Étape 2 : Développement Collaboratif de "TrueNumber Interactif"

Cahier des Charges - Test de Recrutement -

- **Entreprise :** HIGH REFERENCE SARL
- **Projet :** TrueNumber Interactif
- **Version :** 1.0
- **Date :** 17 juillet 2025

---

1. Introduction

Cette deuxième étape du processus de recrutement vise à évaluer la capacité des candidats à travailler en équipe sur un projet existant, à intégrer de nouvelles fonctionnalités complexes et à maintenir les bonnes pratiques de développement. Suite à l'étape 1, les meilleurs candidats seront regroupés pour transformer l'application "TrueNumber" en un jeu multijoueur interactif.

---

2. Objectifs de l'Étape 2

- **Continuité de l'Étape 1 :** Les développeurs devront s'appuyer sur l'application "TrueNumber" développée lors de l'Étape 1.
- **Développement Collaboratif :** Travailler en binôme (un développeur Frontend, un développeur Backend) pour implémenter une nouvelle fonctionnalité majeure.
- **Intégration de Fonctionnalités :** Ajouter la logique de jeu interactive permettant à deux utilisateurs de s'affronter.
- **Qualité et Bonnes Pratiques :** Maintenir un code propre, testable et bien documenté, avec une API accessible via Swagger.
- **Communication et Coordination :** Démontrer une capacité à communiquer efficacement au sein de l'équipe et avec l'administrateur.

---

3. Organisation de l'Équipe et Communication

- **Composition des Groupes :** Chaque groupe sera composé de deux développeurs : un **Développeur Frontend** et un **Développeur Backend**.
- **Création d'un Groupe WhatsApp :** Les deux développeurs de chaque groupe devront créer un groupe WhatsApp et y ajouter l'administrateur.

- **Chef de Groupe :** Le **Développeur Frontend** sera désigné chef de groupe. Il aura la responsabilité de coordonner le travail au sein du groupe et de rapporter les avancées, les défis et les besoins à l'administrateur.

---

4. Durée et Livrables

- **Délai :** 5 jours à compter de la constitution des groupes.
- **Livrables :** À la fin du délai, chaque groupe devra fournir :
    - L'**URL de la documentation Swagger** de l'API Backend mise à jour.
    - L'**URL de l'application Frontend** déployée.
    - **Aucun code source** ne sera demandé à cette étape, l'évaluation se fera sur l'application fonctionnelle et sa documentation API.

---

5. Fonctionnalité à Ajouter : "TrueNumber Interactif"

L'objectif est de transformer le jeu "TrueNumber" en une expérience multijoueur où deux utilisateurs s'affrontent en générant un nombre. Le joueur ayant le nombre le plus élevé gagne la partie.

5.1. Séquence de Jeu Détaillée

1. **Connexion de l'Utilisateur :** Lorsqu'un utilisateur se connecte, il doit pouvoir voir :
    - Les **parties en attente** de joueurs.
    - Un gros bouton **(+)** permettant de **créer une nouvelle partie**.
2. **Création d'une Partie :**
    - En cliquant sur le bouton **(+)**, un formulaire de création de partie s'ouvre.
    - Ce formulaire doit permettre de saisir :
        - La **Mise** : Le nombre de points qui sera ajouté au solde du gagnant et déduit du solde du perdant.
        - Le **Temps de Réflexion** : Le délai (en secondes ou minutes) alloué à chaque joueur pour générer son nombre une fois la partie lancée.
    - Deux boutons : "Créer" (pour valider et créer la partie) et "Annuler".
    - Une fois créée, la partie apparaît dans la liste des parties en attente.
3. **Affichage des Parties en Attente :**
    - Les parties en attente doivent s'afficher sous forme de **boutons cliquables** ou de cartes.
    - Chaque élément doit contenir les informations essentielles de la partie :

