TCG Card Collection Service

Ce projet est un service de gestion de collection de cartes pour un jeu de cartes à collectionner (Trading Card Game - TCG). Il permet aux utilisateurs de gérer leurs collections de cartes, d'effectuer des échanges de cartes, et d'interagir avec un service de profil utilisateur. Le système est conçu avec une architecture microservices utilisant les protocoles de communication REST et gRPC.

Description des fonctionnalités :

- Gestion de collection de cartes : permet de gérer les cartes et les boosters des joueurs.
- Échange de cartes : permet l'échange de cartes entre joueurs.
- Service de profil utilisateur : permet de gérer les informations des utilisateurs et de lier les collections aux utilisateurs.
- Persistance des données : les données sont stockées dans une base de données MySQL avec la capacité de persister les données entre les redémarrages.
- Communication entre microservices : utilise les protocoles REST et gRPC pour la communication entre services.
- Documentation OpenAPI (Swagger) : les endpoints de l'API REST sont documentés avec la spécification OpenAPI.

Structure du projet

- card-collection-service : Service de gestion des collections de cartes et des échanges.
- user-service : Service de gestion des profils utilisateurs et des collections liées.
- myServiceServer : Serveur gRPC pour la gestion de la communication inter-services.
- myServiceInterface : Mini service qui contient les méthode proto

Démarrage

1. Aller dans le dossier du projet :

cd TCG

2. Construire les services avec Gradle :

gradle :card-collection-service:bootRun

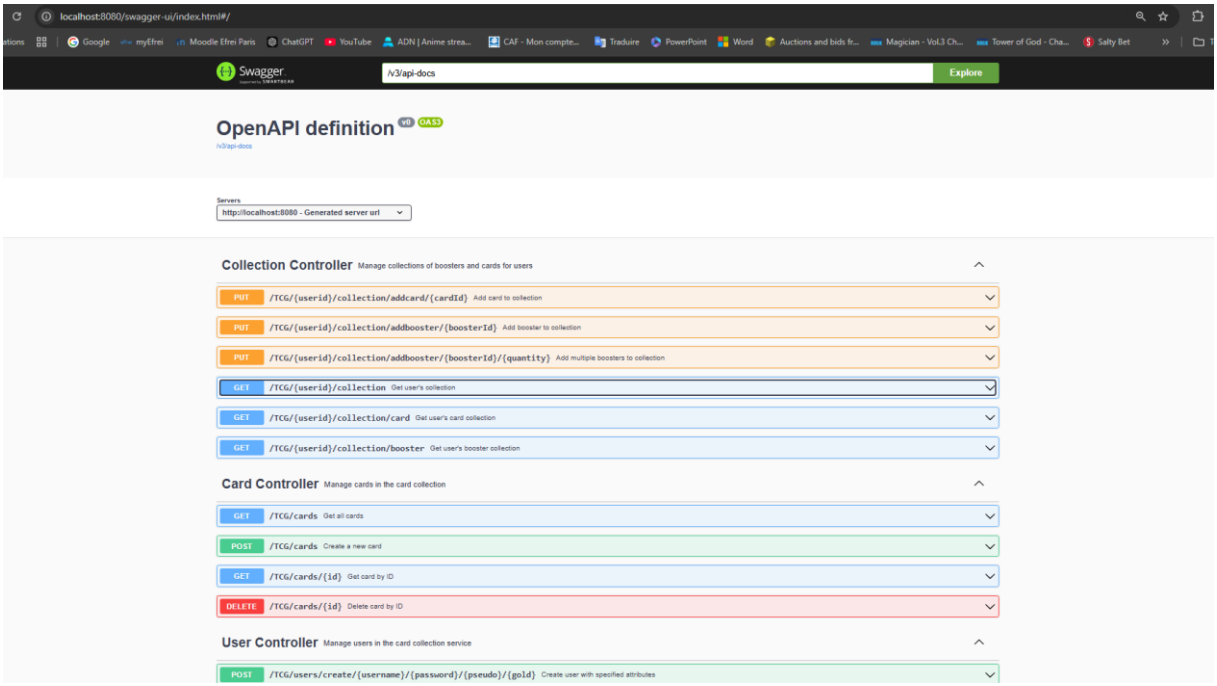
gradle :user-service:bootRun

Documentation de l'API

Les services REST sont documentés avec Swagger et suivent la spécification OpenAPI. Une fois les services démarrés, vous pouvez accéder à l'interface Swagger pour explorer l'API.

