Abschlussprüfung Sommer 2023

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit



Entwicklung eines 2D – „Schiffe versenken“ Spiels  
zur Simulation einer Seeschlacht

Abgabedatum:

Neckargemünd,

den 25.08.2023

**Prüfungsbewerber:**

Denis Ojdanic

Martinstraße 4

89518 Heidenheim

**Ausbildungsbetrieb:**

SRH Berufsbildungswerk Neckargemünd GmbH

Im Spitzerfeld 25

69151 Neckargemünd

Inhaltsverzeichnis

## Inhaltsverzeichnis

[Inhaltsverzeichnis 2](#_Toc141176537)

[Abbildungsverzeichnis 3](#_Toc141176538)

[Tabellenverzeichnis 3](#_Toc141176539)

[Quellcodedokumentation 4](#_Toc141176540)

[Quellcodedokumentation 4](#_Toc141176541)

[Glossar 4](#_Toc141176542)

[1 Einleitung 5](#_Toc141176543)

[1.1 Projektumfeld 5](#_Toc141176544)

[1.2 Projektziel 5](#_Toc141176545)

[1.3 Projektbegründung 7](#_Toc141176546)

[1.4 Projektschnittstellen 7](#_Toc141176547)

[2 Projektplanung 7](#_Toc141176548)

[2.1 Projektphasen 7](#_Toc141176549)

[2.2 Ressourcenplanung 8](#_Toc141176550)

[2.3 Entwicklungsprozess 8](#_Toc141176551)

[3 Analysephase 8](#_Toc141176552)

[3.1 Ist-Analyse 8](#_Toc141176553)

[3.2 Wirtschaftlichkeitsanalyse 9](#_Toc141176554)

[3.2.1 Make or Buy-Entscheidung 9](#_Toc141176558)

[3.2.2 Projektkosten 9](#_Toc141176559)

[3.2.3 Amortisationsdauer 10](#_Toc141176560)

[3.3 Nutzwertanalyse 10](#_Toc141176561)

[3.4 Anwendungsfälle 10](#_Toc141176562)

[3.5 Qualitätsanforderungen 10](#_Toc141176563)

[3.6 Fachkonzept 11](#_Toc141176564)

[4 Entwurfsphase 11](#_Toc141176565)

[4.1 Zielplattform 11](#_Toc141176566)

[4.2 Architekturdesign 11](#_Toc141176567)

[4.3 Entwurf der Weboberfläche 12](#_Toc141176568)

[4.4 Datenmodell 12](#_Toc141176569)

[4.5 Geschäftslogik 12](#_Toc141176570)

[4.6 Maßnahmen zur Qualitätssicherung 13](#_Toc141176571)

[4.7 Pflichtenheft 13](#_Toc141176572)

[5 Implementierungsphase 13](#_Toc141176573)

[5.1 Implementierung der Datenstrukturen 13](#_Toc141176574)

[5.2 Implementierung der Weboberfläche 14](#_Toc141176575)

[5.3 Implementierung der Spiellogik 14](#_Toc141176576)

[5.3.1 Platzierung der Schiffe 14](#_Toc141176582)

[5.3.2 Anmeldung der Spieler 15](#_Toc141176589)

[5.3.3 Spielzüge 17](#_Toc141176590)

[5.3.4 Statistik 18](#_Toc141176591)

[6 Abnahmephase 19](#_Toc141176592)

[7 Dokumentation 19](#_Toc141176593)

[8 Fazit 20](#_Toc141176594)

[8.1 Soll-/Ist-Vergleich 20](#_Toc141176595)

[8.2 Gelerntes 21](#_Toc141176596)

[8.3 Ausblick 21](#_Toc141176597)

[9 Quellenverzeichnis 21](#_Toc141176598)

[10 Anhang 25](#_Toc141176599)

[Abbildung 1: Skizze der Webseite 26](#_Toc141176600)

[A6 Ereignisgesteuerte Prozesskette 30](#_Toc141176601)

[A7 Oberflächenentwürfe 30](#_Toc141176602)

[A8 Screenshots der Anwendung 31](#_Toc141176603)

[A9 Entwicklerdokumentation (Auszug) 32](#_Toc141176604)

[A10 Testfall und sein Aufruf auf der Konsole 33](#_Toc141176605)

[A11 Klasse: ComparedNaturalModuleInformation 35](#_Toc141176606)

[A12 Klassendiagramm 37](#_Toc141176607)

[A13 Benutzerdokumentation (Auszug) 38](#_Toc141176608)

