Protokoll: Sports Exercise Battle

Design / Lessons Learned

Programmierstil auf Testbarkeit auslegen

Am überraschendsten war für mich wie wichtig es ist das Programm so zu gestalten, dass es im Nachhinein einfach getestet werden kann. Der Ansatz des test-driven developements wurde von mir in diesem Projekt leider nur sehr wenig verfolgt. Dies war retrospektiv betrachtet ein Fehler, da das Schreiben der Unit Tests dadurch sehr erschwert wurde. In Zukunft bietet es sich an für vergleichbare Projekte test-driven developement auszuprobieren, oder zumindest vermehrt auf die Testbarkeit zu achten.

Datenbank Handler sauberer gestalten

Für die Datenbank hätte ich mir im Vorhinein mehr Gedanken machen sollen, welche Funktionalitäten oft benötigt werden, und somit in eigene Methoden ausgelagert werden sollten. Dadurch ist der Code am Ende nicht so sauber geworden wie erwünscht.

Zeitaufwand

Feature	Zeitaufwand
Planning/Brainstorming	2 Stunden
REST-Server/Grundstruktur	4 Stunden (bzw. 15-20)*
Create User	2,5 Stunden
Login User / Token	1,5 Stunden
Profile Page	2 Stunden
Stats	2 Stunden
Scoreboard	2 Stunden
Tournament Logic	4 Stunden
Special Feature (World Record)	0,5 Stunden
History	2 Stunden
Dokumentation	3 Stunden
Unit Tests	6 Stunden

*Grundstruktur des REST-Servers war aus dem MTCG vorhanden und wurde wiederverwendet

Gesamtaufwand: 31,5 bzw. 36,5 – 41,5 Stunden

Beim Zeitaufwand ist mir zugutegekommen, dass ich die Grundstruktur des REST-Servers bereits vor dem Projekt implementiert hatte. Somit konnte in diesem Bereich einiges an Zeit gespart werden.

Umsetzung

Die Logik ist im Großen und Ganzen unterteilt in 5 unterschiedliche Klassen: „Server“, „Webhandler“, „ReqContext“, „TCP“ und „DatabaseHandler“.

Server:
„Server“ kümmert sich hauptsächlich um Threading für eintreffende Clients und liefert Statusinformationen.

Webhandler:
Hat primär die Aufgabe HTTP-Antworten zu senden und eingehende HTTP-Requests zu verwalten. Hat ein ReqContext Objekt zur logischen Abarbeitung der Requests und ein TCP Objekt.

ReqContext:
„ReqContext“ beinhaltet die Logik zum Reagieren auf eintreffende Requests. Je nachdem welche Informationen dem Header und dem Body entnommen werden können, reagiert ReqContext anders.

TCP:
Verwaltet die TCP-Verbindungen.

DatabaseHandler:
Ist der Schnittpunkt mit der Datenbank und verwaltet alle Zugriffe auf diese.

Unit Tests

Die Unit Tests sind so gewählt, dass die Grundfunktionalitäten der Request Verarbeitung überprüft werden. Es wird außerdem die Verbindung zur Datenbank getestet, sowie das Umschalten zwischen „es gibt ein aktives Turnier“ und „es gibt kein aktives Turnier“. Außerdem wird das grundsätzliche Beantworten von Requests an den Server getestet. Der Fokus habe ich bei den Unit Tests auf das Testen der Reaktionen bei unterschiedlichen Anfragemethoden und Anfragepfaden gelegt, da ich das für das Wichtigste gehalten habe und hier Fehler sehr leicht passieren können.

GitHub

https://github.com/HotDonut/SportExerciseBattle