**Protokoll**

Okay, das ist jetzt ein bisschen last-minute, also hab ich fürs erste keine „time spent with the project“ section – das wird ich mir aber später aus den commits rausholen (bzw tatsächlich tracken wann ich was mache…)

Unit Tests habe ich jetzt auch keine-

Aber so funktioniert der Code:

1. Gerade sind nur die Files Server/Server.cs, Routing/Router.cs und Routing/Parser.cs relevant. Ich habe zwar auch ein paar Controller usw. erstellt, aber mit denen habe ich noch nichts gemacht.
2. Router

* Bestimmt basierend auf path & method, die er vom Parser (also aus dem HttpRequest Objekt) bekommt, welchen string er returnen soll
* Den string der returned wird benutze ich dann im Server, um zu entscheiden was als nächstes gemacht wird

1. Parser

* Der Parser parst eingehende HTTP requests in ein HttpRequest Objekt (HttpRequest class ist ganz oben (über der Parser class) definiert)
* Initialisiert StreamReader um vom input Stream zu lesen
* Liest die erste line, teilt sie in 3 (method, path, und http version)
* Liest die nächsten lines und speichert sie in die Header Dictionary, bis es zu einer leeren line kommt
* Dann liest es den body (falls es einen gibt (festgestellt durch content length))
* Returned das HttpRequest Objekt
* Die BodyParser method extrahiert username & password und returned diese

1. Server

* Die Task StartAsync startet den server (macht einen neuen TcpListener und startet ihn; wartet dann in einer Endlosschleife auf Clients; wenn sich ein Client verbindet, ruft er die HandleClientAsync Methode auf)
* HandleClientAsync initialisiert einen Streamwriter (um später http responses zu returnen), holt sich das HttpRequest Objekt vom Parser und schickt es an den Router, welcher dann einen string returned, der dann im switch statement den nächsten Schritt bestimmt -> habe ganz unten in der Server class die Methoden NotFound, BadRequest, GenerateResponse, WriteResponse, Login, und Register
* Hab jetzt leider keine Zeit mehr alles genau zu erklären, aber die Register Methode speichert neue User (falls sie noch nicht existieren) in einer json file und die Login Methode liest auch aus der json file (um username & password aud Gültigkeit zu kontrollieren)
* Leider habe ich jetzt auch die User class innerhalb der Server class definiert weils die schnellste Lösung war

Muss noch viel am Code ändern bis er mir passt aber ja