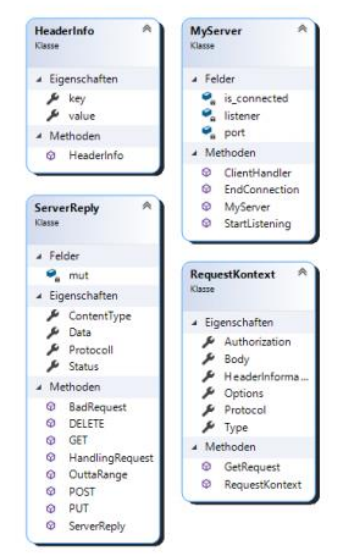
|  |
| --- |
| **MCTG UnitTests** |
| Tarek Abouzaid if18b119 |
| In diesem Protokoll wird grob erklärt, wie die einzelne Komponente des Programmes zusammenhängen und deren Funktionalitäten. Außerdem wird erläutert, warum ich mich für die Unitests entschieden habe, die im Programm geschrieben wurden. |
|  |

|  |  |  |
| --- | --- | --- |
| MCTG UnitTests  Tarek Abouzaid if18b119 MCTG Rest-Server Darunter ist der Server und seine Funktionalitäten implementiert. MCTG Class Library Hier befinden sich die Klassen die ich für die Funktionalität und die Logik hinter der Übung verwendet habe. MCTG Database Die Datenbank, wo die Informationen der User und der Karten gespeichert sind. Unit Tests Das testen der einzelnen Methoden, Klassen. |  | ***Zeitaufwand:*** • • • Für die Übung habe ich ungefähr 110 -150h gebraucht.Rest-Server + Klassendiagramm: 15-25hDatenbank + Klassendiagram: 10 -20hMCTG + Überlegung + getting started + Klassendiagramm: 70-90hProtokoll + Unittest + Git + Curl + Code verschönern: 15h |

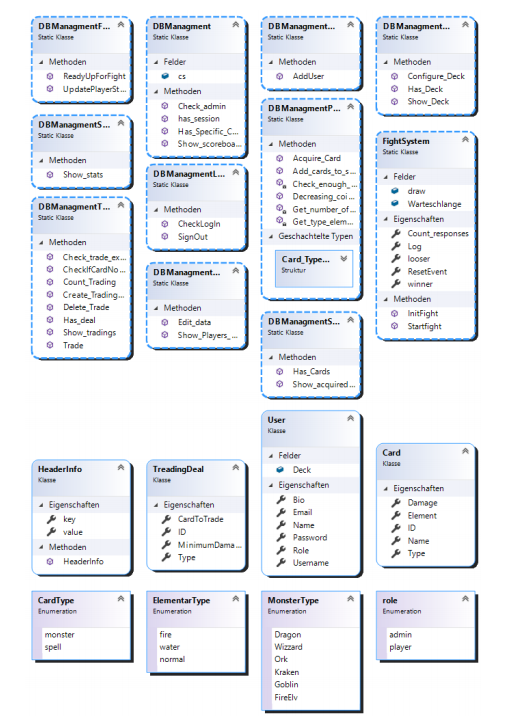
MCTG UnitTests



Die Klasse MyServer startet ein listener und akzeptiert dann einen Client. Aus dem Client wird der Request rausgelesen und der Klasse RequestKontext übergeben.

Diese Klasse tut aus dem Request die Informationen Zerlegen/aufteilen und als key : value speichern (mittels Klasse HeaderInfo). Aus den wichtigsten Informationen (Authorization, Body,Option, Protocol, Type) werden dann ein Objekt erstellt und der MyServer Klasse returned. Diese tut dann das Objekt der ServerReply Klasse übergeben, wo sich die Server Endpoints befinden und aus den Informationen des RequestKontext Objekt ein Response returned. Die Parameter der ServerReply Klasse sind die Informationen die dann im Response stehen.

