Rapport
Technologies du web-seveur

YMERAJ Eldis et HOBEIKA Sandrine

21/05/2023

Figure 1 – Jeu du Pokémons

Table des matières

Introduction **3**

1 Structure du projet **3**

1.1 Fonctionnalitées importantes 3

1.2 Classes principales de l'application 3

1.3 Les tables de la base de données 4

2 L'approche **4**

2.1 MVC 4

3 Routes **4**

3.1 GET "/" 4

3.2 GET "/users" 4

3.3 POST "/users" 5

3.4 GET "/users/userId/pokemons" 5

3.5 POST "/logout" 5

4 Connection **5**

4.1 Fonctionnement 5

4.2 Les étapes du connection 6

Introduction

Dans ce rapport on va décrire la conception et l'implémentation d'un projet web côté serveur visant à mettre en place un jeu de collection autour de l'univers des Pokémons. Ce projet permettra aux utilisateurs de s'inscrire, de recevoir des Pokémons aléatoires quotidiennement, de faire évoluer leurs Pokémons et d'échanger ces créatures avec d'autres joueurs.

La réalisation de ce projet respecte les principes fondamentaux enseignés lors de notre cours, notamment la conception d'un serveur sans état, l'utilisation d'URIs prédictibles et la gestion des verbes et codes HTTP conformément aux bonnes pratiques.

1 Structure du projet

1.1 Fonctionnalitées importantes

1. Inscription d'un utilisateur.
2. Connection d'un utilisateur.
3. 1 utilisateur a une liste de pokemons :
   — Ajouter
   — Supprimer
4. Système d'échange de pokemons avec des autres utilisateurs.
5. Recevoir 1 pokemon par jour.
6. Affichage utilisateur.
7. 1 pokemon a des niveaux,
   — Système de monté de niveau de pokemons.
8. Connecter avec API.

1.2 Classes principales de l'application

Les classes nécessaires comprennent naturellement une classe qui représente les utilisateurs, une pour les pokemons, et une pour les échanges.

UTILISATEUR :

— Nom/Prénom
— Username (VARCHAR) **UNIQUE**
— Mot de passe
— Liste des pokemons
— Dernière connection (DATE)
— ID

POKEMON

— Identifiant
— Niveau
— L'utilisateur ou il appartient
— Nom
— Spite (optionnel)

ECHANGE

— Utilisateur
— Pokemon

1.3 Les tables de la base de données

UTILISATEUR

- id int index(primary key) AUTOINCREMENT
- firstname VARCHAR(100)
- lastname VARCHAR(100)
- username VARCHAR(100) **UNIQUE**
- password VARCHAR(100)

POKEMON

- id int index(primary key) AUTOINCREMENT
- pokemonId int
- Niveau int
- userId int (foreign key reference user.id)
- Nom VARCHAR(255)
- Spite

ECHANGE

- id int index(primary key) AUTOINCREMENT
- userId int (foreign key reference user.id)
- pokemonId int (foreign key reference pokemon.id)

2 L'approche

2.1 MVC

En suivant la méthode générale du dossier de démarrage, on a adopté une approche Model-View-Controller. La classe StartServer lance l'application, puis écoute et attend de recevoir des évènements de la part de la vue. Quand ils sont déclenchés, si l'URI et le verbe HTTP sont reconnus, elle s'occupe d'orchestrer les requêtes et les réponses, et fait appel à un package gui qui prépare des templates. Il doit récupérer des données avant de pouvoir les afficher, donc il fait à son tour appel à un autre package : core. Celui-ci reçoit des appels de gui et appelle des méthodes du 2 package dao : il fait donc la liaison entre la vue et le modèle. C'est dans dao que les appels à la base de données ont lieu. Les classes d'entity sont utilisées dans l'ensemble du projet et définissent comment on représente nos objets en Java.

3 Routes

3.1 GET "/"

Cette route correspond à la page d'accueil de votre application. Elle affiche un message indiquant que l'utilisateur n'est pas connecté si aucune session utilisateur n'est active. Si un utilisateur est connecté, elle affiche un message de bienvenue avec le nom d'utilisateur. Elle inclut également le fichier de modèle "header.ftl" pour l'en-tête commun à toutes les pages.

3.2 GET "/users"

Cette route récupère tous les utilisateurs à partir de la base de données et les affiche dans un tableau. Chaque utilisateur est affiché avec son ID, nom d'utilisateur et mot de passe. Pour chaque utilisateur, il y a un lien "Voir les Pokémon" qui redirige vers la route "/users/userId/pokemons".

