fiche de testabilité — requeteCreationJoueur(nom, prenom)

1. Tableau résumé des critères

Critère	Évaluation	Justification synthétique
Observabilité	(4/5)	Le retour de la méthode est exploitable et son effet peut être vérifié
Contrôlabilité	(4/5)	Il est possible de forcer l'état de départ du serveur et de gérer les entrées
Disponibilité	(4/5)	La méthode peut être invoquée dans tous les contextes de test
Stabilité	(4/5)	Les résultats sont reproductibles sous réserve de réinitialiser l'environnement

2. Détail des critères + pistes

Critère	Analyse détaillée	Pistes d'amélioration éventuelles
Observabilité	La méthode retourne un objet représentant l'identifiant du joueur. Il est également possible de vérifier indirectement l'effet de l'appel via d'autres méthodes (<code>requeteJoueur</code> , <code>requeteJoueurs</code>). L'effet de l'appel est donc visible.	Ajouter des messages d'erreur explicites ou des logs serveur en cas d'échec silencieux
Contrôlabilité	Les paramètres sont intégralement contrôlables. Il est également envisageable de réinitialiser le serveur ou la base de données entre deux appels pour forcer un état de départ. Cela permet de simuler des cas limites (joueur déjà existant, base vide, etc.).	Fournir un outil de réinitialisation automatique pour les tests
Disponibilité	Le code client et serveur étant accessibles, la méthode est testable aussi bien en local qu'en test intégré. Aucun obstacle technique ne limite son usage.	Aucun
Stabilité	Tant que la base de test est nettoyée entre les exécutions, les tests sont reproductibles. Si ce n'est pas le cas, les tests risquent d'échouer à cause de conflits d'état (joueur déjà existant).	Utiliser une base de test volatile ou des scripts de nettoyage

3. Évaluation finale

Critère	Note /5	Risques si non maîtrisé
Observabilité		Risque d'échec silencieux sans log
Contrôlabilité		Risque de dépendance à l'état serveur
Disponibilité		Aucun
Stabilité		Risque de tests non reproductibles

Fiche de testabilité — requeteJoueurs()

1. Tableau de synthèse des critères

Critère	Évaluation	Justification synthétique
Observabilité	(4/5)	Le retour est une liste d'identifiants, directement exploitable dans les assertions
Contrôlabilité	(4/5)	L'état du serveur peut être préparé en amont via des appels à requeteCreationJoueur
Disponibilité	(4/5)	Accessible dans tous les contextes, notamment les tests d'intégration
Stabilité	(4/5)	Retour stable si l'environnement est bien maîtrisé

2. Analyse détaillée par critère

Critère	Analyse détaillée	Pistes d'amélioration éventuelles
Observabilité	La méthode retourne une List<Int> claire, sans ambiguïté. Il est facile de comparer les valeurs attendues et obtenues.	Aucun besoin particulier d'amélioration
Contrôlabilité	L'état observable dépend des joueurs déjà enregistrés. Il est possible de préparer l'environnement à l'avance via des appels à requeteCreationJoueur.	Prévoir une fonction de reset de la base entre les tests
Disponibilité	Ne dépend d'aucune donnée d'entrée, la méthode est testable même seule, dès lors que le serveur est accessible.	Aucun
Stabilité	Si la base est nettoyée ou connue à l'avance, la méthode retourne toujours les mêmes valeurs pour un scénario donné.	Nettoyer la base entre les tests pour garantir la stabilité

3. Évaluation synthétique

Critère	Note /5	Risques si non maîtrisé
Observabilité		Aucun risque majeur
Contrôlabilité		Si des joueurs aléatoires ont été créés, les tests deviennent moins prédictibles
Disponibilité		Aucun
Stabilité		Risque de tests fragiles si la base varie

Fiche de testabilité — requeteJoueur()

1. Tableau de synthèse des critères

Critère	Note sur 5	Justification synthétique
Observabilité	★★★★☆	Retourne un objet Joueur contenant des données lisibles et vérifiables.
Contrôlabilité	★★★★☆	Le paramètre idJoueur est totalement maîtrisable dans le test.
Disponibilité	★★★★☆	Méthode accessible dès lors que le serveur est disponible.
Stabilité	★★★★☆	Résultat stable si la base de test ne change pas entre les exécutions.

