ISIMA - Clermont Auvergne INP

Technologies du Web – Serveur
Z225DM06
Rapport

Pokémon

Élèves :
Valentin Portail
Jérémie Villepreux

Enseignant :
Yannick Loiseau
Youssef Bouaziz

21 mai 2023

Table des matières

1 Introduction

Pour rappel, il nous est demandé de réaliser un jeu de collection de Pokémons. Le but est simple : lorsqu'un utilisateur s'inscrira sur notre site, il recevra un Pokémon au hasard. Puis à chaque nouvelle connexion de la journée, il recevra un autre Pokémon au hasard (Pokémon qu'il a peut-être déjà). La possibilité d'échanger un Pokémon avec d'autres personnes (un pour un) est aussi présente. De plus, chaque jour, un utilisateur pourra faire monter le niveau de 5 Pokémons de son choix (qu'ils soient à lui ou non). Chaque Pokémon pourra aller du niveau 1 au niveau 100. Le Pokémon reçu lors de sa première connexion quotidienne est toujours de niveau 1. Le concept d'évolution n'est pas abordé dans ce projet.

2 Réflexion

Cette partie est consacrée à nos réponses aux questions du TP1.

2.1 Les fonctionnalités importantes

Quelles sont les fonctionnalités qui vous semblent importantes ?

Le but de ce projet est de créer un dispositif de collecte et d'échange de Pokémons. Il faut aussi prévoir un moyen pour améliorer nos Pokémons. Pour cela, nous avons décidé de mettre en place les fonctionnalités suivantes :

- s'inscrire sur le site ;
- se connecter sur le site (et étiqueter la date de la connexion) ;
- distribuer un Pokémon aléatoire à un utilisateur à chacune de ses connexions quotidiennes (tirer aléatoirement un Pokémon puis lui affecter) ;
- gérer les échanges de Pokémons ;
- augmenter le niveau de 5 Pokémons par jour ;
- accéder à la liste des Pokémon d'un utilisateur ;

Nous avons aussi décidé d'ajouter certaines autres fonctionnalités (non demandées explicitement), que celles mentionnées en classe :

- se désinscrire du site ;
- voir la liste de tous les utilisateurs du site : cela est plus pratique pour un utilisateur pour trouver un autre utilisateur avec qui échanger ses Pokémons ;
- voir la liste de tous les Pokémons d'un utilisateur spécifique ;
- voir l'ensemble des Pokémons présents sur le site : cela est plus pratique pour trouver un Pokémon à échanger avec quelqu'un. On peut chercher le Pokémon qui nous intéresse puis proposer un échange à son propriétaire ;
- supprimer un Pokémon pour un utilisateur. Par exemple s'il a trop de fois le même Pokémon et s'il veut s'en débarrasser car il n'arrive pas à l'échanger ;
- renommer son Pokémon ;
- annuler la demande d'échange (fait par l'expéditeur) ;
- avoir la liste de tous les échanges dont l'utilisateur est expéditeur ou destinataire (séparément) ;
- voir les sprites d'un Pokémon ;

2.2 Les principales classes

Nous avons implémenté quatre classes principales dans notre programme. Le diagramme de classe est donné en Annexe A - Figure 1. Nous avons fait comme choix :

UserEntity : classe dont une instance représentera un utilisateur. Un utilisateur a un `id` (référence dans la base de données), `firstName` (prénom), `lastName` (nom de famille), `identifier` (identifiant de connexion), `password` (mot de passe de connexion), `lastConnection` (date de dernière connexion), `nbUpgrades` (nombre d'améliorations restantes : entre 0 et 5) ;

PokemonEntity : classe dont une instance représentera un Pokémon dans la collection d'un joueur. Un Pokémon a un `id` (référence dans la base de données du site), `nickname` (surnom), `level` (un niveau entre 1 et 100), `species` (référence vers une instance de PokedexEntity `owner` (propriétaire du Pokémon). La plupart des autres informations sur le Pokémon (nom, type...) sont récupérées depuis la PokéAPI avec l'`species` ;

PokedexEntity : classe dont une instance représentera un Pokémon de la PokéAPI. Un Pokémon a un `id` (numéro du Pokémon dans l'API), un `name` (nom du Pokémon) et une `spriteURL` (URL vers le sprite du Pokémon : image utilisée pour représenter un Pokémon).