Accéder à Swagger UI :

http://localhost:8080/swagger-ui.html



Endpoints de l'API REST

Liste des principaux endpoints REST disponibles dans le projet. Vous pouvez tester ces endpoints avec les commandes curl. (les commandes n'ont pas était tester car nous passion par postman pour tester nos function)

User Service - TCG

Ce service gère les utilisateurs du système pour une application de Trading Card Game (TCG). Il permet de créer, récupérer, mettre à jour, et supprimer des utilisateurs, ainsi que d'authentifier un utilisateur par identifiant et mot de passe.

Structure de la base de données

Ce service utilise MongoDB pour stocker les informations des utilisateurs dans une collection appelée users. Chaque utilisateur a les attributs suivants :

- id : Identifiant unique de l'utilisateur (généré par MongoDB).
- username : Nom d'utilisateur.
- email : Adresse e-mail de l'utilisateur.
- password : Mot de passe de l'utilisateur (note : il est recommandé de chiffrer les mots de passe dans une application en production).

Endpoints de l'API REST

Voici la liste des endpoints disponibles sur ce service. Tous les endpoints écoutent sur le port 8081.

1. Récupérer tous les utilisateurs

- URL : /users
- Méthode HTTP : GET
- Description : Récupère la liste de tous les utilisateurs.

curl -X GET http://localhost:8081/users

2. Récupérer un utilisateur par ID

- URL : /users/{id}
- Méthode HTTP : GET
- Description : Récupère un utilisateur par son ID.

curl -X GET http://localhost:8081/users/1

3. Connexion utilisateur par identifiant et mot de passe

- URL : /users/login/{username}/{password}
- Méthode HTTP : GET
- Description : Authentifie un utilisateur en utilisant son nom d'utilisateur et son mot de passe.

curl -X GET http://localhost:8081/users/login/{username}/{password}

4. Créer un nouvel utilisateur

- URL : /users
- Méthode HTTP : POST
- Description : Crée un nouvel utilisateur dans le système.
- Exemple de corps de la requête :

```
{
  "username": "exampleUser",
  "email": "example@example.com",
  "password": "examplePassword"
}
```

curl -X POST http://localhost:8081/users -H "Content-Type: application/json" -d '{"username": "exampleUser", "email": "example@example.com", "password": "examplePassword"}'

5. Mettre à jour un utilisateur existant par ID

- URL : /users/{id}
- Méthode HTTP : PUT
- Description : Met à jour les informations d'un utilisateur existant par son ID.
- Exemple de corps de la requête :

json

Copier le code

```
{
  "username": "updatedUser",
  "email": "updated@example.com",
  "password": "newPassword"
}
```

curl -X PUT http://localhost:8081/users/{id} -H "Content-Type: application/json" -d '{"username": "updatedUser", "email": "updated@example.com", "password": "newPassword"}'

6. Supprimer un utilisateur par ID

- URL : /users/{id}
- Méthode HTTP : DELETE
- Description : Supprime un utilisateur par son ID.

curl -X DELETE http://localhost:8081/users/{id}

7. Supprimer tous les utilisateurs

- URL : /users
- Méthode HTTP : DELETE
- Description : Supprime tous les utilisateurs du système.

curl -X DELETE http://localhost:8081/users

Card Collection Service

TCG Card Collection Service - User Management

Ce service gère les utilisateurs du système de collection de cartes pour un jeu de cartes à collectionner (Trading Card Game - TCG). Il permet de créer des utilisateurs avec des paramètres spécifiques et de récupérer tous les utilisateurs.

Structure de la base de données

Ce service utilise une base de données relationnelle pour stocker les informations des utilisateurs dans une table users. Chaque utilisateur a les attributs suivants :

- id : Identifiant unique de l'utilisateur.
- username : Nom d'utilisateur.
- pseudo : Pseudo de l'utilisateur.
- gold : Montant d'or que possède l'utilisateur.
- userBoosterCollection : Collection de boosters de l'utilisateur.
- userCardCollection : Collection de cartes de l'utilisateur.

Endpoints de l'API REST

Liste des principaux endpoints disponibles dans le UserController. Tous les endpoints écoutent sur le port 8080.

1. Récupérer tous les utilisateurs

- URL : /TCG/users
- Méthode HTTP : GET
- Description : Récupère la liste de tous les utilisateurs.

Exemple de requête curl

curl -X GET http://localhost:8080/TCG/users

2. Créer un utilisateur avec des paramètres spécifiques

- URL : /TCG/users/create/{username}/{password}/{pseudo}/{gold}
- Méthode HTTP : POST
- Description : Crée un nouvel utilisateur avec un nom d'utilisateur, un mot de passe, un pseudo et un montant d'or.
- Paramètres :
    - username : Nom d'utilisateur de l'utilisateur (ex. john_doe).
    - password : Mot de passe de l'utilisateur (ex. password123).
    - pseudo : Pseudo de l'utilisateur (ex. Warrior123).
    - gold : Montant d'or que possède l'utilisateur (ex. 100).