2. Détail des critères + pistes

Critère	Limites observées	Pistes d'amélioration
Observabilité	L'objet Joueur est observable, mais en cas d'erreur (ex : id inconnu), la méthode peut échouer silencieusement si non gérée.	Ajouter un retour clair ou une exception documentée en cas d'échec.
Contrôlabilité	L'ID est simple à manipuler. Il est possible de créer un joueur au préalable pour garantir que l'ID existe.	Prévoir une méthode de création dédiée dans le setup du test.
Disponibilité	Aucun paramètre secret, méthode indépendante des autres si un joueur existe.	Aucun besoin particulier.
Stabilité	Si un joueur est supprimé ou modifié entre deux tests, les résultats changent.	Réinitialiser la base de données ou la peupler toujours de la même façon avant chaque test.

3. Évaluation finale

Critère	Étoiles	Risques en cas de mauvaise gestion
Observabilité	★★★★☆	Risque : échec silencieux si l'ID n'existe pas ou que l'erreur n'est pas captée.
Contrôlabilité	★★★★☆	Risque : ID inexistant si le setup est mal préparé.
Disponibilité	★★★★☆	Risque : indisponibilité du serveur ou ID utilisé dans une autre session.
Stabilité	★★★★☆	Risque : changement de données si base non figée.

Fiche de testabilité — requeteCreationPartie()

1. Tableau de synthèse des critères

Critère	Note sur 5	Justification synthétique
Observabilité	★ ★ ★ ☆	Retourne un ID de partie. Cependant, aucune information sur l'état ou la création n'est fournie.
Contrôlabilité	★ ★ ☆☆☆	Le test dépend de la validité du joueur et de sa clé. L'état serveur doit être bien préparé.
Disponibilité	★ ★ ★ ☆	Accessible uniquement après une authentification réussie. Nécessite un joueur valide.
Stabilité	★ ☆☆☆☆	Chaque appel crée une nouvelle partie avec un ID différent. Les résultats varient entre exécutions.

2. Détail des critères + pistes

Critère	Limites observées	Pistes d'amélioration
Observabilité	La méthode retourne uniquement un ID de partie sans détail sur la création.	Retourner un objet contenant aussi le nom du joueur, le statut de la partie, etc.
Contrôlabilité	Le test nécessite un idJoueur + cleJoueur corrects. Impossible de créer une partie si le joueur est invalide.	Créer un joueur dédié dans le setup du test (avec ID + clé connus) avant l'appel.
Disponibilité	Méthode indisponible si aucun joueur n'a été créé ou si la clé est incorrecte.	Ajouter une méthode de setup ou utiliser un joueur de test connu.
Stabilité	L'ID de partie étant généré dynamiquement, il change à chaque exécution.	Comparer le comportement (création OK ou échec), pas la valeur exacte de l'ID retourné.

3. Évaluation finale

Critère	Étoiles	Risques en cas de mauvaise gestion
Observabilité	★ ★ ★ ☆	Risque : pas de moyen de vérifier la validité ou l'état de la partie créée.
Contrôlabilité	★ ★ ☆☆☆	Risque : échec du test si l'ID ou la clé du joueur sont absents ou incorrects.
Disponibilité	★ ★ ★ ☆	Risque : méthode inutilisable sans authentification préalable.
Stabilité	★ ☆☆☆☆	Risque : tests non reproductibles si on compare directement les ID générés.

Fiche de testabilité — requeteEtatPartie()

1. Tableau de synthèse des critères

Critère	Note sur 5	Justification synthétique
Observabilité	★★★★☆	L'état retourné est exploitable et observable sous forme d'un objet structuré.
Contrôlabilité	★★★☆☆	Il est possible de manipuler idPartie, mais difficile de provoquer tous les états sans coordination.
Disponibilité	★★★★☆	La méthode est publique et accessible dès qu'un ID de partie valide est connu.
Stabilité	★☆☆☆☆	Le résultat dépend du déroulement de la partie (non stable si la partie évolue en temps réel).