        - Nom de l'utilisateur qui l'a créée.
        - Temps de jeu (temps de réflexion configuré).
        - Mise de la partie.
4. **Rejoindre une Partie :**
    - Lorsqu'un utilisateur clique sur une partie en attente, il doit voir les détails de cette partie.
    - Un bouton "Rejoindre" doit être présent.
    - Au clic sur "Rejoindre" :
        - Le système doit **vérifier si le solde de l'utilisateur est suffisant** pour couvrir la mise de la partie.
        - Si le solde est insuffisant, un message d'erreur clair doit s'afficher (ex: "Solde insuffisant pour rejoindre cette partie.").
        - Si le solde est suffisant, l'utilisateur rejoint la partie et la partie passe de l'état "en attente" à "en cours".
5. **Déroulement de la Partie (Une fois deux joueurs) :**
    - La partie débute. L'utilisateur qui a créé la partie (Utilisateur 1) joue en premier.
    - L'**Utilisateur 1** clique sur un bouton "JOUER" pour générer aléatoirement un nombre entre 0 et 100.
    - Un **compte à rebours** doit être visible, indiquant le temps restant pour jouer (correspondant au "Temps de Réflexion" configuré). Si le joueur ne joue pas dans le temps imparti, il perd automatiquement la partie.
    - Une fois que l'Utilisateur 1 a joué, son nombre s'affiche.
    - C'est ensuite au tour du deuxième joueur (Utilisateur 2) de générer son nombre en cliquant sur "JOUER", avec son propre compte à rebours.
6. **Détermination du Vainqueur et Mise à Jour des Soldes :**
    - Lorsque les deux joueurs ont généré leur nombre, le système compare les deux nombres.
    - Le joueur qui a obtenu le **plus grand nombre** est déclaré vainqueur.
    - Un message clair doit s'afficher pour annoncer le gagnant, par exemple : **"[Nom du Gagnant] gagne la partie et reçoit [Mise] points !"**
    - Le solde du gagnant est **crédité de la mise** de la partie.
    - Le solde du perdant est **débité de la mise** de la partie.
    - L'historique des parties doit être mis à jour pour les deux joueurs.

5.2. Améliorations de l'Interface (Frontend)

- L'interface doit clairement distinguer les parties "en attente", "en cours" et "terminées".
- Les informations des parties en attente doivent être facilement lisibles.
- L'ergonomie et la réactivité de l'interface sont toujours primordiales.

---

6. Exigences Techniques

- **Backend (API) :**
    - Développé en **Node.js** (Express, NestJS, ou autre framework JS de votre choix).
    - Utilisation de **MongoDB** comme base de données.
    - Implémentation de nouvelles routes API pour la gestion des parties (création, rejoindre, jouer, consultation des parties).
    - Mise à jour et extension de la documentation **Swagger** pour inclure les nouvelles routes et modèles.
    - Gestion des états de partie (en attente, en cours, terminée).
    - Gestion de la concurrence et des transactions pour la mise à jour des soldes.
    - Considération pour les WebSockets ou Server-Sent Events pour une mise à jour en temps réel de l'état des parties (non obligatoire mais un plus).
- **Frontend :**
    - Développé avec un framework **JavaScript** (Next.js, React, Angular, Vue.js). L'utilisation de **TypeScript** est un atout.
    - Consommation des nouvelles routes API du backend.
    - Mise en place de l'interface utilisateur pour la création, la consultation et la participation aux parties.
    - Gestion des comptes à rebours pour les tours de jeu.
    - Affichage dynamique des messages de victoire/défaite et des mises à jour de solde.
- **Bonnes Pratiques :**
    - Code propre, modulaire et maintenable.
    - Gestion des erreurs robuste.
    - Sécurité des APIs (authentification, autorisation).
    - Versionnement du code sur GitHub (le dépôt de l'étape 1 peut être forké ou un nouveau créé pour le groupe).

---

8. Livrables Finaux (Format du message à l'administrateur)

Le chef de groupe (Développeur Frontend) enverra un message à l'administrateur sous le format suivant :

Nom du Groupe: [Nom du Développeur Frontend] & [Nom du Développeur Backend]
Technologies utilisées: [Technologies Frontend], [Technologies Backend], [Base de Données]
Lien Swagger: https://

Lien Appli: https://
Observations: [Toute observation pertinente sur le développement, les défis rencontrés, les choix techniques, ou les fonctionnalités implémentées.]

Nous attendons avec impatience de voir vos compétences collaboratives et votre capacité à développer des solutions interactives. Bonne chance à tous les groupes !