

# LABORPROTOKOLL

im Studiengang Informatik / Computer Science  
Lehrveranstaltung Software Engineering Labor 1

## Final Hand-In

Ausgeführt von: Johanna Nasr  
Personenkennzeichen: if23b284

Begutachter: Hr. Michael Strommer

Wien, 07.01.2025

# Inhaltsverzeichnis

1	Allgemeines Appdesign.....	3
1.1	Bewusste Entscheidungen.....	3
1.2	Struktur.....	5
1.3	Klassendiagramm .....	8
1.4	Lessons learned.....	8
1.5	Unit testing decisions .....	9
1.6	Tracked time .....	9
1.7	Git-Link .....	9

# 1 Allgemeines Appdesign

In meinem Projekt "MCTG" habe ich mich dafür entschieden „Httpserverbase“ als Vorlage zu verwenden. Erst gegen Ende ist mir das Github-Projekt „SvenLayeredArchitecture“ wieder in den Sinn gekommen, welches ich bei besserer Zeiteinteilung meinerseits als Vorlage genommen hätte.

## 1.1 Bewusste Entscheidungen

Wir haben keine http Helper Frameworks verwendet, weshalb ich einen eigenen http Server für das Anfragen – Parsen und Antworten schicken, gebaut habe. JSON Serialisierung wird durch die Jackson Bibliothek übernommen.

Die Rest API mappt http requests, um die richtigen Handlermethoden aufzurufen. Die Routen sessions und users werden verwendet.

Die Token-based Security sagt uns, dass ein Session Token generiert wurde. Hier speichern wir aber alles mal in memory und in keiner Datenbank.

Bei der Struktur habe ich immer darauf geachtet, dass User und Sessions voneinander getrennt sind.

**Änderungen seit dem intermediate Hand-In:** Ich habe eine Datenbank in PostgreSQL erstellt mit der Tabelle users und den Spalten username, password, score, token und coins.

```
postgres=# SELECT * FROM users;
username | password | score | token | coins
-----+-----+-----+-----+-----
(0 Zeilen)
```

Der username wurde unique gemacht, damit dort keine ID benötigt wird.

```
postgres=# ALTER TABLE users
postgres=# ADD CONSTRAINT unique_name UNIQUE (username);
ALTER TABLE
```

Im DatabaseManager habe ich den User und das entsprechende Passwort hineingegeben. In der Main hatte ich dann eigentlich getConnection aufgerufen, nur leider hat das Verbinden da nicht funktioniert. Ich habe das Authentifizieren mit md5 in dem file „pg\_hba.conf“ angepasst. Dann habe ich auch das Passwort geändert, also alles zu „trust“ umgeschrieben und in der Psql Shell das Passwort für den user postgres bearbeitet, aber auch da kam dann dieser Fehler:

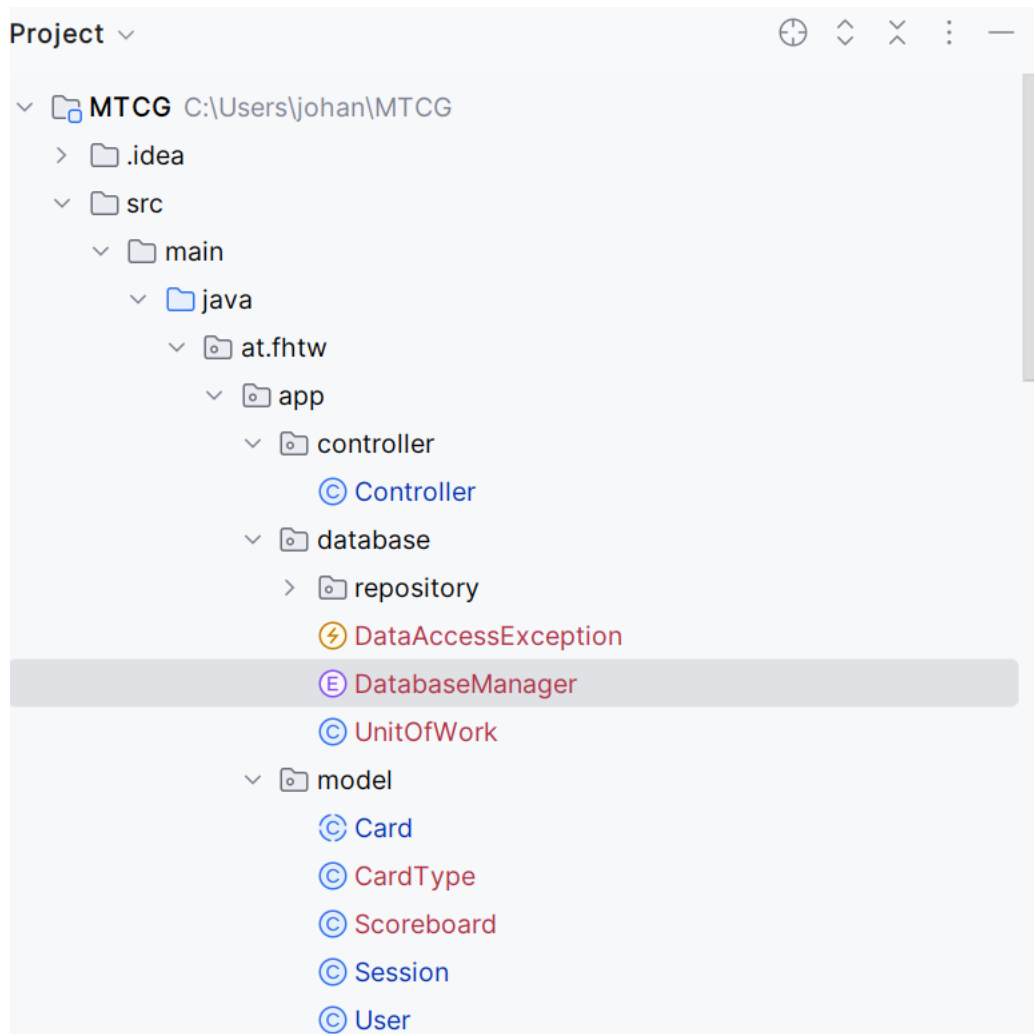
Caused by: org.postgresql.util.PSQLException Create breakpoint : FATAL: Passwort-Authentifizierung für Benutzer postgres fehlgeschlagen