2. Détail des critères + pistes

Critère	Limites observées	Pistes d'amélioration
Observabilité	L'objet EtatPartie est observable. En cas d'erreur (ex : partie inexistante), le retour peut être peu explicite.	Ajouter une réponse claire ou une exception bien identifiée en cas d'échec.
Contrôlabilité	Il est difficile de provoquer un état précis (ex : partie en attente vs en cours vs terminée) sans séquence complexe.	Prévoir une méthode de simulation ou des scénarios prédéfinis en base de test.
Disponibilité	Accessible dès qu'un ID de partie est connu.	Aucun problème majeur.
Stabilité	Si un autre test modifie l'état de la partie (ex. un joueur rejoint), le résultat change.	Travailler avec une base isolée ou une partie stable pour le test.

3. Évaluation finale

Critère	Étoiles	Risques en cas de mauvaise gestion
Observabilité	★★★★☆	Risque : erreur peu lisible si l'ID de la partie n'existe pas.
Contrôlabilité	★★★☆☆	Risque : état difficile à forcer (risque d'ambiguïté dans les assertions).
Disponibilité	★★★★☆	Risque : test bloqué si l'ID de partie est inconnu ou incorrect.
Stabilité	★☆☆☆☆	Risque : test non reproduitible si la partie évolue entre les appels.

Fiche de testabilité — requête RejoindrePartie()

1. Tableau de synthèse des critères

Critère	Note sur 5	Justification synthétique
Observabilité	★★★★☆	Retourne un objet complet (EtatPartie), ce qui rend le résultat bien observable.
Contrôlabilité	★★☆☆☆	Dépend de conditions externes : la partie doit exister et être dans un état précis (attente).
Disponibilité	★★★☆☆	Nécessite un joueur et une partie valides + partie non encore complète.
Stabilité	★★☆☆☆	Le résultat varie selon l'état courant de la partie (peut évoluer rapidement).

2. Détail des critères + pistes

Critère	Limites observées	Pistes d'amélioration
Observabilité	L'objet EtatPartie retourné contient des informations utiles (joueurs inscrits, état).	Ajouter un journal côté client pour confirmer l'identité du joueur ayant rejoint.
Contrôlabilité	La méthode ne fonctionnera que si la partie est dans un état précis (en attente d'un second joueur).	Créer la partie juste avant l'appel et s'assurer qu'elle n'est pas encore commencée.
Disponibilité	Le test est impossible si aucun joueur valide n'existe ou si la partie est déjà lancée.	S'assurer de créer un joueur de test et une partie fraîche à chaque exécution.
Stabilité	Un test lancé deux fois peut échouer si la partie a déjà été rejoindre entre-temps par un autre joueur.	Lancer les tests dans un environnement isolé ou en séquence rigide.

3. Évaluation finale

Critère	Étoiles	Risques en cas de mauvaise gestion
Observabilité	★★★★☆	Risque : pas de message clair si la partie est pleine ou invalide.
Contrôlabilité	★★☆☆☆	Risque : difficile de garantir que la partie est encore "rejoignable" au moment du test.
Disponibilité	★★★☆☆	Risque : le test dépend d'un joueur authentifié et d'une partie dans un bon état.
Stabilité	★★☆☆☆	Risque : si la partie change d'état avant le test, le résultat devient non prédictible.

Fiche de testabilité — requeteChoixPersonnage()

1. Tableau de synthèse des critères

Critère	Note sur 5	Justification synthétique
Observabilité	★★★★☆	Retourne un EtatPartie contenant des infos vérifiables (joueurs, tour, statut...).
Contrôlabilité	★★★★☆	Le joueur choisit les coordonnées, et peut contrôler le moment où il effectue le choix.
Disponibilité	★★★★☆	La méthode est accessible une fois la partie lancée et les joueurs connectés.
Stabilité	★★★★☆	Pour un contexte donné (même grille, joueur et coordonnées), le résultat est stable.

