Compte rendu des modifications apportés au dossier de conception du projet

1. Modifications dans la structure des données

Table `Equipe`

- Ajout de l'attribut `solde` (int) pour gérer l'argent disponible pour les achats de joueurs.

Table `Joueur`

- Ajout de l'attribut `prenom` (varchar) qui semblait être oublié mais était présent dans les maquettes.
- Ajout de l'attribut `aVendre` (boolean) pour indiquer si un joueur est sur le marché des transferts.

Table `History`

- Cette table n'existait pas dans le diagramme de classes initial. Elle a été ajoutée pour stocker l'historique des transferts des joueurs entre les équipes.
- Colonnes ajoutées : `oldEquipe_id`, `newEquipe_id`, `joueur_id`, `prix`, `date`.

Table `Resultat`

- Dans le diagramme de classe initial, `Resultat` était directement lié au `Match`, mais il ne contenait pas de relation explicite pour stocker les scores. La table finale inclut maintenant `score_equipe`, `score_adversaire` et `match_id`.

Table `Composition`

- Elle existait dans le diagramme initial, mais elle a été implémentée avec une clé primaire propre (`id`), et elle relie désormais explicitement les joueurs aux matchs.

2. Modifications fonctionnelles

- **Sélection de l'équipe** : Ajout de la sélection de son équipe qui est un premier pas vers un jeu multijoueur. La sélection nous semblait obligatoire pour pouvoir réaliser des échanges de joueurs au sein du mercato.

- **Gestion des finances de l'équipe** : Ajout de la gestion du solde de l'équipe pour permettre les transactions de transfert.

- **Gestion des matchs et compositions** : Le projet n'intègre pas encore à la fin du premier sprint les fonctionnalités de matchs et compositions bien qu'ils pourront rapidement être ajoutés dans des sprints suivants (Blessures, résultats, staff, composition)

- **Marché des transferts multijoueur**: Une interface spécifique pour mettre les joueurs en vente et acheter de nouveaux joueurs avec une dimension multijoueur entre les équipes.

3. Modifications dans l'implémentation

- **Ajout d'un fichier de configuration (`config.txt`)** pour stocker les informations de connexion à la base de données.

- **Génération d'un `.exe` et d'un `.jar`** pour permettre aux utilisateurs de lancer l'application sans avoir besoin d'un IDE.

- **Guide d'installation mis à jour** pour inclure la procédure de configuration de la base de données et du fichier `config.txt`.

- **Structuration du code** pour faciliter une maintenabilité du code et l'évolution continue, en séparant un maximum les fonctionnalités techniques, métiers et visuel, ce qui permet également un meilleur travail en groupe. La structuration du code étant absente du dossier de conception.

Ces modifications ont permis d’adapter le projet aux exigences fonctionnelles qui n’étaient pas complètement définies dans le cahier des charges initial.