Zwischenabgabe Protokoll

Julia Pabst

Github Link: https://github.com/JuliaPabst/Monster-Trading-Cards-Game

1. Serverarchitektur

Der Server ist modular aufgebaut und in verschiedene Komponenten unterteilt, die den gesamten HTTP-Kommunikationsprozess unterstützen:

- **Request- und Response-Klassen**:
  - Die Request-Klasse repräsentiert eingehende HTTP-Anfragen.
  - Die Response-Klasse generiert und verwaltet die HTTP-Antworten.
  - Status und HTTP-Methoden werden durch **Enums** (z. B. Status, Method) definiert, um typsichere und leicht erweiterbare Werte zu gewährleisten.
- **Hilfsfunktionen (Utility-Funktionen)**:
  - HttpRequestParser: Verantwortlich für das Parsen der eingehenden HTTP-Anfragen und das Extrahieren relevanter Daten.
  - HttpResponseFormatter: Formatierung der HTTP-Antworten in ein korrektes HTTP-Protokollformat.
  - HttpSocket: Kommunikation zur Annahme und Beantwortung von Client-Anfragen.
- **Server-Komponenten**:
  - **Request Handler**: Nimmt eingehende Anfragen entgegen, verarbeitet sie und gibt eine Antwort zurück.
  - **Server**: Die Server-Klasse öffnet einen Server-Socket auf Port 10001, akzeptiert eingehende Client-Verbindungen und delegiert deren Verarbeitung an einen RequestHandler, der mit der übergebenen Application-Instanz arbeitet.
  - **Application Interface**: Definiert die Interaktion zwischen Server und der zugrunde liegenden Applikation, wodurch die Serverlogik unabhängig von der Applikation bleibt.

2. Applikationsstruktur

Die Applikation ist nach dem Prinzip der Schichtenarchitektur in drei Layer aufgeteilt, die klar voneinander getrennt sind, um Flexibilität und Austauschbarkeit zu gewährleisten:

1. **Presentation Layer**:
   - UserController: Verarbeitet Anfragen zur Benutzerverwaltung (z. B. Registrierung).
   - SessionController: Verantwortlich für die Benutzeranmeldung und Token-Generierung.

    - **Controller Interface**: Definiert allgemeine Methoden, die alle Controller implementieren müssen.
2. **Application Layer**:
    - TokenService: Handhabt die Logik zur Generierung und Validierung von Tokens.
    - UserService: Verwaltet die Geschäftslogik für Benutzeroperationen, wie das Erstellen und Suchen von Benutzern.
3. **Data Layer**:
    - Beinhaltet die Datenzugriffsschicht:
        - UserRepository: Abstraktes Repository zur Verwaltung und Speicherung von Benutzerinformationen in der Applikation.

3. Datenklassen

Die Applikation verwendet mehrere Datenklassen, um die Domain-Objekte zu modellieren. Hier werden jene genannt, die bereits in Verwendung sind:

- User-Klasse: Repräsentiert Benutzer und deren Attribute
- Card-Klasse: Repräsentiert eine Karte, die in Decks und Stacks verwendet wird
- TokenRequest-Klasse: Repräsentiert Anfragen zur Token-Authentifizierung