2. Détail des critères + pistes

Critère	Limites observées	Pistes d'amélioration
Observabilité	Le retour donne bien l'état de la partie, mais ne précise pas directement le personnage choisi.	Ajouter dans EtatPartie une confirmation explicite du personnage sélectionné.
Contrôlabilité	Les coordonnées (ligne/colonne) sont contrôlables. Le test peut forcer des cas valides/invalides.	Simuler une grille connue dans les tests pour vérifier si les coordonnées sont valides.
Disponibilité	Nécessite une partie en cours, deux joueurs inscrits et un tour précis.	Utiliser un scénario automatisé : création joueur → création partie → rejoindre partie.
Stabilité	Le résultat est déterministe si l'état initial est toujours le même.	Réinitialiser la partie avant chaque test ou utiliser un ID de partie temporaire.

3. Évaluation finale

Critère	Étoiles	Risques en cas de mauvaise gestion
Observabilité	★★★★☆	Risque : difficile de confirmer dans les logs le personnage réellement sélectionné.
Contrôlabilité	★★★★☆	Risque : coordonnées non valides si la grille n'est pas connue ou si l'état de la partie a changé.
Disponibilité	★★★★☆	Risque : test impossible si la partie n'est pas prête ou si le joueur n'est pas dans le bon état.
Stabilité	★★★★☆	Risque : résultat variable si plusieurs tests modifient la même partie sans reset.

Fiche de testabilité — requeteGrilleJoueur()

1. Tableau de synthèse des critères

Critère	Note sur 5	Justification synthétique
Observabilité	★★★★☆	Retourne une structure exploitable (List<List>) : grille directement observable.
Contrôlabilité	★★★★☆	On peut contrôler la partie et le joueur, mais pas directement l'état interne de la grille.
Disponibilité	★★★★☆	Méthode accessible dès que le joueur et la partie sont valides.
Stabilité	★☆☆☆☆	Le contenu de la grille change en fonction du déroulement du jeu (ex : questions posées).

2. Détail des critères + pistes

Critère	Limites observées	Pistes d'amélioration
Observabilité	La grille retournée est lisible et structurée. On peut observer les personnages affichés.	Ajouter éventuellement un identifiant de version ou d'étape de partie pour aider au suivi.
Contrôlabilité	L'appel dépend de l'état du jeu. On ne peut pas imposer directement le contenu de la grille.	Préparer la partie à l'avance en la plaçant dans un état connu (par ex. aucun filtrage encore appliqué).
Disponibilité	La méthode est toujours disponible une fois la partie commencée.	Aucun problème majeur.
Stabilité	La grille évolue au fil du jeu. Le même appel peut donner un résultat différent à cause des filtres appliqués.	Réinitialiser la partie avant chaque test ou utiliser un scénario fixe reproductible.

3. Évaluation finale

Critère	Étoiles	Risques en cas de mauvaise gestion
Observabilité	★★★★☆	Faible lisibilité si la structure Personnage est trop complexe ou non affichable.
Contrôlabilité	★★★★☆	Risque : grille imprévisible si des actions précédentes l'ont modifiée.
Disponibilité	★★★★☆	Aucun risque si les identifiants sont bien préparés.
Stabilité	★☆☆☆☆	Risque : résultats non reproductibles sans reset du jeu entre les tests.

Fiche de testabilité — requetePoserQuestion()

1. Tableau de synthèse des critères

Critère	Note sur 5	Justification synthétique
Observabilité	★★★★☆	Retourne un EtatPartie contenant des informations sur l'état actuel du jeu après la question.
Contrôlabilité	★★★★☆	Le joueur contrôle entièrement la question posée et le moment de l'appel.
Disponibilité	★★★★☆	Méthode accessible uniquement lorsque c'est au tour du joueur de poser une question.
Stabilité	★★★★☆	Stable : une même question posée dans un même contexte produit la même modification de l'état.

2. Détail des critères + pistes

Critère	Limites observées	Pistes d'amélioration
Observabilité	L'état retourné permet de vérifier le passage de tour, mais ne contient pas toujours l'historique exact de la question.	Ajouter un champ dans EtatPartie pour visualiser la dernière question posée.
Contrôlabilité	Le texte de la question est maîtrisé, mais le contexte (ex. moment dans le tour) doit être correctement préparé.	Encadrer l'appel dans un scénario prédéfini : début de partie, tour connu, etc.
Disponibilité	Si ce n'est pas le tour du joueur, la méthode renvoie un refus.	Forcer un état de partie adéquat pour garantir que c'est bien au tour du joueur.
Stabilité	La réponse dépend uniquement de la question et du personnage secret adverse.	Prévoir des tests avec personnage cible connu pour vérifier la réponse retournée.