**Abbildungsverzeichnis**

## Abbildungsverzeichnis

[Abbildung 1: Skizze der Webseite 22](#_bookmark56)

[Abbildung 2: Programmablaufplan zum Registrierungsvorgang zum Spiel y](#_bookmark59)

[Abbildung 3: Programmablaufplan zum Spielvorgang vi](#_bookmark60)

[Abbildung 4: Prozess des Einlesens eines Moduls vi](#_bookmark62)

**Tabellenverzeichnis**

## Tabellenverzeichnis

[Tabelle 1: Grobe Zeitplanung 7](#_bookmark13)

[Tabelle 2: Zeitvergleich 20](#_bookmark23)

[Tabelle 3: Detaillierte Ablaufplanung 25](#_bookmark31)

[Tabelle 4: Soll-/Ist-Vergleich 8](#_bookmark46)

[Tabelle 5: Detaillierte Zeitplanung ii](#_bookmark53)

## Quellcodedokumentation

## Quellcodedokumentation

[Listing 1: Testklasse xii](#_bookmark73)

[Listing 2: Klasse ComparedNaturalModuleInformation xiv](#_bookmark75)

**Dokumente**

## Dokumente

[Pflichtenheft xii](#_bookmark73)

[Benutzerdokumentation xiv](#_bookmark75)

**Glossar**

## Glossar

HTML Hypertext Markup Language

Auszeichnungssprache

CSS Cascading Style Sheets  
 Stylesheet-Sprache

JS JavaScript  
Skriptsprache

PHP Personal Home Page  
 Skriptsprache

JSON JavaScript Object Notation  
 Datenformat

PAP Programmablaufplan  
 Ablaufdiagramm

XAMPP X für alle beliebigen Betriebssysteme, Apache, MySQL, PHP und Perl  
 Programmpaket mir freier Software  
Apache Webserver-Software

GUI Graphical User Interface  
 Grafische Benutzeroberfläche

**Einleitung**

1. **Einleitung**

## Projektumfeld

Ich bin Auszubildender Fachinformatiker für Anwendungsentwicklung am SRH Berufsbildungswerk Neckargemünd GmbH. Das SRH Berufsbildungswerk Neckargemünd GmbH bildet über 900 Auszubildende in über 40 anerkannten Ausbildungsberufen aus und bietet ihnen somit die Chance, auf die Integration in den ersten Arbeitsmarkt.

Im Rahmen meines betrieblichen Abschlussprojektes bin ich Auftragnehmer einer Förderschule, die ein „Schiffe versenken“-Spiel entwickelt haben möchte.

Der Auftraggeber ist eine Förderschule mit Förderschwerpunkt körperliche und motorische Entwicklung in Heidenheim. Die Schule bietet Kindern die Möglichkeit an, einen Hauptschulabschluss zu erwerben. Diese Schule hat ca. 200 Schüler insgesamt, unterrichtet grundsätzlich aber in kleinen Klassengrößen, um besser auf die individuellen Bedürfnisse der Schüler eingehen zu können. Das Ziel ist es, die Schüler in ihrer Entwicklung in vollen Umfang zu unterstützen und sie somit auf das spätere Leben trotz motorischen Einschränkungen vorzubereiten.

Der Kontakt ist durch einen Freund meines Vaters entstanden, der Lehrer dieser Schule ist. Er hat mich bei der Schulleitung empfohlen. Diese hatte das Vorhaben bezüglich des Projekts „Schiffe versenken“ danach bei mir in Auftrag gegeben.

Das Projekt soll in Zukunft im EDV-Unterricht zum Einsatz kommen, um mit motorisch beeinträchtigten Personen an die Handkoordination anhand der Mausbewegung zu sensibilisieren. Die Demoversion wird auf einen lokalliegenden Webserver erstellt und getestet, bevor es dort in Zukunft regulär zum Einsatz kommt.

Das Projekt wird im Rahmen der betrieblichen Projektarbeit in der SRH Neckargemünd erstellt.

## Projektziel

Die Förderschule Ojde-Schule hat mich beauftragt, ein „Schiffe versenken“-Spiel zu entwickeln, um Kindern mit motorischen Einschränkungen den Umgang mit der Computer-Maus zu trainieren.

Das Projekt soll webbasiert als Demoversion zunächst auf einen lokalliegenden Server installiert werden. Der Grund für die webbasierte Entwicklung ist der, dass auf den Schüler-PCs keine zusätzliche Software dafür installiert werden soll. Die Schüler-PCs sind alle standardmäßig mit einem Browser ausgestattet. Somit ist die webbasierte Umsetzung am besten geeignet. Diese Demoversion ist zu Testzwecken auf 2 Spieler im Netzwerk beschränkt und kann in Zukunft erweitert werden, um mehrere Matches gleichzeitig im Netzwerk zu ermöglichen.