MCTG UnitTests

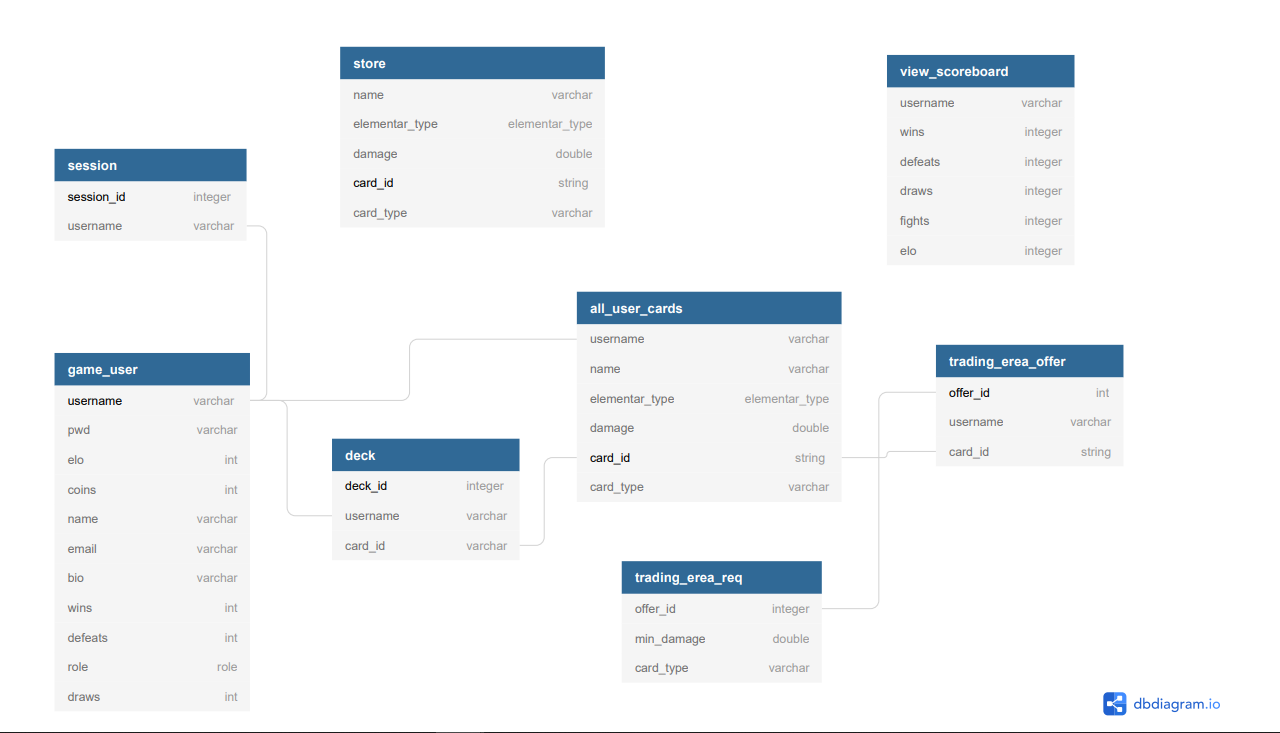


Die DBManamgent Klassen sind statische Klassen und werden von den Serverendpoints verwendet um auf den Request eingehen zu können. Diese holen sich dann die Informationen, die benötigt werden aus der Datenbank und arbeiten damit weiter. Im Namen der Klassen steht wofür die Klasse verwendet wird bsp

* DBManagmentAddUser
* DBManagmentDeck
* Usw..

Die anderen Klassen (User, Card, Treading Deal) werden dafür verwendet um aus dem payload des Request JASON zu parsen.

MCTG UnitTests



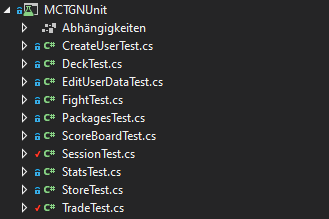
In der Datenbank werden die Informationen gespeichert, die zum erfolgreichen ausführen der Übung gebraucht werden.

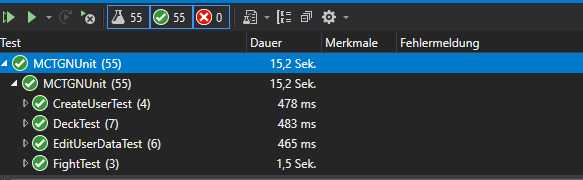
In der Tabelle game\_user befinden sich alle User die ein „Account“ besitzen. Darunter befinden sich normale User und Admins. Auch Admins sind in der Lage das Spiel zu spielen und beitzen keinerlei Vorteile den normalen User gegenüber. Die Tabelle all\_user\_cards soll den stack des uSers darstellen (alle Karten die der User besitzt). Karten kann man sich aus dem store kaufen wenn genug Karten vorhanden sind und der User genug Coins besitzt. Admins können Karten in den Store einfügen.

Die Tabelle session ist dafür da, um festzuhalten welche User eingeloggt sind. Nach dem erfolgreichen Einloggen kann der User die Funktionalitäten verwenden die das Programm anbietet.

Alle anderen Tabellen sind selbsterklärend.

MCTG UnitTests





Mit den Unittests habe ich die Endpoints (Methoden) des Server getestet mit allen möglichen Ausgängen und Fehler die auftreten können. Durch das Testen dieser Methoden, Klassen kann ich sicher gehen, dass die Clients auch die richtigen Responses bekommen und Fehler die auftreten können auch aufgefangen und richtig behandelt werden. Dieses Methoden rufen Außerdem andere Klassen und Methoden auf die für die Logik zuständig sind, somit decke ich großteil (meine Schätzung bei über 90%) des Codes ab.