3. Évaluation finale

Critère	Étoiles	Risques en cas de mauvaise gestion
Observabilité	★★★★☆	Risque : sans log ou suivi, difficile d'associer une réponse à une question précise.
Contrôlabilité	★★★★☆	Risque : test invalide si ce n'est pas le bon moment dans le tour de jeu.
Disponibilité	★★★★☆	Risque : erreur d'exécution si le contexte du tour n'est pas respecté.
Stabilité	★★★★☆	Risque : instabilité uniquement si le personnage secret change entre les tests.

Fiche de testabilité — requeteDonnerReponse()

1. Tableau de synthèse des critères

Critère	Note sur 5	Justification synthétique
Observabilité	★★★★☆	L'objet EtatPartie retourné est structuré et permet de vérifier que la réponse a bien été prise en compte.
Contrôlabilité	★★★★☆	Le testeur contrôle la réponse ("oui"/"non"), et les identifiants.
Disponibilité	★★★★☆	Accessible uniquement après qu'une question a été posée et que c'est au tour du joueur de répondre.
Stabilité	★★★★☆	Stable si le personnage cible et l'état de partie ne changent pas.

2. Détail des critères + pistes

Critère	Limites observées	Pistes d'amélioration
Observabilité	La réponse n'est pas explicitement inscrite dans l'état retourné (pas de trace directe de la réponse donnée).	Ajouter un champ dans EtatPartie listant la dernière question + réponse fournie.
Contrôlabilité	Le paramètre reponse est contrôlable et limité (via l'UI aux réponses valides).	Permettre dans les tests d'exécuter la méthode avec des réponses simulées.
Disponibilité	Nécessite que la méthode soit appelée au bon moment dans le déroulement du tour.	Créer un scénario de test où une question a été posée juste avant.
Stabilité	Identique si les conditions de jeu sont constantes.	Réinitialiser la partie avant chaque test pour éviter les dépendances contextuelles.

3. Évaluation finale

Critère	Étoiles	Risques en cas de mauvaise gestion
Observabilité	★★★★☆	Risque : difficile de prouver que la réponse a été acceptée sans trace explicite.
Contrôlabilité	★★★★☆	Risque : test impossible si aucun scénario préalable n'a posé de question.
Disponibilité	★★★★☆	Risque : méthode inaccessible hors du bon tour ou si aucune question n'est en attente.
Stabilité	★★★★☆	Risque : état de partie instable si les tests modifient le tour ou les personnages.

Fiche de testabilité — requeteChercherEncore()

1. Tableau de synthèse des critères

Critère	Note sur 5	Justification synthétique
Observabilité	★★★☆	Le retour est un objet EtatPartie, mais il ne précise pas directement que "ChercherEncore" a été invoqué.
Contrôlabilité	★★★☆	Appel possible uniquement dans un contexte précis (après une réponse) ; dépend donc de l'état du jeu.
Disponibilité	★★★☆	Nécessite que la méthode soit appelée juste après une réponse donnée par l'adversaire.
Stabilité	★★★★☆	Comportement stable si le contexte du tour de jeu est bien contrôlé.

2. Détail des critères + pistes

Critère	Limites observées	Pistes d'amélioration
Observabilité	L'action "chercher encore" n'est pas explicitement marquée dans l'objet EtatPartie retourné.	Ajouter un champ de type "dernièreAction" dans l'objet retourné pour faciliter l'observation.
Contrôlabilité	Le testeur ne peut pas forcer directement cette étape sans passer par les actions précédentes.	Préparer un scénario fixe : poser question → recevoir réponse → appeler chercherEncore.
Disponibilité	Impossible d'appeler cette méthode si ce n'est pas le bon moment (risque d'erreur silencieuse).	Ajouter une exception claire ou un code d'erreur si la méthode est appelée au mauvais moment.
Stabilité	Stable si l'environnement est correctement contrôlé.	Réinitialiser la partie entre les tests pour garantir la reproductibilité.

