Protokoll Monster Card Game SWEN

GIT: https://github.com/RxndyOG/SoftwareEngStefanWerner.git

Server eigenschaften:

Der Server verwendet einen TcpListener, um auf Verbindungen von Clients auf einem angegebenen Port zu warten. Nach dem Aufbau einer Verbindung wird ein NetworkStream verwendet, um Daten zu empfangen und zu senden.

Es werden HTTP Methoden wie GET, PUT und POST vom Client erkannt und geparsed.

Datenstrukturen bzw. Datentypen:

User werden als klasse in einer Liste gespeichert. Das selbe wird mit packages gemacht. User sind eine große klasse die Listen beinhält für die Decks und den Stack. Die Listen haben den Datentyp Cards. Cards ist eine eigene Klasse mit zwei Unterklassen, Monstercards und Spellcards.
Die Package Klasse besteht aus einer Liste die die unterschiedlichen karten speichert. Hier wird wieder die selbe Cards Klasse benutzt wie beim User.

Fürs speichern von werten im Body der Nachricht die der User schickt wird ein Dictionary benutzt. Hier werden die Namen der JSON Objekte zb. „Password“ etc... als Key benutzt um später darauf zugreifen zu können.
Das selbe wird mit den Headers gemacht jedoch werden diese nochmal abgefragt ob sie den richtigen Header für angefragte Funktionen haben. Z.B. „Authorization“.

Funktionen Server:

Je nachdem welche URL gegeben ist wird eine andere Funktion aufgerufen. Dies passiert mittels eines Switch Blocks. Es wäre ebenfalls mit einem Dictionary gegangen nur dies verbraucht mehr Speicher. Der geschwindigkeitsunterschied zwischen beiden ist so gering das geringerer Speicher hier bevorzugt wurde.

Jede Funktion hat einen bestimmten Zweck.

- HandleUserControllPUT: Diese Funktion liest die vom User eingegebenen Profil daten ein und speichert die neuen daten des Users (z.B.: Image oder Bio).
- HandleUserControllGET: Diese Funktion wie der Name schon versucht näher zu bringen „bekommt“ die daten des Users. In dem Fall werden die Profildaten des Users ausgegeben.
- HandleStackCards: Diese Funktion printed alle Karten im Stack des Users oder „No Cards in Stack“ wenn es keine gibt.
- HandleDeckCards: Diese Funktion printed alle Karten, die der User im Deck hat oder „No Cards in Deck“ wenn es keine gibt.

- HandlePutDeck: Diese Funktion lässt den User Karten wählen, die er dann in sein Deck geben kann.
- HandlePackages: Diese Funktion fragt ab, ob der User ein Admin ist, mittels Tokens, und wenn ja ließt es den Body ein und erstellt neue Karten bzw. Packages.
- HandleTransPackages: Hier wird sich darum gekümmert, ob der User genug coins hat und verarbeitet bzw. speichert die Karten im Stack des Users.
- HandleLogin: Hier wird der User getestet ob er schon existiert und wenn ja bekommt er einen Token.
- HandleRegisterUser: Diese Funktion erzeugt neue User.

Methoden im User:

Diese Methoden sollten alle selbstverständlich sein. Ich werde dennoch kurz etwas über sie schreiben.

- printDeck: printed das gesamte Deck
- printStack: printed den gesamten Stack
- tryPackages: dies Funktion wird bei der momentanen Abgabe nicht benutzt ist jedoch gedacht das sie testet ob der User genug coins hat und wenn ja kann er ein Package öffnen.
- changeAccount: hier werden die angegebenen User Profile Optionen in die Optionen geschrieben. Also das Profil wird mit neuen Daten aktualisiert.
- printUserData: printed alle Profil daten.
- addToStack: diese Funktion wird ebenfalls nicht benutzt und ist ein hinterbleibsel einer alten Version. Es wurde noch nicht gelöscht, weil momentan probiert wird den Stack sowie das Deck als Array zu speichern und nicht als Liste.