Exemple de requête curl

curl -X POST "http://localhost:8080/TCG/users/create/john_doe/password123/Warrior123/100"

Ce POST créera un utilisateur avec :

- username : john_doe
- password : password123
- pseudo : Warrior123
- gold : 100

TCG Card Collection Service - Card Management

Ce service permet de gérer les cartes dans la collection d'un jeu de cartes à collectionner (Trading Card Game - TCG). Il expose des endpoints pour récupérer toutes les cartes, récupérer une carte par ID, créer une nouvelle carte, et supprimer une carte par ID.

Structure de la base de données

Ce service utilise une base de données relationnelle pour stocker les informations des cartes dans une table cards. Chaque carte a les attributs suivants :

- id : Identifiant unique de la carte.
- name : Nom de la carte.
- defensiveStat : Statistique défensive de la carte.
- offensiveStat : Statistique offensive de la carte.
- effect : Effet de la carte.
- cost : Coût de la carte.
- rarity : Rareté de la carte.
- type : Type de la carte.

Endpoints de l'API REST

Liste des principaux endpoints disponibles dans le CardController. Tous les endpoints sont accessibles sous le chemin /TCG/cards sur le port 8080.

1. Récupérer toutes les cartes

- URL : /TCG/cards
- Méthode HTTP : GET
- Description : Récupère la liste de toutes les cartes disponibles.

Exemple de requête curl

curl -X GET http://localhost:8080/TCG/cards

2. Récupérer une carte par ID

- URL : /TCG/cards/{id}
- Méthode HTTP : GET
- Description : Récupère une carte par son ID.

Exemple de requête curl

curl -X GET http://localhost:8080/TCG/cards/{id}

3. Créer une nouvelle carte

- URL : /TCG/cards
- Méthode HTTP : POST
- Description : Crée une nouvelle carte avec les attributs spécifiés.
- Corps de la requête :

json

```
{
  "name": "Fire Dragon",
  "defensiveStat": 10,
  "offensiveStat": 15,
  "effect": "Burns enemy",
  "cost": 5,
  "rarity": "Rare",
  "type": "Dragon"
}
```

Exemple de requête curl

```
curl -X POST http://localhost:8080/TCG/cards -H "Content-Type: application/json" -d '{"name": "Fire Dragon", "defensiveStat": 10, "offensiveStat": 15, "effect": "Burns enemy", "cost": 5, "rarity": "Rare", "type": "Dragon"}'
```

4. Supprimer une carte par ID

- URL : /TCG/cards/{id}
- Méthode HTTP : DELETE
- Description : Supprime une carte par son ID.

Exemple de requête curl

```
curl -X DELETE http://localhost:8080/TCG/cards/{id}
```

TCG Card Collection Service - Booster Management

Ce service permet de gérer les boosters dans la collection d'un jeu de cartes à collectionner (Trading Card Game - TCG). Il expose des endpoints pour récupérer tous les boosters, récupérer un booster par ID, créer un nouveau booster et supprimer un booster par ID.

Structure de la base de données

Ce service utilise une base de données relationnelle pour stocker les informations des boosters dans une table boosters. Chaque booster a les attributs suivants :

- id : Identifiant unique du booster.
- name : Nom du booster.
- rarity : Rareté du booster.
- cost : Coût du booster.
- dropRate : Taux de drop du booster.

Endpoints de l'API REST

Liste des principaux endpoints disponibles dans le BoosterController. Tous les endpoints sont accessibles sous le chemin /TCG/boosters sur le port 8080.

1. Récupérer tous les boosters

- URL : /TCG/boosters
- Méthode HTTP : GET
- Description : Récupère la liste de tous les boosters disponibles.

Exemple de requête curl

curl -X GET http://localhost:8080/TCG/boosters

2. Récupérer un booster par ID

- URL : /TCG/boosters/{id}
- Méthode HTTP : GET
- Description : Récupère un booster par son ID.

Exemple de requête curl

curl -X GET http://localhost:8080/TCG/boosters/{id}

3. Créer un nouveau booster

- URL : /TCG/boosters
- Méthode HTTP : POST
- Description : Crée un nouveau booster avec les attributs spécifiés.
- Corps de la requête :

json

```
{
  "name": "Ultimate Booster",
  "rarity": "Legendary",
  "cost": 100,
  "dropRate": "5%"
}
```

Exemple de requête curl

```
curl -X POST http://localhost:8080/TCG/boosters -H "Content-Type: application/json" -d '{"name": "Ultimate Booster", "rarity": "Legendary", "cost": 100, "dropRate": "5%"}'
```

4. Supprimer un booster par ID

- URL : /TCG/boosters/{id}
- Méthode HTTP : DELETE
- Description : Supprime un booster par son ID.