3. Évaluation finale

Critère	Étoiles	Risques en cas de mauvaise gestion
Observabilité	★★★☆	Risque : impossible de vérifier que l'action a été comprise sans modification explicite de l'état.
Contrôlabilité	★★★☆	Risque : test échouant silencieusement si la méthode est appelée dans un mauvais contexte.
Disponibilité	★★★☆	Risque : méthode inaccessible si aucune réponse n'a été donnée avant.
Stabilité	★★★★☆	Risque : instabilité faible, dépendant uniquement de l'état du tour de jeu.

Fiche de testabilité — requête Trouve()

1. Tableau de synthèse des critères

Critère	Note sur 5	Justification synthétique
Observabilité	★★★★☆	Le résultat (true/false) est clairement observable. Mais aucune trace directe de la tentative n'est gardée.
Contrôlabilité	★★★★☆	Le joueur choisit précisément la position (ligne, colonne), donc contrôle total sur l'action.
Disponibilité	★★★★☆	Méthode accessible à tout moment de la partie, mais souvent réservée à un moment stratégique.
Stabilité	★★★★☆	Si le personnage cible ne change pas, la méthode est stable : même coordonnées → même résultat.

2. Détail des critères + pistes

Critère	Limites observées	Pistes d'amélioration
Observabilité	Seul un booléen est retourné. On ne sait pas quel personnage était visé ni si le tour est terminé.	Ajouter un log dans EtatPartie ou une trace serveur pour confirmer la tentative.
Contrôlabilité	Le testeur contrôle les coordonnées et le moment de l'appel.	Utiliser une grille de test connue pour vérifier les bons cas.
Disponibilité	La méthode peut être appelée dès que le joueur a la main.	Préparer des parties simples avec un seul personnage possible pour les tests unitaires.
Stabilité	Le résultat est stable si le personnage secret reste inchangé.	Réinitialiser la partie à chaque test pour garantir reproductibilité.

3. Évaluation finale

Critère	Étoiles	Risques en cas de mauvaise gestion
Observabilité	★★★★☆	Risque : peu d'infos contextuelles — on ne sait pas quel était le personnage visé ni l'état du jeu.
Contrôlabilité	★★★★☆	Risque : test peu pertinent si la grille ou le personnage secret ne sont pas connus.
Disponibilité	★★★★☆	Risque : erreur si la méthode est appelée trop tôt (ex. sans personnage sélectionné).
Stabilité	★★★★☆	Risque : changement de cible secret → résultats non reproductibles sans reset.

Fiche de testabilité — requeteListeParties()

1. Tableau de synthèse des critères

Critère	Note sur 5	Justification synthétique
Observabilité	★★★★☆	Le résultat est observable : une liste d'entiers correspondant aux ID des parties est retournée.
Contrôlabilité	★★☆☆☆	L'utilisateur ne contrôle pas la structure ni le contenu de la base de parties.
Disponibilité	★★★★★	Méthode accessible à tout moment, sans condition préalable.
Stabilité	★★☆☆☆	Le résultat dépend fortement de l'état du serveur et des autres utilisateurs.

2. Détail des critères + pistes

Critère	Limites observées	Pistes d'amélioration
Observabilité	Retourne directement les identifiants des parties, mais sans contexte ou métadonnées.	Ajouter une méthode complémentaire pour obtenir plus d'informations par ID.
Contrôlabilité	Le testeur ne peut pas facilement prédire les parties existantes sauf à lancer une session isolée.	Injecter des parties factices dans un environnement de test local.
Disponibilité	Toujours disponible, aucun prérequis de session ou d'état.	Aucun besoin d'amélioration spécifique.
Stabilité	Si d'autres joueurs créent ou suppriment des parties entre les tests, les résultats peuvent varier.	Utiliser un serveur local ou un mock pour figer l'état du système.

