Monster Trading Card Game (MTCG) - App-Design-Dokumentation

1. Überblick

Das MTCG-Projekt ist eine RESTful-Serveranwendung, die für ein Sammelkartenspiel entwickelt wurde. Es bietet Funktionen für die Benutzerregistrierung, den Login, die Kartenverwaltung und Kämpfe zwischen Benutzern. Dieses Dokument bietet einen Überblick über die Architektur der App, einschließlich der Komponenten, ihrer Rollen und ihrer Interaktionen.

2. Architektur

Die Anwendung folgt einer geschichteten Architektur, bei der verschiedene Komponenten für unterschiedliche Funktionalitäten verantwortlich sind. Die Hauptkomponenten umfassen Modelle, Handler, Utility-Klassen und den REST-Server.

2.1 Komponenten

2.1.1 Modelle

Dieses Package enthält alle Klassen, die die Spiellogik abbilden.

- **User**: Repräsentiert einen Spieler. Ein Benutzer hat einen Benutzernamen, ein Passwort, standardmäßig 20 Münzen, einen Kartenstapel und ein Deck für Kämpfe. Der Benutzer kann sich registrieren, anmelden, Pakete kaufen, ein Deck definieren, Karten tauschen und an Kämpfen teilnehmen.
- **Card**: Repräsentiert eine allgemeine Karte im Spiel. Jede Karte hat eine ID, einen Namen, einen Typ und einen Schadenswert.
- **Package**: Repräsentiert eine Sammlung von Karten, die ein Benutzer kaufen kann. Jedes Package enthält fünf Karten.
- **TradingDeal:** Ermöglicht das Tauschen von Karten zwischen Spielern. Ein Deal spezifiziert eine Karte, die angeboten wird, sowie eine Bedingung, die erfüllt sein muss, um den Handel abzuschließen (z.B. Mindestschaden oder Kartentyp).

2.1.2 Controller

Dieses Package enthält die Kernlogik der Anwendung.

- **UserService:** Verwaltet Benutzeroperationen wie Registrierung, Login und Profilverwaltung.
- **BattleService:** Organisiert Kämpfe zwischen Spielern, überprüft Regeln und verwaltet das Ergebnis.
- **CardService:** Verwaltet die Karten eines Benutzers, ermöglicht den Kauf von Paketen und das Management des Kartenstapels.
- **DeckService:** Ermöglicht Spielern, ihr Deck zu verwalten und Karten auszuwählen.
- **PackageService:** Verwaltet den Erwerb und die Verwaltung von Kartenpaketen.
- **StatsService:** Erfasst und verwaltet Spielerstatistiken und Ranglisten.
- **TradingService:** Erlaubt Spielern, Karten zu tauschen, indem Handelsangebote erstellt, akzeptiert oder abgelehnt werden.
- **TransactionService:** Verwaltet Transaktionen im Spiel, z.B. den Kauf von Paketen.

2.1.3 Repository

Dieses Package enthält alle Datenbank-Operationen und den Zugriff auf persistente Daten.

- **DatabaseManager:** Stellt eine Verbindung zur PostgreSQL-Datenbank her.
- **UnitOfWork:** Unterstützt Transaktionen und sichert Konsistenz in der Datenbank.
- **BattleRepository:** Speichert und lädt Daten für die Kampflogik.
- **CardRepository:** Verwaltet die Speicherung und den Abruf von Karten.
- **DeckRepository:** Speichert und verwaltet Deck-Zusammenstellungen.
- **PackageRepository:** Organisiert Kartenpakete und deren Kauf.
- **StatsRepository:** Verarbeitet Spielerstatistiken und Ranglisten.
- **TradingRepository:** Handhabt Handelsangebote und deren Transaktionen.
- **UserRepository:** Verwaltet Benutzerinformationen und Authentifizierung.