Nachdem ich dann recherchiert habe und leider zu keiner Lösung gekommen bin, habe ich mich dazu entschieden in der Main die Datenbankverbindung auszukommentieren, was sehr schade ist, weil es anscheinend kein „schwieriger Fehler“ ist / war.

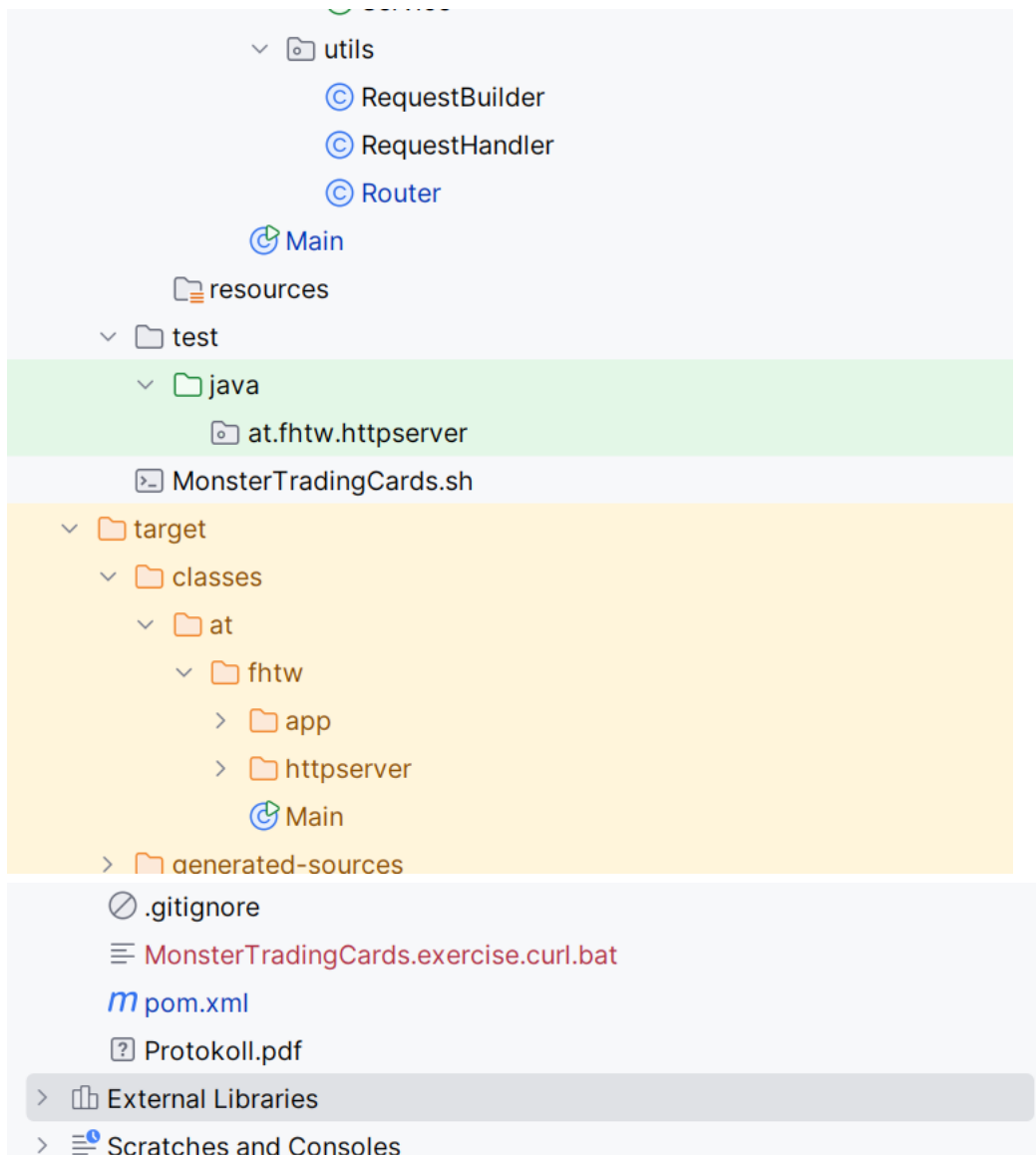
In dem Model-Package habe ich noch CardType und Scoreboard hinzugefügt, welche ich mit den entsprechenden Features gefüllt hätte. Ich habe ein BattleService erstellt, wo die Battle-Logic hätte drin sein sollen. Dadurch, dass die Datenbankanbindung nicht erfolgreich war, habe ich mich dazu entschieden, die Dummy-Daten-Dateien beizubehalten.

