Protokoll - Monster Trading Card Game (MTCG)

Projektentwurf und technische Umsetzung

Initialisierung und Serverstart

Die Main Klasse erstellt die Datenbanken Tabelle und startet den Server.

Server- und Client-Management

Die Server Klasse hört auf eingehende Verbindungen auf dem Port 1001. Für jede eingehende Verbindung wird ein neuer ClientHandler Thread erzeugt. Der Server agiert als die zentrale Schnittstelle, die Client-Anfragen empfängt. Und die Bearbeitung an den ClientHandler weitergibt.

Der ClientHandler ist zuständig für die direkte Kommunikation mit dem Client. Er liest die eingehenden http-Anfrage und extrahiert die wichtigen Informationen wie die gewünschte Route, HTTP-Methode und Daten aus dem Request. Nach der Verarbeitung der Anfrage leitet der ClientHandler diese Informationen an den Router weiter.

Anfrageverarbeitung und Routing

Die Request-Klasse repräsentiert eine eingehende HTTP-Anfrage. Sie enthält wichtige Informationen wie die HTTP-Methode, den Pfad der angeforderten Ressource, Header und den Body der Anfrage. Der ClientHandler nutzt diese Klasse, um eine strukturierte und leicht verständliche Repräsentation der eingehenden Anfrage zu schaffen.

Der Router empfängt die verarbeiteten Anfragedaten vom ClientHandler und entscheidet, welche Aktion basierend auf dem Pfad und der Methode der Anfrage ausgeführt werden soll. Er leitet die Anfrage an den entsprechenden Controller weiter, der für die spezifische Geschäftslogik zuständig ist. Der Router spielt eine zentrale Rolle in der Bestimmung des Flusses innerhalb der Anwendung.

Antworterstellung

Der Respone wird nach der Verarbeitung durch den entsprechenden Controller generiert. Diese Klasse repräsentiert die HTTP-Antwort und beinhaltet Statuscode, Inhaltstyp und den tatsächlichen Inhalt. Der Controller gibt die Response zurück zum Router, der sie an den ClientHandler weiterleitet. Der ClientHandler sendet dann die strukturierte Antwort an den Client. Diese Prozesskette gewährleistet eine klare Trennung der Verantwortlichkeiten und effiziente Antworterstellung.

HTTP-Kommunikation

Die Methode Klasse beinhaltet ein Enum, das verschiedene HTTP-Methoden (GET, POST, PUT und DELETE). Sie dient dazu den Typ der eingehenden Anfrage zu bestimmen.

Die HTTPStatus Klasse beinhaltet ein Enum, das mögliche HTTP-Statuscodes repräsentiert, wie 200 OK oder 404 NOT_FOUND. Sie wird für die Übermittlung des Anfrageergebnisses an den Client verwendet.

Die ConentType Klasse beinhalten ein Enum, das verschiedene Arten von Content-Types (text/html, text/plan und application/json) definiert. Es dient dazu den Clients das Format des übertragenen Inhalts mitzuteilen.

Controller

Die Controller Klassen (User, Cards, Package, Game, Trading) beinhalteten unterschiedliche Funktionalitäten. Jeder Controller empfängt Anfragen vom Router, führt Geschäftslogik aus und gibt entsprechende Antworten zurück.

Service

Die Service Klassen (Auth, User, Cards, Package, Game, Trading) unterstützten die Controller durch Bereitstellung von Diensten und Hilfsfunktionen, die für Geschäftslogiken wie Benutzerverwaltung, Authentifizierung, Trading benötigt werden.

Models

Das Card Modell speichert wichtige Informationen über die Spielkarten, darunter Namen und Schaden, welche im Zentrum der Spielstrategie stehen.

Das Stats Modell zeichnet die Spielstatistiken der Spieler auf, einschließlich ihrer Siege und Niederlagen, was für die Bewertung ihrer Leistung und für Wettkämpfe relevant ist.

Das TradeOffer Modell ist für das Verwalten von Handelsangeboten zuständig und hält Details zu den angebotenen und geforderten Karten fest, was den Spielern ermöglicht, aktiv am Kartenmarkt teilzunehmen.

