ePortfolio Modul 226B

## Intro

Der persönliche Lernfortschritt und die eingebrachten Nachweise sammeln und strukturieren Sie in diesem Dokument oder geben hier den Link bekannt, wodurch ich Zugriff auf Ihr ePortfolio habe. So könnte ich Sie in Ihren persönlichen Reflexionen und Erkenntnissen begleiten.

Zu allen Kompetenzen des Rasters, welche mit einem P versehen sind, dokumentieren Sie im ePortfolio folgende Punkte:

* eine Beschreibung (womit und wie wird der Kompetenznachweis erbracht, Sie könnten z.B. Ihre Entwicklung aufzeigen)
* ein Verweis auf das Lernprodukt (Dokument, Diagramm, Programmteil, Programm)
* eine persönliche Reflexion

Sie dürfen Kompetenzen auch in einem Portfolio Eintrag kombinieren, wenn eine Kombination sinnvoll erscheint.

## Mögliche Leitpunkte für die Reflexion:

1. Das habe ich gemacht.

2. Das habe ich gelernt.

3. Das ist mir gut gelungen.

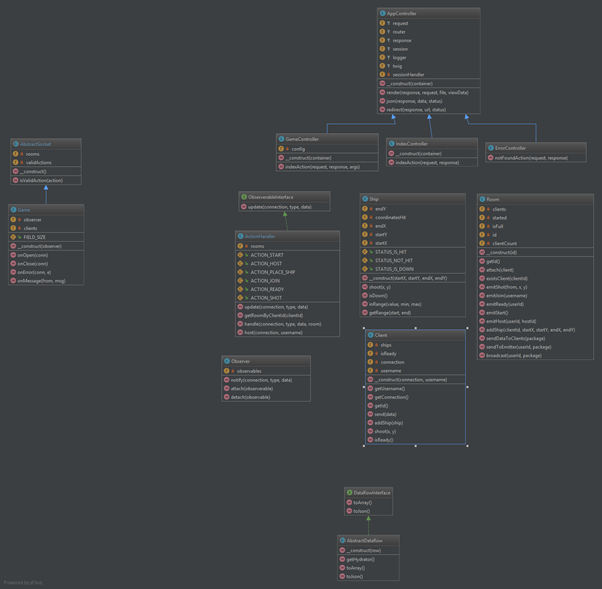
4. Damit hatte ich Schwierigkeiten

5. So habe ich auf die Schwierigkeiten reagiert.

6. Was kann ich jetzt mehr als vorher?

# 

# Design

Unser Design haben wir zeitgleich zum restlichen Projekt gemacht. Dies war vielleicht nicht ganz optimal, da man vor allem die UML Diagramme eigentlich zu Beginn des Projektes machen sollte und im Anschluss den Code danach strukturieren. Dafür konnten wir einige Teile des Diagramms automatisch per IDE generieren lassen, was auch sehr praktisch war.

Bei einem produktiven Projekt mit einem echten Kunden würde ich dies natürlich anders herum machen, sodass man dem Kunden etwas präsentieren kann. Da ich Applikationsentwickler lerne, macht mir die Softwareentwicklung natürlich mehr Spass, weshalb ich in privaten Projekten niemals auf die Idee käme, ein Klassendiagramm zu erstellen.

Natürlich haben wir in unserem Projekt, wo sinnvoll, auch abstrakte Klassen und Interfaces verwendet. Ein Beispiel dafür wäre der AbstractSocket.

Einige Teile des Diagramms haben sich mit der IDE, wie bereits erwähnt, generieren lassen. Die fehlenden Klassen nachzuführen war auch relativ einfach. Was jedoch Probleme bereitet hat, waren die jeweiligen Verbindungen zwischen den Klassen, da dies in unserer IDE irgendwie nicht richtig funktioniert hatte und immer wieder zurückgesetzt wurde. Allerdings konnten wir mit etwas Aufwand auch diese Hürde überwinden und das Diagramm sinnvoll gestalten.

Viel Neues habe ich nicht gelernt, da wir dies bereits in anderen Modulen sowie in ÜKs behandelt hatten.

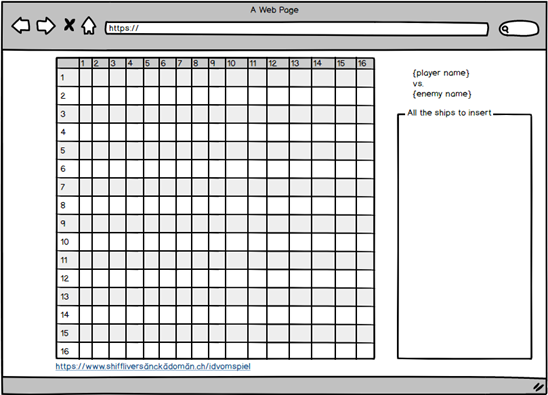
# Implementation

Bei diesem Punkt ging es um die eigentliche Entwicklung der Applikation. Meine primäre Arbeit war die Entwicklung im Backend. Dort habe ich mich sehr stark auch mit der Architektur auseinandergesetzt. Die Implementierung von Interfaces und abstrakten Klassen war bei der Umsetzung des Projektes sehr wichtig. Somit haben sich Code-Duplikationen verhindern lassen und die Lesbarkeit ist ebenfalls gestiegen.

