Realtime Elo Ranker

Realtime Elo Ranker est une application web permettant de simuler des matchs entre des joueurs et de calculer et afficher leur classement Elo en temps réel.

Conteneurs Métiers

`apps/realtime-elo-ranker-server` : Serveur de l'application (à implémenter)

- Sera basé sur NestJS
- Gèrera le calcul des résultat de matchs et le classement des joueurs
- Exposera l'API Web pour fournir les fonctionnalités métiers :
  - Calcul des matchs (cf: Classement Elo)
  - Récupération du classement
  - Mise à jour du classement en temps réel via des évènements
  - Création de nouveaux joueurs

`apps/realtime-elo-ranker-simulator` : Simulateur de matchs (à implémenter - bonus)

- Simulera des matchs entre des joueurs
- Enverra les résultats des matchs au serveur
- Vous pouvez utiliser un simple script NodeJS pour simuler les matchs

`apps/realtime-elo-ranker-client` : Client de l'application

- Basé sur NextJS
- Sert une IHM pour :
  - Saisir les résultats des matchs
    - ID du joueur 1
    - ID du joueur 2
    - Résultat du match
  - Afficher le classement des joueurs
  - Créer de nouveaux joueurs

`libs/ui` : Librairie de composants graphiques

- Contiendra les composants graphiques réutilisables pour le client

Lancer l'application cliente

Prérequis :

Le client repose sur une source CSS exposée par la librairie `libs/ui` (@realtime-elo-ranker/libs/ui). Il est nécessaire de construire la lib pour rendre la source accessible.

Pour ce faire :

```
pnpm run libs:ui:build
```

Puis

Lancer l'application :

```
pnpm run apps:client:dev
```

Lancer la doc swagger

```
pnpm run apps:swagger:start
```

Le serveur Swagger sera accessible à l'adresse `http://localhost:3001/api-docs`.

Le serveur est en hot-reload : les modifications apportées au `swagger.yaml` seront automatiquement prises en compte.

Note : Si le live-reload ne fonctionne pas, pensez à forwarder le port `35729` dans l'IDE.

Lancer le mock de l'API

```
pnpm run apps:api-mock:start
```

Le mock de l'API sera accessible à l'adresse `http://localhost:8080`. Il ne doit pas être lancé en même temps que le serveur de l'application.

Architecture

Back for Front

Le serveur de l'application est un serveur BFF (Back for Front) qui expose une API Web pour gérer les fonctionnalités métiers de l'application. Il devra être basé sur NestJS, un framework Node.js pour construire des applications serveur.

Le code serveur est à implémenter dans le dossier `apps/realtime-elo-ranker-server`.

Frontend

Le client de l'application est une application web basée sur Next.js, un framework React pour construire des applications web. Il servira une interface utilisateur pour saisir les résultats des matchs et afficher le classement des joueurs.

Le client devra communiquer avec le serveur via l'API Web pour gérer les fonctionnalités métiers de l'application.

Le classement, une fois restitué dans son état initial par le serveur, sera mis à jour en temps réel sur l'interface utilisateur du client.

Le code client est fourni dans le dossier `apps/realtime-elo-ranker-client`.

API

L'API Web du serveur devra exposer les fonctionnalités telles que décrites dans le swagger fourni.

Mock de l'API

Un mock de l'API est fourni pour tester le client sans avoir à implémenter le serveur. Le mock de l'API répondra aux requêtes du client avec des messages prédéfinis. Il ne gère pas les mises à jour en temps réel du classement.

Règles de gestion

Joueurs

- Un joueur est identifié par un ID unique (entité).
- Un joueur a un classement Elo qui est mis à jour après chaque match (objet valeur).
- Un joueur peut participer à des matchs contre d'autres joueurs.
- Un joueur peut être créé avec un classement initial.
    - Le classement initial par défaut est égal à la moyenne du classement de tous les joueurs existants.

Matchs

- Un match est une confrontation entre deux joueurs.
- Un match a un résultat (victoire, défaite ou égalité).
- Le résultat du match est utilisé pour mettre à jour le classement des joueurs.
- Le classement des joueurs est mis à jour en suivant les fonctions de calcul du classement Elo.

Classement

- Le classement Elo est un système de classement des joueurs basé sur le calcul de la probabilité de victoire de chaque joueur en fonction de leur classement respectif.
- Le classement est une valeur numérique qui représente le niveau de compétence d'un joueur et symbolise un ordre de classement relatif entre les joueurs et un "budget" de points semblable à une monnaie d'échange.

Guide d'implémentation

Serveur

Le serveur de l'application devra être implémenté en utilisant NestJS, un framework Node.js pour construire des applications serveur. Le serveur devra exposer une API Web pour gérer les fonctionnalités métiers de l'application.

Structures de données

Les données des joueurs et des matchs devront être stockées en mémoire dans des structures de données appropriées.

- Les joueurs devront être stockés dans une liste ou un tableau avec leur score
- L'historique des matchs devra être stocké dans une liste ou un tableau.
- Les

Calcul du classement

Le calcul du classement Elo devra être implémenté en suivant les règles mathématiques décrites dans la section Classement Elo.

