

Schema de l'architecture

ChampionSphere

29.11.2023

Yahyaoui Firas

Dallel Mohamed

[<https://github.com/moDallel/ChampionSphere>]

Abstract

Le projet consiste en le développement d'une plateforme de jeu permettant à des joueurs, d'organiser et de participer à des combats entre des équipes de créatures. Chaque joueur peut constituer son équipe en achetant des créatures disponibles dans un magasin virtuel, et affronter d'autres joueurs pour remporter des crédits et des badges. La plateforme offrira des fonctionnalités de gestion de compte et d'interaction entre joueurs.

Domaines métier

Quel critère pour le découpage ?

On a découpé selon les fonctionnalités offertes par chaque service i.e chaque service est responsable seulement d'un aspect spécifique du domaine métier pour assurer une cohérence et une responsabilité claire.

Est-ce qu'il a une taille idéale ?

Il n'y a pas de taille idéale prédéfinie pour un service, mais on a essayé de les garder aussi petits que possible pour favoriser la modularité et la facilité de maintenance.

Domain Match

1. **Services:** Service Match
2. **Ressources gérées:** Matches, manches , joueurs
3. **Technologies:**
 - Langage de programmation: TypeScript
 - stockage de données: base de données NoSQL
4. **API exposée:** Créer un match, rejoindre un match, consulter les détails d'un match, envoyer une créature à l'arène, consulter le statut d'une manche

Domain Creature

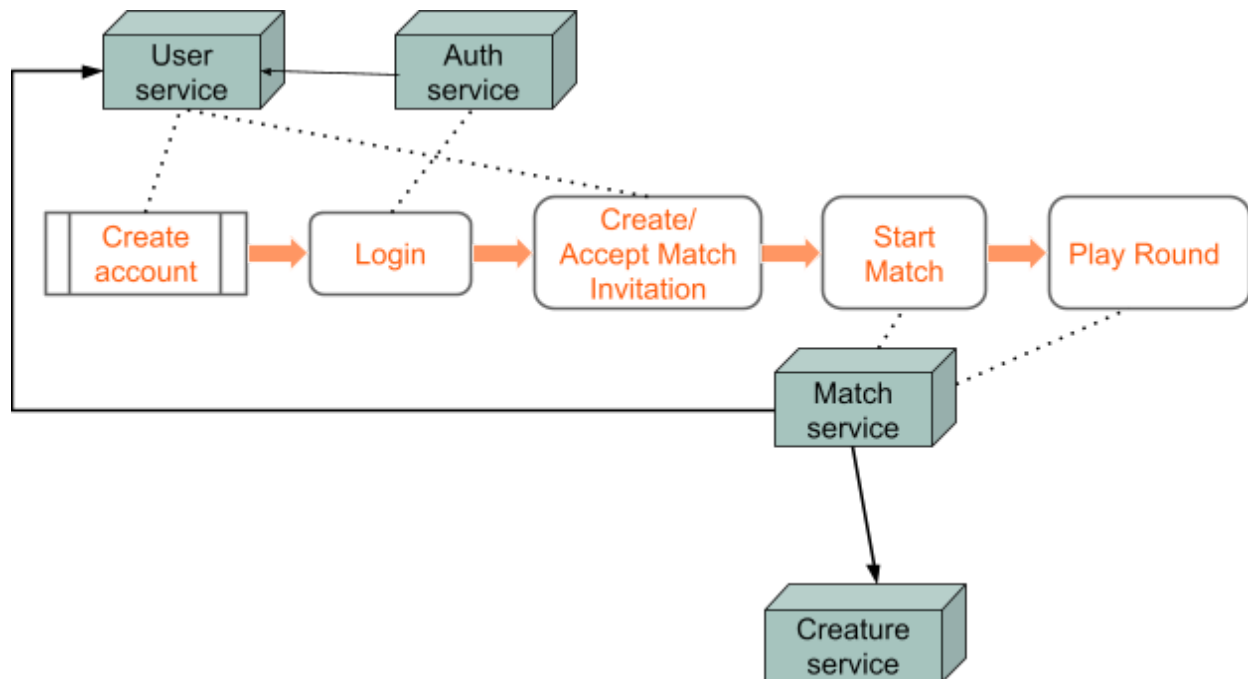
5. **Services:** Service Creature
6. **Ressources gérées:** Créatures, Store, crédits
7. **Technologies:**
 - Langage de programmation: TypeScript
 - stockage de données: base de données NoSQL

8. **API exposée:** Consulter la liste de créatures disponibles


Domain User

9. **Services:** Service User, Service Auth
10. **Ressources gérées:** Compte utilisateur, créatures, interactions avec d'autres joueurs, badges
11. **Technologies:**
Langage de programmation: TypeScript
stockage de données: base de données NoSQL
12. **API exposée:** Créer un compte, se connecter, gérer les créatures, interagir avec les autres joueurs (participer à un match...), consulter les badges

Path critique



Le chemin critique consiste pour un utilisateur à s'inscrire à l'aide de l'API exposée par le service utilisateur, à se connecter avec son nom d'utilisateur et son mot de passe avec le service d'authentification, puis à créer ou à accepter une invitation à un match. Une fois que le match est prêt, son statut est modifié pour commencer et les 5 manches sont jouées



en utilisant le service de match, où chaque joueur invoque une créature (avec l'aide du service de créature). Une fois qu'un gagnant est déterminé, le match est terminé.