3.3 POST "/users"

Cette route est utilisée pour créer un nouvel utilisateur dans la base de données. Les données du formulaire (nom d'utilisateur et mot de passe) sont envoyées en tant que requête POST. Une fois l'utilisateur créé avec succès, il est redirigé vers la page d'accueil.

3.4 GET "/users/userId/pokemons"

Cette route récupère tous les Pokémon associés à un utilisateur spécifié par son ID. Les Pokémon sont récupérés à partir de la base de données en utilisant la méthode "getPokemonsUser(userId)".

Les Pokémon sont affichés dans une liste.

3.5 POST "/logout"

Cette route est utilisée pour déconnecter l'utilisateur actuellement connecté. Lorsqu'elle est appelée, la session utilisateur est détruite, ce qui conduit à la déconnexion de l'utilisateur. L'utilisateur est ensuite redirigé vers la page d'accueil avec le message indiquant qu'il n'est pas connecté.

4 Connection

4.1 Fonctionnement

Lorsque l'utilisateur se connecte avec succès, une requête est envoyée à l'API Pokémon pour récupérer un Pokémon aléatoire.

L'application utilise la méthode **getRandomPokemon()** pour envoyer une requête HTTP GET à l'API Pokémon. Cette requête est envoyée à l'URL appropriée pour obtenir un Pokémon aléatoire entr 1 et 1008.

L'API Pokémon renvoie les détails du Pokémon au format JSON en réponse à la requête.

Les données JSON de la réponse sont analysées et converties en un objet Pokémon à l'aide de la méthode **convertJsonToPokemon()**.

L'objet Pokémon obtenu représente le Pokémon aléatoire récupéré de l'API. Cet objet est ensuite associé à l'utilisateur qui s'est connecté, enregistré dans la base de données et attribué à cet utilisateur.

Ainsi, à chaque connexion, un Pokémon aléatoire est récupéré à partir de l'API Pokémon et attribué à l'utilisateur. Cette approche garantit que chaque utilisateur reçoit un Pokémon unique et aléatoire lors de sa connexion à l'application.

4.2 Les étapes du connection

Le principe de connexion dans l'application repose sur les étapes suivantes :

Sur la page de connexion, l'utilisateur saisit son nom d'utilisateur et son mot de passe dans les champs prévus à cet effet.

Lorsque l'utilisateur soumet le formulaire de connexion, une requête est envoyée au serveur de l'application.

Le serveur reçoit la requête de connexion et vérifie les informations fournies par l'utilisateur. Il recherche dans la base de données si un utilisateur correspondant aux informations de connexion existe.

Si les informations sont valides, l'utilisateur est considéré comme authentifié et une session est créée pour lui. Les informations de connexion, telles que l'identifiant de l'utilisateur, sont généralement stockées dans la session pour une utilisation ultérieure.

L'utilisateur est redirigé vers une page d'accueil ou une autre page appropriée pour les utilisateurs connectés.

Si les informations de connexion sont incorrectes ou si l'utilisateur n'existe pas, un message d'erreur est affiché, indiquant à l'utilisateur que la connexion a échoué.

Le principe de connexion garantit que seuls les utilisateurs authentifiés avec des informations de connexion valides peuvent accéder aux fonctionnalités de l'application.

Conclusion

En conclusion, ce projet de jeu de collection de Pokémons nous a permis d'appliquer les principes essentiels de la conception d'une application web côté serveur. En utilisant Java Spark, FreeMarker et la base de données H2, nous avons réussi à créer un serveur sans-état avec des URIs prédictibles et respectant les normes HTTP.

Nous avons implémenté les fonctionnalités demandées, telles que l'inscription des utilisateurs, la gestion des listes de Pokémons, l'attribution quotidienne aléatoire de Pokémons, le système d'échange entre utilisateurs, etc.

Ce projet nous a également permis de nous familiariser avec l'utilisation de l'API PokéAPI pour obtenir les données sur les Pokémons.

En résumé, ce projet nous a offert une expérience pratique précieuse dans le développement web côté serveur, renforçant nos connaissances et compétences. Nous sommes maintenant mieux préparés pour aborder des projets plus avancés et explorer des frameworks tels que Springboot dans le futur.

Table des figures

1 Jeu du Pokémons 1