Schließlich gibt es das UserData Modell, das persönliche Informationen der Spieler wie Namen und Biografie enthält, was vor allem für die Benutzerverwaltung und Personalisierung des Spielerlebnisses wichtig ist.

Unit-Tests

Es werden eine Vielzahl von Funktionen durch Unit-Tests überprüft, um die Funktionalität und Sicherheit des Systems zu gewährleisten.

Der testRegisterUser stellt sicher, dass Benutzer sich erfolgreich registrieren können, eine grundlegende Voraussetzung für die Teilnahme am Spiel. Parallel dazu prüft der testLoginUser, ob Benutzer sich einloggen können, was für den Zugriff auf das Spiel essenziell ist.

Um die Benutzerdaten zu verwalten und zu aktualisieren, gibt es den testUpdateUser, während der testSaveUserToken und testDeleteUserToken sicherstellen, dass Benutzersitzungen sicher gehandhabt werden. Der testExtractUserIdFromAuthHeader gewährleistet, dass die Benutzeridentifikation aus dem Authentifizierungsheader korrekt extrahiert wird, was für die Sicherheit und Benutzerverwaltung wichtig ist.

Die Integration und Verwaltung der Spielkarten wird durch den testConvertJsonToCards getestet, der prüft, ob Karteninformationen korrekt aus JSON konvertiert werden. Der testCheckCardsExistence hilft, die Spielintegrität zu wahren, indem er überprüft, dass keine

doppelten Karten existieren. Der testGetCardsForUserReturnsCorrectCards stellt sicher, dass Benutzer ihre Karten korrekt abrufen können, und der testConvertToJsonReturnsValidJson überprüft die korrekte Datenkommunikation.

Für das Gameplay ist es wichtig, dass Spieler ihre Decks gemäß den Regeln konfigurieren können. Dies wird durch den testIsDeckSizeValidWithInvalidSize und den testConfigureDeckForUserSuccessfullyConfiguresDeck überprüft. Um den Fortschritt der Spieler zu verfolgen und Wettbewerb zu fördern, gibt es den testGetStats und den testGetScoreboard.

Die Kämpfe im Spiel und die Auswahl der Gegner sind essentielle Elemente, die durch den testGetOpponent und den testBattle überprüft werden. Die Verwaltung und Aktualisierung des Spiels wird durch testUpdateScoreboard und testHandleCreatePackageAsAdmin überprüft.

Schließlich wird die Interaktion mit der Spielökonomie und der Handel zwischen den Spielern durch Tests wie testHandleAcquirePackage, testHandleCreatePackageWithExistingCards, testHandleGetTradingDeals, testHandleCreateTradingDeal und testHandleDeleteTradingDeal überprüft.

Unique Feature: Logout-Funktion

Die Logout-Funktion im Spiel ermöglicht Spielern, sich abzumelden. Diese benutzen dafür den Pfad /logout über eine POST-Methode. Die Funktion im UserController überprüft dann den Benutzer-Token und löscht ihn, was die Sitzung des Spielers beendet.

Gewonnene Erkenntnisse

Während Projekts habe ich einige wertvolle Erkenntnisse gewonnen. Besonders die Wichtigkeit von sauberem und modularisiertem Code wurde mir bewusst, der das Identifizieren und Beheben von Fehlern erleichtert und das Hinzufügen neuer Funktionen vereinfacht. Zudem habe ich praktische Erfahrung mit den Tools Curl, Java, PostgreSQL und Docker gesammelt, was mein technisches Verständnis erweitert hat. Die Bedeutung und Anwendung von Unit-Tests haben mir auch gezeigt wie wichtig es ist die Qualität und Funktionalität des Codes kontinuierlich zu überprüfen.

Aufgewendete Zeit

Das gesamte Projekt hat insgesamt 53 Stunden in Anspruch genommen, wobei die aufgewendete Zeit mit WakaTime getrackt wurde. Hier ist der Link zur Aufzeichnung: https://wakatime.com/@018b6d16-b576-4d15-8e74-23440d32542a/projects/qjnhhifall?start=2023-12-31&end=2024-01-06

Git-Repository

Das Git-Repository für das Projekt befindet sich unter: https://github.com/magnusgoeppel/MTCG