ExchangeEntity : classe dont une instance représentera un échange de Pokémons entre 2 utilisateurs. Un échange a un `id` (référence dans la base de données), `pokemonExp` (le Pokémon de l'expéditeur), `pokemonDest` (le Pokémon du destinataire) ;

2.3 Le schéma de notre base de données

Notre base de données est constituée de 3 tables :

users : contient l'ensemble de tous les utilisateurs. La clé primaire est `id` (en `auto increment`, c'est-à-dire qu'il sera ajouté automatiquement au moment de la création d'un nouvel utilisateur).

pokemons : contient tous les pokémons. La clé primaire est `id` (en `auto increment`). `owner` fait référence au propriétaire du Pokémon référencé par `id` de de la table `users`.

exchanges : contient tous les échanges de Pokémons. La clé primaire est `id_exch`. Les clés étrangères `id_pokemon_exp` et `id_pokemon_dest` font référence à `id` de la table `pokemons`.

Le schéma est disponible en Annexe B - Figure 2.

A noter que les tables ont été modifiées par rapport à celles données en classe. En effet, nous n'avons plus la table `users_pokemons`. Cette table servait de table *d'association* entre les tables `users` et `pokemons`. En regardant de plus près la cardinalité, nous nous sommes rendus compte qu'un utilisateur peut posséder autant de Pokémons que possible, mais qu'un Pokémon est forcément associé à un seul utilisateur. Nous avons donc supprimé la table `users_pokemons` et rajouté un attribut `owner` dans la table `pokemons` qui fait référence à la propriété `id` de `users`.

2.4 Les solutions technologiques envisagées

Les solutions technologiques envisagées sont celles évoquées en classe : une base de données H2 compatible avec SQL, le framework JavaSpark, le moteur de templates FreeMarker, la PokéAPI et les langages HTML et CSS pour la mise en page du site, mais aussi JWT (JsonWebToken) pour générer un token JWT, et jbcrypt pour chiffrer, déchiffrer et hacher les mots de passe.

3 Notre choix d'URI (Uniform Resource Identifiers)

Les mots clés PUBLIC et PRIVÉ entre parenthèses après les URI permettent de connaître les permissions d'accès à chaque page : les URI PUBLIC sont accessibles à tous les utilisateurs sans connexion nécessaire, tandis que les URI PRIVÉ ne sont accessibles que si l'utilisateur est connecté et concernent en général ses propres données (Pokémons, échanges...).

3.1 Accueil

GET / : Menu principal ;

3.2 Connexion et déconnexion

GET /**login** : Formulaire de connexion ;

POST /**login** : Envoi du formulaire de connexion ;

GET /**logout** : Déconnexion de l'utilisateur ;

3.3 Inscription et désinscription

GET /**signin** : Formulaire d'inscription ;

POST /**signin** : Envoi du formulaire d'inscription et identification de l'utilisateur ;

GET /**signout** : Désinscription de l'utilisateur actuellement connecté (en le supprimant définitivement de la base de données) ;

3.4 Utilisateurs et Pokémons

GET /**users** (PUBLIC) : Liste de tous les utilisateurs ;

GET /**users**/ **:id-user** (PUBLIC) : Liste des Pokémons de l'utilisateur `id-user`

GET /**pokemons** (PUBLIC) : Liste de tous les Pokémons présents sur le site ;

GET /**pokemons**/ **:id-pkmn** (PUBLIC) : Informations détaillées sur un Pokémon présent sur le site ;

3.5 Actions sur les Pokémons

POST /**pokemons**/ **:id-pkmn**/**levelup** (PRIVÉ) : Fait monter de niveau le Pokémon référencé par `id-pkmn` ;

POST /**pokemons**/ **:id-pkmn**/**rename** (PRIVÉ) : Modifie le surnom du Pokémon référencé par `id-pkmn` ;

POST /**pokemons**/ **:id-pkmn**/**delete** (PRIVÉ) : Supprime le Pokémon référencé par `id-pkmn` ;

3.6 Échanges

GET /**exchanges** (PUBLIC) : Liste de tous les échanges ;

GET /**exchanges**/ **:id-exchange** (PUBLIC) : Permet d'obtenir plus de précisions sur les conditions d'un échange potentiel ;

GET /**exchanges**/**exp**/ **:id-user** (PUBLIC) : Liste de tous les échanges dont l'utilisateur est expéditeur ;

GET /**exchanges**/**dest**/ **:id-user** (PUBLIC) : Liste de tous les échanges dont l'utilisateur est destinataire ;

POST /**exchanges**/**start** (PRIVÉ) : Démarre un échange ;