Es wird eine HTML-Datei erstellt, die mit JavaScript interaktiv bedienbar wird. Auf dem Server wird eine Textdatei mit Daten der Positionen der Schiffe und Schüsse abgelegt. Diese Informationen werden mit PHP über Ajax ausgelesen und im HTML dynamisch mit JavaScript und CSS grafisch dargestellt.

Es wurden Features und Regeln festgelegt, die im Programm umgesetzt werden sollen.

Folgende Features sollen implementiert werden:

* Spieler gegen Spieler
* Online über einen Webserver nutzbar
* Statistik für Züge, Trefferquoten und verlorene Schiffe

Es wurden folgende Regeln definiert:

1. Die Spielfeldgröße beträgt 10 x 10 Felder.
2. Der anzufangende Spieler soll zufällig ausgewählt werden.
3. Die Schiffe dürfen nicht aneinanderstoßen.
4. Die Schiffe dürfen nicht über Eck gebaut sein oder Ausbuchtungen besitzen.
5. Die Schiffe dürfen auch am Rand des Spielfelds liegen.
6. Die Schiffe dürfen nicht diagonal aufgestellt werden.
7. Jeder Spieler verfügt über insgesamt zehn Schiffe (in Klammern die Größe):

* ein Schlachtschiff (5 Felder)
* zwei Kreuzer (je 4 Felder)
* drei Zerstörer (je 3 Felder)
* vier U-Boote (je 2 Felder)

Mit dem Spiel „Schiffe versenken“ soll eine Verbesserung der motorischen Fähigkeiten der Schüler auf lange Sicht erzielt werden. Hierbei wird der Fokus auf die Bedienung mit der Computer Maus gelegt.

## Projektbegründung

Wie in 3.2.1 Make-or-Buy-Entscheidung erklärt, handelt es sich hierbei um Individualsoftware und konnte nicht für den Kunden passend gekauft werden. Trotz langen Recherchen wurde kein geeignetes Produkt für den Kunden gefunden. Somit musste das Produkt kundengerecht entwickelt werden.

## Projektschnittstellen

Damit die Daten der Spieler, wie Spielernamen, Aufstellungen der Schiffe und Spielzüge, in die Textdatei geschrieben werden können, ist es nötig, mit JS die Informationen aus dem HTML‑Dokument zu lesen. Diese werden dann über Ajax an einen PHP‑Skript gesendet, die die erhaltenen Daten dann in eine Textdatei als JSON‑String schreibt. Dies ist wichtig, da die Spieler auf die Daten entsprechend zugreifen müssen und diese Daten, wie gemachte Spielzüge des Gegners, ebenfalls wieder in das HTML‑Dokument eingetragen werden müssen.

1. **Projektplanung**

## Projektphasen

Hier befindet sich die Tabelle für die grobe Zeitplanung, wie auch in dem Projektantrag zu finden. Eine detaillierte Ablaufplanung befindet sich im Anhang.

Tabelle 1: Grobe Zeitplanung

|  |  |
| --- | --- |
| **Projektphasen** | **geplante Zeit** |
| Kundengespräch | 1 h |
| Projektplanung | 4 h |
| GUI‑Entwurf | 4 h |
| Erstellung der GUI | 6 h |
| Implementierung der Features | 12 h |
| Implementierung der Regeln | 16 h |
| Funktionsprüfung & Qualitätskontrolle | 4 h |
| Soll‑Ist‑Vergleich | 1 h |
| Abnahme | 2 h |
| Dokumentation | 30 h |
| Summe | 80 h |

## Ressourcenplanung

Im Rahmen des Projektes wurden sowohl personelle als auch sachliche Ressourcen benötigt. Zum einen eine Arbeitskraft, die die 80 Stunden Arbeit leistet. Zum anderen die Arbeitsplatzausstattung, die diese Arbeitskraft benötigt. Dazu zählen zwei Monitore, ein PC inklusive Verkabelung, eine Tastatur und Maus, ein Bürostuhl, ein Computerschreibtisch. Zusätzlich wurde folgende Software zur Entwicklung genutzt: XAMPP als Softwarepaket mit Apache, Visual Studio Code, Webbrowser Mozilla Firefox & Google Chrome.