In der pom-Datei habe ich noch die benötigten Dependencies hinzugefügt. Beispielsweise für die Datei unitTest in dem testing-Package habe ich 20 theoretische Unit-Tests hinzugefügt, die ich dann noch in der Realität hätte ausprobieren wollen, wenn ich die entsprechenden Klassen mit den Methoden kreiert hätte. Derzeit sind die Klassen leer, aber vorhanden, weil ich mir zur Struktur Gedanken gemacht habe, aber nicht zur Funktionsimplementierung gekommen bin.

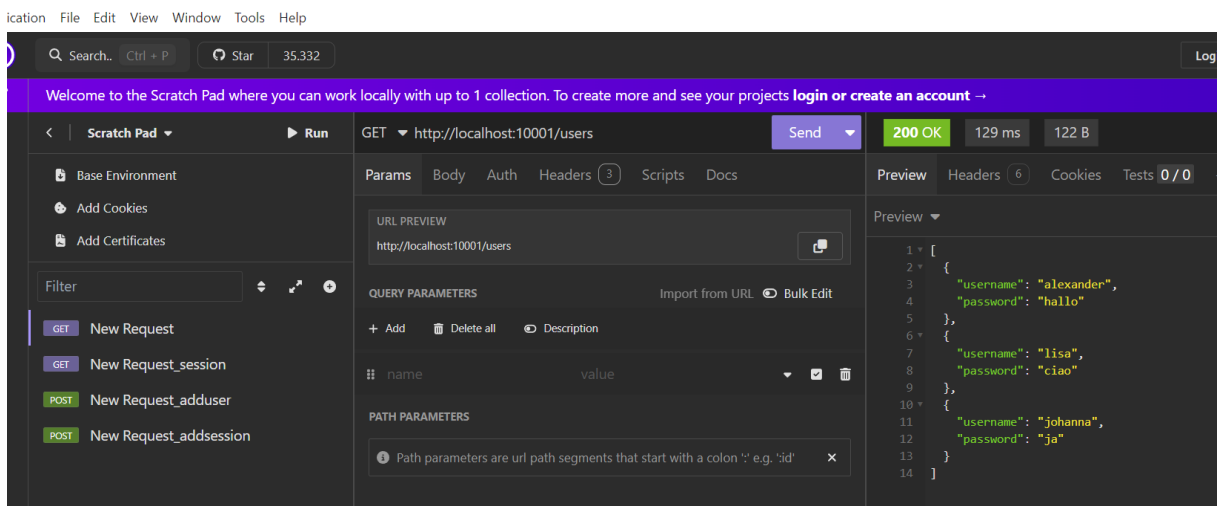
## 1.2 Struktur



- ▼  battle
  - © BattleService
- ▼  session
  - © SessionController
  - © SessionDummyDAL
  - © SessionService
- >  user
- ▼  testing
  - ☕ unitTest.java
- ▼  httpserver
  - ▼  http
    - Ⓔ ContentType
    - Ⓔ HttpStatus
    - Ⓔ Method
  - ▼  server
    - © HeaderMap
    - © Request
    - © Response
    - © Server
    - Ⓢ Service