Exemple de requête curl

```
curl -X DELETE http://localhost:8080/TCG/boosters/{id}
```

TCG Card Collection Service - Collection Management

Ce service gère les collections de boosters et de cartes pour les utilisateurs d'un jeu de cartes à collectionner (Trading Card Game - TCG). Il permet de récupérer la collection d'un utilisateur et d'ajouter des boosters et des cartes à la collection de l'utilisateur.

Structure de la base de données

Le service utilise une base de données relationnelle pour stocker les informations des collections d'utilisateurs dans les tables UserBoosterCollection et UserCardCollection.

- UserBoosterCollection : Contient la collection de boosters d'un utilisateur.
- UserCardCollection : Contient la collection de cartes d'un utilisateur.

Endpoints de l'API REST

Voici la liste des principaux endpoints disponibles dans le CollectionController. Tous les endpoints sont accessibles sous le chemin /TCG/{userid}/collection sur le port 8080.

1. Récupérer la collection d'un utilisateur

- URL : /TCG/{userid}/collection
- Méthode HTTP : GET
- Description : Récupère un résumé de la collection de l'utilisateur.

Exemple de requête curl

curl -X GET http://localhost:8080/TCG/{userid}/collection

2. Récupérer la collection de boosters d'un utilisateur

- URL : /TCG/{userid}/collection/booster
- Méthode HTTP : GET
- Description : Récupère la collection de boosters de l'utilisateur.

Exemple de requête curl

curl -X GET http://localhost:8080/TCG/{userid}/collection/booster

3. Récupérer la collection de cartes d'un utilisateur

- URL : /TCG/{userid}/collection/card
- Méthode HTTP : GET
- Description : Récupère la collection de cartes de l'utilisateur.

Exemple de requête curl

curl -X GET http://localhost:8080/TCG/{userid}/collection/card

4. Ajouter un booster à la collection d'un utilisateur

- URL : /TCG/{userid}/collection/addbooster/{boosterId}
- Méthode HTTP : PUT
- Description : Ajoute un booster spécifique à la collection de l'utilisateur.
- Paramètres :
    - boosterId : Identifiant du booster à ajouter.

Exemple de requête curl

curl -X PUT http://localhost:8080/TCG/{userid}/collection/addbooster/{boosterId}

5. Ajouter plusieurs boosters à la collection d'un utilisateur

- URL : /TCG/{userid}/collection/addbooster/{boosterId}/{quantity}
- Méthode HTTP : PUT
- Description : Ajoute un nombre spécifique de boosters à la collection de l'utilisateur.
- Paramètres :
    - boosterId : Identifiant du booster à ajouter.
    - quantity : Nombre de boosters à ajouter.

Exemple de requête curl

curl -X PUT http://localhost:8080/TCG/{userid}/collection/addbooster/{boosterId}/{quantity}

6. Ajouter une carte à la collection d'un utilisateur

- URL : /TCG/{userid}/collection/addcard/{cardId}
- Méthode HTTP : PUT
- Description : Ajoute une carte spécifique à la collection de l'utilisateur.
- Paramètres :
    - cardId : Identifiant de la carte à ajouter.

Exemple de requête curl

curl -X PUT http://localhost:8080/TCG/{userid}/collection/addcard/{cardId}

Endpoint REST

Ce service expose un endpoint REST qui permet d'interagir avec le service gRPC en utilisant une requête HTTP.

1. Échanger une carte

- URL : /exchange-card
- Méthode HTTP : GET
- Description : Effectue un échange de carte entre les joueurs en appelant le service gRPC. (La logique des échange n'est pas implémenter pour l'instant)
- Paramètres :
    - idDeJoueur : Identifiant du joueur (requis).
    - idDeCarte : Identifiant de la carte à échanger (requis).

Exemple de requête curl

curl -X GET "http://localhost:8080/exchange-card?idDeJoueur=1&idDeCarte=1

Utilisation

Après avoir démarré les services, vous pouvez utiliser les commandes curl ou Swagger UI pour interagir avec les API REST.

Pour la communication gRPC, assurez-vous que card-collection-service et user-service sont configurés pour utiliser les ports adéquats pour la communication gRPC comme défini dans les fichiers application.properties.

Stack Technique

- Java 17
- Spring Boot pour les services REST
- gRPC pour la communication inter-services
- MySQL pour la persistance des données
- Docker pour la containerisation
- Swagger/OpenAPI pour la documentation de l'API