Etape I (create-account)

Il s'agit d'une requête POST au service User pour créer un nouvel utilisateur avec les informations nécessaires.

Etape II (login)

Il s'agit d'une requête POST au service Auth avec le nom d'utilisateur et le mot de passe. Si l'utilisateur existe et le mot de passe correspond à celui de l'utilisateur, un token est retourné qui sera utilisé dans toutes les requêtes ultérieures.

Etape III (create-match-invitation)

A l'aide du service Match, l'utilisateur va créer une invitation pour jouer un match avec quelqu'un qu'il connaît. Sinon, il peut créer un match ouvert auquel n'importe quel utilisateur peut participer.

Etape IV (start-match)

Une fois l'invitation du match acceptée, un nouveau match est créé avec le service Match et les deux joueurs peuvent commencer.

Etape V (play-round)

A chaque manche, les deux joueurs lancent leurs créatures et un algorithme de jeu détermine celui qui gagne. 5 manches sont jouées et à la fin un des deux joueurs est déclaré comme gagnant.

Obstacles remarquables

I. La communauté Fastify est relativement petite

Il y a moins de ressources disponibles que Express pour aider à résoudre les bugs. Cela peut poser un problème pour un projet important.

II. La gestion des tokens avec JWT

est complexe et peut être source d'erreurs. Les tokens JWT sont utilisés pour authentifier les utilisateurs et protéger les données sensibles. Il est important de les gérer correctement pour éviter les failles de sécurité.

III. La conteneurisation avec Docker de MongoDB

peut être complexe et nécessite une bonne compréhension des principes de la conteneurisation surtout avec la gestion de l'unicité des valeurs.

Plateforme technique

Pour réaliser le projet, nous utiliserons les technologies suivantes :

TypeScript : un langage de programmation orienté objet qui est une surcouche de JavaScript. TypeScript offre de nombreux avantages, notamment une meilleure sécurité, une meilleure productivité et une meilleure évolutivité.

Fastify : un framework Node.js pour la création d'API REST. Fastify est un framework performant et scalable qui est idéal pour les applications à grande échelle.

MongoDB : une base de données NoSQL qui est idéale pour le stockage de données de grande taille. MongoDB est un choix naturel pour un jeu en ligne qui doit pouvoir gérer un grand nombre de joueurs et de matchs.

Docker : un outil de conteneurisation qui permet de créer des images de logiciels portables et reproductibles. Docker simplifie le déploiement et la gestion des applications en production.

Évolutions fonctionnels

Ajouter une IA au jeux de Pokémon

L'un des modes de jeu qui sera proposé est un match en local contre des bots entraîné à l'aide des algorithmes de l'intelligence artificielle.

Proposer d'autres jeux dans la plateforme

On va créer une interface Game dans le service Match qui sera implémentée par chaque jeu et qui va être injectée à la création d'un match. De cette façon, on peut avoir plusieurs modes de jeu et on peut ajouter d'autres modes sans modifier le code métier juste en

ajoutant une autre classe qui implémente l'interface Game. Il s'agit de l'injection de dépendance.

Annexe : Documentation de l'API

L'API est documentée à l'aide de la spécification OpenAPI. Cette approche offre plusieurs avantages, notamment la génération automatique de la documentation de référence pour l'API. En utilisant OpenAPI, nous visons à améliorer la clarté, la cohérence et l'accessibilité de la documentation de l'API, facilitant ainsi sa compréhension et son utilisation par les développeurs et les utilisateurs.

Vous trouverez la documentation de chaque API dans le répertoire GitHub dans le fichier: **service_name-service-openapi.yml**

Afin de bien visualiser les documentations, veuillez les ouvrir avec un plugin openApi adéquat avec votre IDE ou bien en ligne sur le site <https://editor.swagger.io/> en important le fichier dans l'interface.

API User:

<https://github.com/moDallel/ChampionSphere/blob/main/user/user-service-openapi.yaml>

API Match:

<https://github.com/moDallel/ChampionSphere/blob/main/match/match-service-openapi.yml>

API Creature:

<https://github.com/moDallel/ChampionSphere/blob/main/creature/creature-service-openapi.yml>

API Auth:

<https://github.com/moDallel/ChampionSphere/blob/main/auth/auth-service-openapi.yaml>

Annexe : Présentation

La présentation du projet est disponible sur Canva:

https://www.canva.com/design/DAF2xV3aUiM/Xp9Ag1a5y4pq0GZCmPP1WQ/edit?utm_content=DAF2xV3aUiM&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