- Le classement des joueurs devra être mis à jour après chaque match en fonction du résultat du match et de l'écart de classement entre les joueurs.
- Le serveur devra, à la demande, servir un état actuel du classement des joueurs.
- Le serveur devra publier des événements en temps réel pour informer les clients connectés des mises à jour du classement.

Conseils

Ces conseils sont donnés à titre indicatif et ne sont pas exhaustifs. Les suivre n'est pas obligatoire mais peut vous aider à construire une application de qualité.

- Utilisez la POO pour modéliser les structures de données.
- Utilisez la PF (Programmation Fonctionnelle) pour les diverses fonctions de calcul de résultats, les créations d'objets et les .manipulations de données primitives
- Utilisez les services pour encapsuler la logique métier.
- Utilisez les contrôleurs pour gérer les requêtes HTTP.
- Utilisez les Websockets ou les Server-Sent Events pour la mise à jour en temps réel du classement.
- Utilisez les tests unitaires pour valider les fonctions critiques métiers.
- Utilisez les tests d'intégration pour valider les interactions entre les composants.

Annexes

Classement Elo

Le classement Elo est un système de classement des joueurs dans un jeu à deux joueurs. Il est basé sur le calcul de la probabilité de victoire de chaque joueur en fonction de leur classement respectif. Le classement est mis à jour après chaque match en fonction du résultat du match et de l'écart de classement entre les joueurs.

Ce système est typiquement utilisé dans les jeux d'échecs, mais peut être adapté à d'autres jeux à deux joueurs.

Calcul du classement

Le classement Elo est calculé en fonction du classement actuel des joueurs et du résultat du match. Le calcul est basé sur la formule suivante :

```
Rn = Ro + K * (W - We)
```

Avec :

- `Rn` : Nouveau classement du joueur
- `Ro` : Ancien classement du joueur
- `K` : Coefficient de pondération (facteur de sensibilité du classement)
- `W` : Résultat du match (1 pour une victoire, 0.5 pour une égalité, 0 pour une défaite)
- `We` : Probabilité de victoire du joueur en fonction de son classement et du classement de son adversaire

Exemple de calcul du classement de deux joueurs après un match :

```
# Joueur vainqueur
# Ro = 1200 # Ancien classement
# K = 32 # Coefficient de pondération
# W = 1 # Victoire
# We = 0.76 # Probabilité de victoire

Rn = 1200 + 32 * (1 - 0.76)
Rn = 1200 + 32 * 0.24
Rn = 1200 + 7.68
Rn = 1207.68
Rn ≈ 1208

# Joueur perdant
# Ro = 800 # Ancien classement
# K = 32 # Coefficient de pondération
# W = 0 # Défaite
# We = 0.24 # Probabilité de victoire

Rn = 800 + 32 * (0 - 0.24)
Rn = 800 + 32 * -0.24
Rn = 800 - 7.68
Rn = 792.32
Rn ≈ 792
```

Remarque : Le classement est arrondi à l'entier le plus proche. Le coefficient de pondération `K` est arbitraire et peut être ajusté pour augmenter ou diminuer la sensibilité du classement aux résultats des matchs.

Probabilité de victoire

La probabilité de victoire `We` est calculée en fonction de la différence de classement entre les deux joueurs. Plus la différence de classement est grande, plus la probabilité de victoire du joueur le mieux classé est élevée.

Le calcul de la probabilité de victoire est basé sur la formule suivante :

```
We = 1 / (1 + 10^((Rh - Rl) / 400))
```

Avec :

- `Rh` : Classement du joueur le mieux classé
- `Rl` : Classement du joueur le moins bien classé

Exemple de calcul de la probabilité de victoire d'un joueur avec un classement de 1200 contre un joueur avec un classement de 800 :

```
# Joueur le mieux classé
# Ro = 1200 # Classement du joueur
# Rn = 800 # Classement de l'adversaire

WHe = 1 / (1 + 10^((1200 - 800) / 400))
WHe = 1 / (1 + 10^(400 / 400))
WHe = 1 / (1 + 10^1)
WHe = 1 / (1 + 10)
WHe = 1 / 11
WHe ≈ 0.09

# Joueur le moins bien classé
# Ro = 800 # Classement du joueur
# Rn = 1200 # Classement de l'adversaire

WLe = 1 / (1 + 10^((800 - 1200) / 400))
WLe = 1 / (1 + 10^(-400 / 400))
WLe = 1 / (1 + 10^-1)
WLe = 1 / (1 + 0.1)
WLe = 1 / 1.1
WLe ≈ 0.91

# Ou plus simplement
WLe = 1 - WHe
```

Remarques :

- La probabilité de victoire est arrondie à deux décimales. Plus la différence de classement est grande, plus la probabilité de victoire est proche de 0 ou 1.
- La formule de calcul de la probabilité de victoire est basée sur la formule de la fonction logistique, qui est couramment utilisée pour modéliser des phénomènes binaires (victoire/défaite, succès/échec, etc.).
- La constante `400` est un paramètre empirique qui détermine la sensibilité de la probabilité de victoire à la différence de classement. Plus la constante est grande, plus la probabilité de victoire change rapidement avec la différence de classement.
- Par sa conception, le système Elo est mathématiquement un système transactionnel, c'est-à-dire que la somme des classements des joueurs reste constante après chaque match. Cela signifie que le classement gagné par un joueur est équivalent au classement perdu par son adversaire.