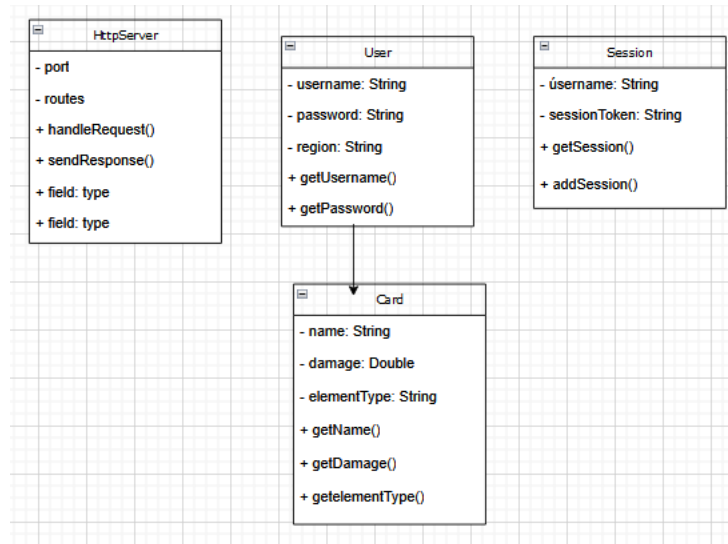


Jetzt ist alles auf Git gepusht und commitet, also keine rotgekennzeichneten Namen mehr.



Die Routen funktionieren.

## 1.3 Klassendiagramm



Änderungen zu dem intermediate Hand-In: Ich habe noch die Klassen Battle, CardType und Scoreboard hinzugefügt.

## 1.4 Lessons learned

Bei dem Projekt habe ich gelernt, was es heißt auf sich selbst gestellt ein Projekt zu recherchieren, studieren und zu implementieren. Ich habe verstanden, dass das Vorbereiten und Durchdenken einer Struktur bzw. vieler theoretischer Entscheidungen relevant und zeitintensiv ist und vor dem praktischen Umsetzen gemacht werden muss, damit das Projekt in Wunschvorstellung abläuft.

Zeitmanagement ist ein Muss (!), was man bei meinen fehlenden Teilen schon sieht. Ich hätte mich vor den Ferien an die theoretischen Ansätze setzen müssen, damit das Programmieren selbst in den Ferien nur mehr ein „Abarbeiten von Punkten“ ist. Stattdessen habe ich mich viele Tage mit beispielsweise der Datenbankbindung und den dabei entstandenen Problemen beschäftigt. Weiters habe ich den Error bzw. Bug, den ich noch vor den Ferien hatte, eine Woche lang bearbeitet, damit die mir unverständlichen Rätsel gelöst werden können. Ich weiß jetzt, wie ich das Projekt angehen würde, wenn ich die Chance hätte, nochmal im September bzw. Oktober damit zu beginnen.

Bei den git pushes bzw. commits hätte ich real arbeiten sollen, also nicht alles am letzten Tag pushen und commiten, sondern als ich daran gearbeitet habe. So habe ich immer so viele Befehle an einem Tag mit eigentlich „verfälschtem“ Datum. Im echten Software-Engineering-Leben würde das kein gutes Bild machen.



## 1.5 Unit testing decisions

Hierbei habe ich mich von theoretischen Szenarien leiten lassen. Die Angabe hat mir gesagt, welche Features in welcher Art und Weise implementiert sein sollen und diese habe ich dann in „theoretische“ Tests mithilfe von @Test geschrieben.

## 1.6 Tracked time

~September 2024: Auseinandersetzung mit dem Projekt = 13 Stunden

~Oktober 2024: Beginn Implementierung Projekt, Intermediate Hand-In, Auseinandersetzung mit Java = 20 Stunden

~November 2024: 10 Stunden

~Dezember 2024: 23.12.24-28.12.24: 2h/Tag = 12 Stunden, davor insgesamt 5 Stunden für Präsentation in der FH

~Jänner 2025: von 03.01.25-05.01.25: 2h/Tag = 6 Stunden

INSGESAMT = ~66 Stunden

Ich feiere orthodoxes Weihnachten, weshalb ich in der ersten Ferienwoche Zeit hatte und dann in der zweiten kurz vor der Abgabe gar nicht mehr. Zum Thema Zeitmanagement passend, wie oben erwähnt, hätte ich aus diesem Grund die erste Woche stärker ausnützen sollen, um das Projekt rechtzeitig abschließen zu können.

## 1.7 Git-Link

<https://github.com/NaJoKa/MTCG>