## Entwicklungsprozess

Das Projekt wurde nach der Wasserfallmethode entwickelt. Die Vorteile darin liegen in der einfachen Umsetzung für die Demoversion mit einem klaren Ergebnisvorschlag, das dem Kunden präsentiert werden soll. Dadurch soll der Kunde einen konkreten Eindruck des Ergebnisses für die Vollversion bekommen und gewünschte Änderungen beziehungsweise Erweiterungen mitteilen. Somit erfolgt nach der Einführung der Demoversion eine stärkere Einbeziehung der Ojde-Schule.

1. **Analysephase**

## Ist-Analyse

Wie schon in 1.1 Projektumfeld erwähnt, ist die Ojde‑Schule eine Förderschule für Schüler mit motorischen Einschränkungen. Um die Schüler bei der Entwicklung zu unterstützen, malten die Schüler in Kunst Mandalas mit Farben nach Wunsch aus. Dies sollte ihre Koordination mit ihren Händen verbessern, um sie auf das spätere Leben trotz motorischen Einschränkungen vorzubereiten.

Durch die Entwicklung des Spiels „Schiffe versenken“ soll für das Unterrichtsfach „EDV“ eine weitere Möglichkeit der Förderung entstehen, die gezielt auf dem Umgang mit Desktop‑PCs angepasst ist. Hierbei soll für die Schüler gezielt die Koordination der Mausbewegungen verbessert werden, die später im Leben beziehungsweise im Beruf benötigt werden könnten.

Die Ojde-Schule betreibt einen lokalen Webserver, mit dem alle in der Schule befindlichen Windows 10 PCs beziehungsweise Laptops mit Mozilla Firefox und Google Chrome zugreifen können. Auf dem Schulserver ist PHP bereits installiert und dieser wird vom IT‑Lehrer verwaltet.

## Wirtschaftlichkeitsanalyse

Im Folgendem wird der wirtschaftliche Aspekt des Projektes betrachtet.



### Make or Buy-Entscheidung

Das Projekt soll kundengerecht für die Ojde-Schule entwickelt werden. Dabei wird Zugriff auf den Quellcode benötigt, um bei Bedarf gewünschte Anpassungen vornehmen zu können. Somit ist ein Kauf eines Standardproduktes nicht geeignet und muss individuell angepasst entwickelt werden.

### Projektkosten

Die Kosten der Durchführung des Projektes setzen sich aus Ressourcenkosten und Personalkosten zusammen. Der Stundensatz für Personal wird auf 70 € angesetzt. Die Sachmittelkosten betragen 40 € für Hardware und Software für den Zeitraum der Durchführung des Projektes. Sowohl die Kosten für Hard- und Software, als auch der Stundensatz wurden von der Abteilung Finanzcontrolling angegeben.

Berechnung der Gesamtpersonalkosten:

70 € / Stunde \* 80 Stunden = 5.600 €  
Somit belaufen sich die Gesamtkosten für das Projekt auf 5.640 €.

### Amortisationsdauer

Eine Amortisierung ist nicht berechenbar. Es wird lediglich durch die Entwicklung der Demoversion ein Teil der Vorarbeit geleistet. Dies hat zur Folge, dass die spätere Arbeitszeit für das finale Produkt eingespart wird.

## Nutzwertanalyse

Der nicht monetäre Nutzen ist schwierig in Zahlen zu messen. Dieser ist nämlich, einerseits für die Schüler, die durch das Projekt besser mit dem Umgang der Computer Mause werden sollen. Dies hilft ihnen, da ihre eingeschränkten motorischen Fähigkeiten damit zusätzlich trainiert werden. Andererseits profitiert auch die Gesellschaft davon, wenn die Schüler in ihrem späteren Leben besser mit ihrer Einschränkung umgehen können. Diese sind dann im Alltag auf weniger Unterstützung angewiesen.

## Anwendungsfälle

Das Projekt soll im Schulunterricht der Ojde-Schule im Fach „EDV“ begleitend zum Einsatz kommen. Die Schüler werden bei der Benutzung des Spiels hauptsächlich mit der Maus interagieren. Lediglich der Spielernamen muss über die Tastatur eingegeben werden. Die Schiffe müssen durch Mausklick auf die Startfläche und die Endfläche platziert werden, worauf diese sich anschließend durch einen Klick auf die Schaltfläche „Spiel beginnen“ für ein Spielmatch eintragen lassen können. Sobald der zweite Spieler sich registriert hat, geht der Match los und die Spieler können abwechselnd durch einen Klick auf dem gegnerischen Spielfeld die Felder des Gegners beschießen. Nach jedem getätigten Spielzug wird überprüft, ob ein Schiff getroffen oder sogar versenkt wurde. Falls alle Schiffe eines Spielers versenkt wurden ist das Spiel zu Ende und eine Statistik wird für die Spieler ausgegeben. (siehe Abbildung 3: PAP des Spielvorgangs)