POST /**exchanges**/ **:id-exchange**/**cancel** (PRIVÉ) : Permet d'annuler la demande d'échange ;

POST /**exchanges**/ **:id-exchange**/**agree** (PRIVÉ) : Accepter l'échange (pour le destinataire) ;

POST /**exchanges**/ **:id-exchange**/**disagree** (PRIVÉ) : Refuser l'échange (pour le destinataire) ;

A Diagramme de classe des entités

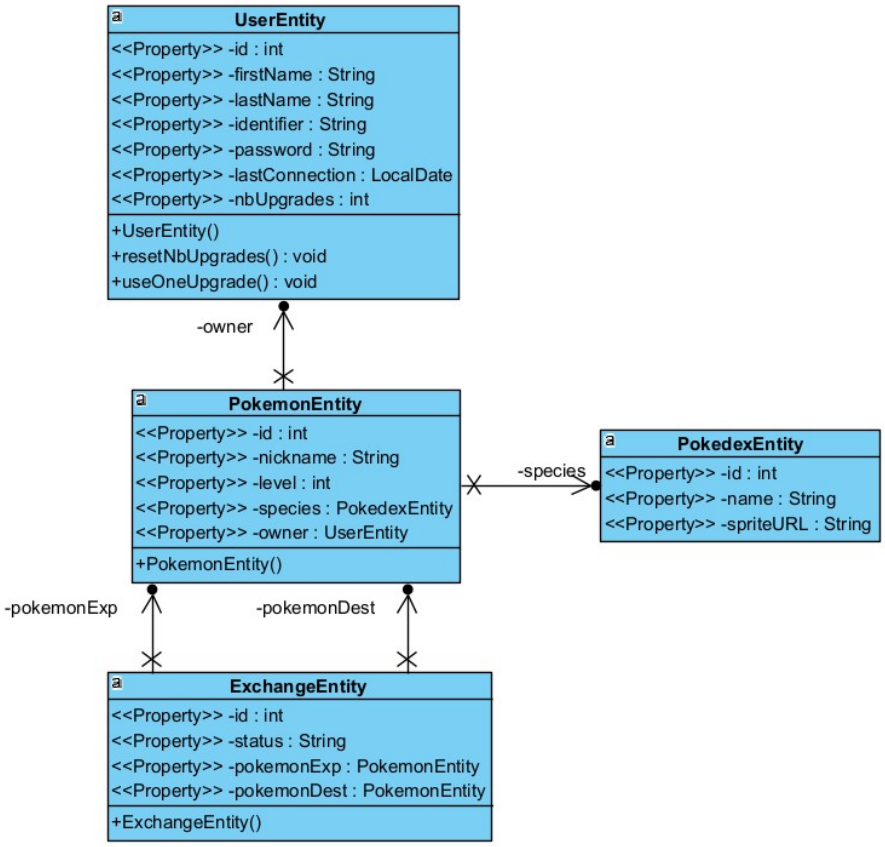

FIGURE 1 – Nos principales classes

B Schéma relationnel de notre base de données

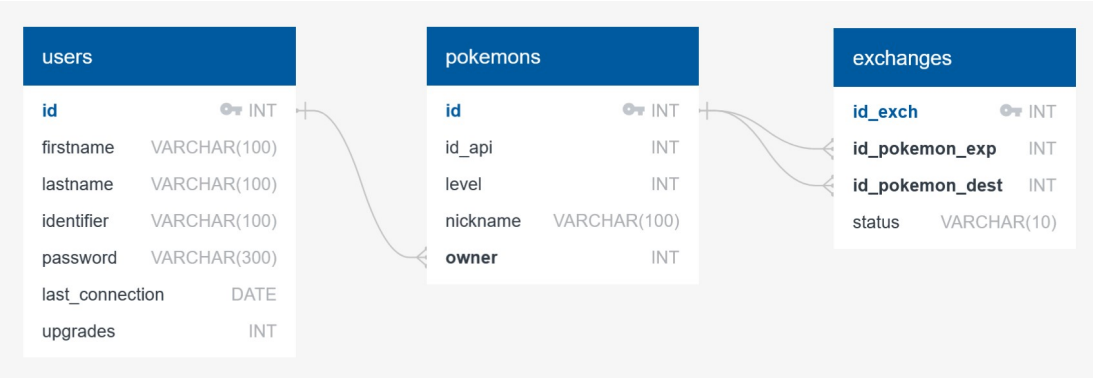

FIGURE 2 – Schéma relationnel de notre base de données