3. Zeiterfassung

Geschätzt wurden für das Projekt 150h Zeit aufgewandt

4. Design-Entscheidungen

- **Geschichtete Architektur**: Die Anwendung ist in Modelle, Controller und Repositorys unterteilt, um eine klare Trennung der Verantwortlichkeiten zu gewährleisten, was die Wartbarkeit und Erweiterbarkeit des Systems erleichterten.
- **Token-basierte Authentifizierung**: Benutzer authentifizieren sich mit Tokens, die nach erfolgreichem Login generiert werden. Diese Tokens werden in der Datenbank gespeichert und für nachfolgende Anfragen validiert, um die Endpunkte abzusichern.
- **Passwortsicherheit**: Passwörter werden mit einem einzigartigen Salt gehasht, um die Sicherheit zu gewährleisten. Dadurch wird sichergestellt, dass selbst bei einem Datenbankkompromiss die Benutzerpasswörter nicht leicht zugänglich sind.

5. Unique Feature: Terrain-Effekte in Kämpfen

Einzigartig in dieser Implementierung ist die Einführung von Terrain-Effekten in den Kämpfen. Vor jedem Kampf wird zufällig ein Terrain ausgewählt, das den Schadensausgang beeinflusst:

- Ozean: Wasser-Karten erhalten +50% Schaden, Feuer-Karten erleiden -50% Schaden.
- Vulkan: Feuer-Karten erhalten +50% Schaden, Normal-Karten erleiden -50% Schaden.
- Gebirge: Normal-Karten erhalten +50% Schaden, Wasser-Karten erleiden -50% Schaden.

6. Unit Tests

6.1 UserTests (Benutzerverwaltung)

Diese Tests prüfen die Registrierung, Anmeldung und Aktualisierung von Benutzerprofilen.

- Registrierung:
    - Erfolgreich mit gültigen Daten (201 CREATED).
    - Fehler bei bereits vorhandenem Benutzer (409 CONFLICT).
    - Fehler bei ungültigen Eingaben (400 BAD_REQUEST).

- Login:
    - Erfolgreich mit gültigen Anmeldeinformationen (200 OK).
    - Fehler bei falschem Passwort (401 UNAUTHORIZED).
    - Fehler bei fehlenden Anmeldeinformationen (400 BAD_REQUEST).
- Profilaktualisierung:
    - Erfolgreich mit gültigen Daten und Token (200 OK).
    - Fehler bei ungültigem Token (403 FORBIDDEN).
    - Fehler bei fehlenden Feldern (400 BAD_REQUEST).

6.2 PackageTests (Pakete verwalten)

Testet die Erstellung von Paketen mit 5 Karten durch einen Administrator.

- Erstellung:
    - Erfolgreich mit gültigen Daten (201 CREATED).
    - Fehler bei ungültiger Authentifizierung (403 FORBIDDEN).
    - Fehler, wenn das Paket weniger als 5 Karten enthält (400 BAD_REQUEST).

6.3 TransactionTests (Paketkauf)

Testet den Kauf von Kartenpaketen durch Benutzer.

- Kauf von Paketen:
    - Erfolgreich mit gültigem Token und genügend Coins (201 CREATED).
    - Fehler bei fehlendem Token (401 UNAUTHORIZED).
    - Fehler bei unzureichenden Coins oder fehlenden Paketen (400 BAD_REQUEST).

6.4 StatsTests (Statistiken & Scoreboard)

Testet die Anzeige von Benutzerstatistiken und der Bestenliste.

- Benutzerstatistiken:
    - Erfolgreich mit gültigem Token (200 OK).
    - Fehler bei ungültigem Token (401 UNAUTHORIZED).
    - Fehler, wenn der Benutzer nicht existiert (404 NOT_FOUND).
- Scoreboard:
    - Erfolgreich, gibt die Liste der besten Spieler zurück (200 OK).

6.5 DeckTests (Deck-Konfiguration)

Testet das Erstellen und Validieren eines Decks mit genau vier Karten.

- Deck-Konfiguration:
    - Fehler bei fehlendem oder ungültigem Token (401 UNAUTHORIZED).
    - Fehler, wenn weniger als vier Karten angegeben sind (400 BAD_REQUEST).