Enchères Pokemon !

L'objectif de ce projet est de réaliser une application Quarkus + React pour un site de jeu d'enchères de Pokemon.

Aucun argent réel ne sera utilisé bien entendu, nous utiliserons pour ce projet une monnaie créée pour l'occasion : des **Limcoins**.

Cette application sera dotée, d'une interface Web réalisée en React permettant à l'utilisateur de s'identifier en fonction de son type. Elle permettra également d'effectuer toutes les interactions entre l'utilisateur et le reste de l'application. Elle devra également assurer la persistance des données à l'aide de JPA et d'une base de données H2 externe.

Vous devrez également utiliser une API REST externe pour obtenir les informations sur les Pokemons : `https://pokeapi.co/` ou `https://tyradex.vercel.app/`

Les principales classes à écrire et les règles

1.1 Structure générale

Votre application devra comporter a minima 4 classes :

- La classe `Administrateur` qui est chargé de la création, modification, suppression des `Utilisateurs`
- La classe `Utilisateur` qui se caractérisera au minimum par son nom, et l'argent dont il dispose. Ce n'est pas limitatif, vous pouvez lui donner plus de propriétés si nécessaire.
- La classe `Pokemon` qui désignera les objets à vendre dans notre application
- La classe `Enchere`

Vous devez respecter ces noms de classes !

1.2 Règles du jeu

Par défaut tout le monde démarre avec 1000 **Limcoins** en poche. Un `Utilisateur` peut poser des enchères sur tous les `Pokemon` qui l'intéressent dans la limite des fonds dont il dispose. Il lui est par contre possible de dépasser son plafond en posant plusieurs enchères en parallèle (chacune étant fixée par la limite). Si jamais il remporte plus d'enchères que sont plafond total ne le lui permet, il devra choisir celles qu'il abandonne.

Un `Pokemon` se caractérisera par un nom, une description, une mise à prix, une valeur réelle (qui n'est pas communiquée). Pour des questions de traçabilité, les `Pokemon` gardent un historique de toutes les enchères qui ont été faites (et pas uniquement l'enchère la plus haute). La mise à prix sera tirée aléatoirement à partir de la valeur réelle (+/- 40% autour de celle-ci). Lors de l'affichage des enchères, les caractéristiques du `Pokemon` seront toutes visibles sauf sa valeur réelle. La valeur réelle est calculée

suivant la méthode de votre choix, en tenant compte des statistiques du `Pokemon` (plus ses statistiques sont élevées, plus sa valeur est élevée).

Au début, personne ne possède de `Pokemon` : c'est le serveur Quarkus qui va générer des enchères, et donc créer des `Pokemon` pour commencer à jouer. Il s'assurera qu'il y ait toujours un certain nombre de `Pokemon` à vendre (au moins 5).

Un `Pokemon` est remporté à l'expiration de l'enchère par `Utilisateur` ayant placé la plus haute mise. Cet `Utilisateur` a alors un temps limite pour régler le `Pokemon`. Si ce temps expire ou si l'Utilisateur gagnant ne dispose pas des fonds, c'est le second meilleur enchérisseur qui remporte le `Pokemon` et ainsi de suite.

Une fois le `Pokemon` remporté, l'`Utilisateur` peut soit le conserver (et le mettre dans une liste de `Pokemon`), soit le vendre au système pour sa valeur réelle. Si jamais, il juge la valeur trop basse, il a moyen de l'entraîner avant de le revendre. Pour cela, il dispose d'une méthode d'entraînement qui prend un `Pokemon` à la fois. Ce `Pokemon` voit sa valeur augmenter de 1% par tranche de 5min (ce qui se traduit par une augmentation linéaire de ses stats de 1% également). Cet entraînement faisant intervenir des entraîneurs sur-qualifiés, il est bien entendu payant. Chaque tranche de 1% va coûter 250 limcoins et un `Pokemon` ne pourra jamais gagner plus de 10% dans une séance d'entraînement. Il peut également remettre son `Pokemon` aux enchères en fixant un prix de départ de son choix.

Pour la démo, vous pouvez raccourcir le temps mis pour les entraînements.

En plus :

- Les informations sur les `Pokemon` seront obtenues à partir d'une API REST en ligne (privilégiez `https://pokeapi.co/` si vous le pouvez). Pour obtenir un `Pokemon` aléatoirement, faites un tirage aléatoire entre 1 et 718.
- Votre serveur Quarkus devra faire en sorte de limiter les appels au service REST en gardant un cache local des `Pokemon` déjà obtenus.
- Au niveau de l'interface du frontend React, l'`Utilisateur` devra pouvoir chercher les `Pokemon` aux enchères en fonction de leur type (eau, feu, terre, électricité...), de leur valeur...
- La compétition étant partout, vous devrez afficher dans votre frontend la liste des 5 `Pokemon` les plus chers de l'histoire du jeu ainsi qu'un classement des meilleurs `Utilisateur` (en se basant sur la quantité d'argent dont ils disposent)

Consignes supplémentaires et fonctionnalités avancées

La section précédente présente les fonctionnalités minimum attendues (pour viser 10/20), vous pouvez bien sur aller beaucoup plus loin. Chaque amélioration sera prise en compte pour l'évaluation de votre projet.

Voici quelques pistes possibles :

- Services REST : Un `Utilisateur` pourrait réaliser toutes les opérations d'enchères à travers des services Web REST plutôt que de passer par React. Ces services utiliseront le format JSON pour les échanges

— Définition d'un document YAML compatible OpenAPI pour décrire les services REST précédents
— un classement dynamique des `Pokemon` sur des périodes de temps définies par l'`Utilisateur`
— Authentification et sécurité : Ajout d'un mode d'authentification
— Utilisation de système de communication (mail, twitter etc) pour signaler à un `Utilisateur` qu'il a remporté une `Enchere` ou que quelqu'un a fait une meilleure offre sur un `Pokemon` sur lequel il a déjà enchéri. De la même manière, vous pouvez prévoir une liste de `Pokemon` favoris qui signale à l'`Utilisaeur` qu'un `Pokemon` qu'il attend vient d'arriver aux enchères
— Ajout d'un moteur de combat entre pokemon à travers un système d'invitation entre `Utilisateur` .

Mode d'évaluation

Le projet se fera en binôme !

Pour l'évaluation vous devrez rédiger un rapport présentant vos choix de conceptions ainsi que les extraits de code les plus pertinent dans un rapport au format **PDF** (**et uniquement PDF**).

Vous devrez également remettre un **zip** contenant toutes les sources de votre projet. Ces remises se feront exclusivement à travers la plateforme **Communities** de l'Université. **Aucune remise par mail ne sera acceptée.**

Une séance de présentation de votre projet devrait également être mise en place. La date exacte sera communiquée sur le forum.

Concernant l'évaluation, la qualité de conception de votre application sera primordiale. Essayez de privilégier une approche par micro-services.

En particulier :

— La manière de répartir les différents composants et la façon dont ceux-ci vont communiquer entre eux sera déterminante dans la notation
— La qualité et la clarté du code sera également prise en compte tout comme sa modularité et son extensibilité.
— La gestion des erreurs est également un plus.
— Le fait de répartir votre application entre plusieurs serveurs d'applications sera fortement valorisée.
— L'ergonomie générale de l'application sera bien entendu importante pour l'évaluation mais elle ne sera pas valorisée au même niveau que les nouvelles fonctionnalités que vous pourrez apporter ou encore que l'architecture générale de votre application.