Im Frontend habe ich, nach dem Marc zu mir gekommen ist und mich um Hilfe gefragt hat, alles weiterentwickelt, was nach dem Drag and Drop gekommen ist. Das korrekte platzieren der Schiffe sowie das anzeigen eines “Hits” oder das Benachrichtigen aufgrund von Events wurde von mir umgesetzt. Wichtig dabei zu erwähnen ist, dass ich bereits ein bisschen mehr Erfahrung mit den Tücken von JavaScript habe.

Trotzdem konnte ich noch vieles im Umgang mit Drag and Drop lernen.

Allgemein bin ich zufrieden mit der Aufteilung der Arbeit. Ich habe ein bisschen mehr Entwicklungsarbeit betrieben, während Marc sich um Dokumentationsarbeit gekümmert hat.



# 3-Schicht Architektur

Da wir sowohl in diesem Projekt als auch in unserem Lehrbetrieb mit PHP arbeiten, haben wir uns für die Architektur entschieden, mit welcher wir bereits vertraut waren - MVC (Model View Controller). Bei dieser Architektur befindet sich die gesamte Geschäftslogik im Model/Service Bereich. Die Anfragen an unseren Webserver werden von den Controllern empfangen und beantwortet. In der View Schicht findet man, wie sich vom Namen her erraten lässt, die UI Elemente der Applikation. Der Vorteil einer 3-Schicht Architektur besteht darin, dass jede Schicht unabhängig voneinander ausgetauscht werden könnte, was wir erreicht haben.

Da wir bereits seit über 1 ½ Jahren im Lehrbetrieb mit dieser Architektur arbeiten, konnte ich leider auch hier nicht viel Neues lernen. Alles in allem hatten wir bei der Implementation jedoch keine Probleme.

Allerdings gab es einige Verständnisprobleme, da diese Architektur in dieser Art anscheinend in Java nicht verwendet wird. Würde man dies so ähnlich in Java umsetzen, käme man je nach dem nur auf zwei Schichten. Da wir aber eine Webapplikation entwickeln, konnten wir uns mit der Lehrperson darauf einigen, dass dies so durchaus Sinn macht.

Die Drei-Schichten Architektur wurde ebenfalls “probiert” zu implementieren. Da wir keine Persistenzschicht brauchen, ist diese nicht implementiert. Jedoch wird die Präsentationsschicht durch die Controller ausgeliefert (HTML-Dateien => Präsentation).

Die Businessschicht ist in unseren Services/Models repräsentiert.

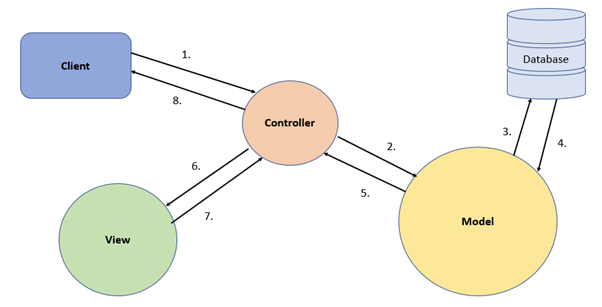
Der Websocket entspricht keinem der drei Layer. Am er agiert als Presentationlayer und Businesslogiclayer gleichzeitig. Jedoch ist er eher der Businessschicht ähnlich.

# 

# Version Control System

Damit wir sinnvoll am gleichen Projekt arbeiten können, brauchen wir natürlich ein Version Control System (VCS), auf welchem wir unseren Code zentral verwalten und lagern können. Da uns GitHub gut gefällt, haben wir uns für das VCS Git entschieden, mit welchem das eben erwähnte GitHub betrieben wird. Dazu mussten wir lediglich ein neues Repository anlegen, auf welches wir beide Schreibzugriff haben. GitHub Accounts hatten wir bereits zuvor, somit mussten wir diese nicht extra erstellen. Auch wenn dies zwar technisch möglich und auch oft so verwendet wird, fänden wir es doch beide etwas umständlich, unsere Änderungen jedes Mal per Command Line zu Commiten. Glücklicherweise bietet die IDE PHPStorm eine sehr gute Integration für Git, respektive GitHub.

Im Gegensatz zu Marc hatte ich keine Probleme mit dem Pushen auf GitHub. Ich hatte immer meinen eigenen Laptop dabei, welcher nicht über das “tolle” Schul-WLAN und einen Proxy gestört wurde.



Quelle: <https://journal.tekk.ch/doku.php/programmieren/server-client/mvc>

# Disclaimer

Dieses Dokument wurde in Zusammenarbeit von Marc Wilhelm und Björn Pfoster erarbeitet. Da wir beide an demselben Projekt gearbeitet haben, entsprechen unsere Interpretationen der angegebenen Kritikpunkte meist der gleichen Meinung.

Da jedoch gewisse Entwicklungen in dem Projekt individuell stattgefunden haben, wurden gewisse Passagen dieses Dokumentes dem entsprechend angepasst.