**Protokoll**

Okay, das ist jetzt ein bisschen last-minute, also hab ich fürs erste keine „time spent with the project“ section – das werd 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

**Final Hand-In Protokoll**

*Vorab sorry fürs Denglisch!*

**Time spent:** Die letzten ca. 40 Stunden vor der Abgabe *(Habe prokrastiniert, wie ich weiß nicht was)*

Ich schaue jetzt mal, dass ich die Must-Have Requirements hinbekomme, weil ich nicht durchfliegen will.

\*\*\*

**Gerade** (10h 12min vor Abgabe) **schaut die Lage folgendermaßen aus:**

* Uses C# or Java

Easy, das Projekt ist in C#

* Implements a server listening to incoming clients

Ja, gottseidank habe ich das auch schon, weil es schon bei der Intermediate Abgabe eine Voraussetzung war (siehe Server.cs: die Server.StartAsync() Methode listened für incoming Verbindungen auf Port 10001)

* Implements multiple threads to serve client requests

Ja, hab auch in inline comments erklärt, warum das so ist. Evtl. füg ich ein Screenshot davon ein.

* Does not use an HTTP helper framework

Yup, erfüllt. Weil ich meine eigene parsing- und routing logic implementiert habe.

* Uses a Postgres Database for storing data

Noch nicht!

* Does not allow for SQL injection

Noch nicht und ganz ehrlich hab ich noch keine Ahnung, wie man das macht.

* Does not use an OR-Mapping Library

Nope.

* Implements at least 20 Unit Tests

Noch nicht, aber das geht dann eh relativ schnell.

Also, die **next things to tackle** sind:

Postgres Database,

Schutz vor SQL injection,

20 Unit Tests

**Random Gedanken/Bemerkungen** (9h 7min vor Abgabe)**:**

* Hätte früher anfangen sollen, eigentlich macht das eh Spaß
* Ich hab die letzte Stunde damit verbracht, mir meinen eigenen Code wieder zu erklären (deshalb die vielen inline comments lol)
* Werd jetzt damit weitermachen bis ich wieder alles verstehe und dann kommt die ProstgreSQL db)
* Noch eine Bemerkung beiseite: Ich würde echt gern die Login und Register Methoden in eine separate File (k.A., z.B. „UserServices.cs“ oder so) moven statt sie im Server.cs zu haben, weil ich das kinda messy finde, aber ich hab nur noch 9h also ist das gerade keine Priorität lol

**Random Gedanken/Bemerkungen** (7h 17min vor Abgabe)**:**

* Es sind nur noch 7 Stunden bis zur Abgabe und alles was ich gemacht hab, ist meinen Server.cs code im Detail zu kommentieren 💀 Naja, egal. Wenigstens versteh ich meinen code jetzt besser….

A screenshot of a computer program

Description automatically generated

What clinical insanity looks like

* Jetzt werde ich noch meine