3. Évaluation finale

Critère	Étoiles	Risques en cas de mauvaise gestion
Observabilité	★★★★☆	Risque : l'utilisateur ne peut pas comprendre à quoi correspondent les ID retournés.
Contrôlabilité	★★☆☆☆	Risque : tests non reproductibles si l'environnement serveur est partagé.
Disponibilité	★★★★★	Aucun risque majeur, méthode fiable et accessible.
Stabilité	★★☆☆☆	Risque : résultats différents à chaque exécution selon les activités des autres joueurs.

Fiche de testabilité — requeteListePartiesCreees()

1. Tableau de synthèse des critères

Critère	Note sur 5	Justification synthétique
Observabilité	★★★★☆	Retourne une liste d'identifiants directement vérifiables, correspondant à des parties en attente.
Contrôlabilité	★★☆☆☆	Nécessite que des parties aient été créées préalablement, donc dépend d'un état serveur non figé.
Disponibilité	★★★★★	Méthode accessible en tout temps, sans prérequis d'authentification ou de tour de jeu.
Stabilité	★★☆☆☆	Résultat variable selon les créations de parties (par soi-même ou d'autres utilisateurs).

2. Détail des critères + pistes

Critère	Limites observées	Pistes d'amélioration
Observabilité	Les identifiants sont visibles, mais on ne sait pas à quel joueur ou état précis ils correspondent.	Ajouter des métadonnées comme nom créateur ou date de création.
Contrôlabilité	Nécessite un environnement vierge ou préparé pour garantir un nombre fixe de parties créées.	Injecter des parties en début de test (ex. via requeteCreationPartie).
Disponibilité	Aucun prérequis particulier : utilisable dans tous les contextes.	—
Stabilité	Un même test peut renvoyer un résultat différent si des parties ont été créées entre les appels.	Utiliser un serveur isolé ou un état réinitialisé entre les tests.

3. Évaluation finale

Critère	Étoiles	Risques en cas de mauvaise gestion
Observabilité	★★★★☆	Risque : manque de clarté sur le statut précis de chaque partie retournée.
Contrôlabilité	★★☆☆☆	Risque : tests non reproductibles si d'autres parties existent en parallèle.
Disponibilité	★★★★★	Aucun risque : méthode toujours disponible.
Stabilité	★★☆☆☆	Risque : liste évolutive selon l'activité serveur, difficile à figer.

Fiche de testabilité — requeteListePartiesTerminees()

1. Tableau de synthèse des critères

Critère	Note sur 5	Justification synthétique
Observabilité	★★★★☆	La liste retournée est facilement testable via la vérification des IDs.
Contrôlabilité	★★☆☆☆	Il faut avoir simulé ou provoqué des victoires pour que la méthode retourne des résultats.
Disponibilité	★★★★★ ★	Méthode toujours disponible, indépendante du statut du joueur.
Stabilité	★★☆☆☆	Le résultat varie à chaque fois qu'une partie se termine, donc instabilité si le serveur est partagé.

2. Détail des critères + pistes

Critère	Limites observées	Pistes d'amélioration
Observabilité	Liste simple d'identifiants : pas de précision sur date ou joueur gagnant.	Ajouter un accès aux métadonnées des parties terminées (état final, durée...).
Contrôlabilité	Il faut simuler entièrement une partie jusqu'à la fin pour pouvoir vérifier l'apparition dans la liste.	Créer un script de test qui joue automatiquement une partie jusqu'à la fin.
Disponibilité	Aucun blocage, fonction activable sans aucune condition préalable.	—
Stabilité	Un test identique peut donner un résultat différent si d'autres parties se terminent entre-temps.	Utiliser un environnement serveur réinitialisé pour figer les résultats.

3. Évaluation finale

Critère	Étoiles	Risques en cas de mauvaise gestion
Observabilité	★★★★☆	Risque : la liste ne dit rien du contexte des parties terminées.
Contrôlabilité	★★☆☆☆	Risque : difficile de garantir qu'une partie terminée s'ajoute à la liste sans test automatisé.
Disponibilité	★★★★★	Aucun risque : méthode utilisable dans tous les cas.
Stabilité	★★☆☆☆	Risque : tests fragiles si l'état serveur n'est pas maîtrisé.