## Qualitätsanforderungen

Der Fokus für Qualität wurde auf die Funktionsfähigkeit der Demoversion gelegt. Diese soll als Vorlage zur späteren Weiterentwicklung für die Vollversion dazu beitragen, die grundlegende Funktionsweise mit dem Spielablauf der Ojde-Schule ein Prototyp zu demonstrieren, der durch Kundenwünsche noch angepasst werden kann.

## Fachkonzept

Die Schüler der Ojde-Schule benötigen eine zusätzliche Möglichkeit, ihre eingeschränkten motorischen Fähigkeiten zu trainieren. Hierbei wurde sich für eine Möglichkeit im Schulfach „EDV“ entschieden, bei der die Schüler mit dem Umgang der Computermaus zusätzlich trainiert werden sollen. Dafür wurde sich entschieden, jemanden zu beauftragen, ein „Schiffe versenken“-Spiel zu entwickeln, dass für die Schüler ausschließlich mit der Maus zu bedienen ist.

Hierbei sollen die Spieler zunächst ihre Schiffe platzieren, die jedoch nicht aneinanderstoßen sollen. Danach sollen zwei Schüler gegeneinander Spielen und abwechselnd ihre Spielzüge tätigen. Am Ende des Spieles soll eine Statistik bezüglich Trefferquote ausgegeben werden. Dies soll die Schüler motivieren, sich zu steigern und den Wiederspielwert erhöhen.

1. **Entwurfsphase**

## Zielplattform

Wie in 3.1 IST‑Analyse beschrieben, betreibt die Schule einen lokalen Webserver, auf dem alle im Schulnetz befindenden Geräte mit einem Browser zugreifen können. Des Weiteren ist es erwünscht zusätzliche Installationen auf den PCs beziehungsweise Laptops zu vermeiden, wenn möglich. Somit bietet sich eine webbasierte Entwicklung an, damit die Schüler der Ojde-Schule ohne zusätzliche Software einfach über den Browser, wie Mozilla Firefox oder Google Crome, auf das Spiel zugreifen können.

Um zusätzliche Installationen zu vermeiden wird somit HTML, CSS zur Gestaltung der Spielseite eingesetzt, die durch JavaScript interaktiv bedienbar wird und über Ajax ein PHP Skript aufruft, das die Verwaltung der benötigten Daten während eines Spiels in eine Textdatei übernimmt.

## Architekturdesign

Das Spiel „Schiffe versenken“ läuft linear ab. Zunächst müssen die Spieler ihre Schiffe platzieren, sich dann in eine Textdatei registrieren lassen und können anschließend abwechselnd ihre Spielzüge tätigen. Aus diesem Grund wurde sich für eine funktionell orientierte Programmierung entschieden, jedoch mit Beihilfe der JSON Klasse Spiel, die nur als Vorlage dient, die dann durch eine Funktion als JSON-String in die Textdatei geschrieben wird. Sie speichert die nötigten Daten, die während eines Spielmatches regelmäßig benötigt werden.

## Entwurf der Weboberfläche

Zuerst wurde eine Skizze für die webbasierte Benutzeroberfläche erstellt. (siehe Anhang Abbildung 1) Darauf wurde geachtet, dass sie vollständig mit der Maus bedienbar gestaltet wird. Diese beinhaltet sowohl Textfelder für Spielernamen, als auch Spielfeldern für die jeweiligen Spieler. Ebenfalls wurde eine Schiffanzeige, sowie ein Bereich für Statusmeldungen vorgesehen.

## Datenmodell

Zur Datenspeicherung der Demoversion wurde sich aufgrund der leichten Handhabung für eine Textdatei entschieden. Diese soll für die einfachere Verarbeitung Informationen als JSON-String abspeichern. Die Textdatei soll als JSON-String Daten bezüglich des laufenden Spiels speichern.

Dabei werden sowohl die Spielernamen und die Aufstellungen beider Spieler des aktuellen Spiels benötigt. Zusätzlich muss die Information, welcher Spieler den Spielzug hat, mit abgelegt werden. Der Grund dafür ist, dass die Spieler den gegnerischen Spielzug bei sich eintragen muss. Dieser wird als „Token“ bezeichnet.

Die Textdatei wird unformatiert in einer Zeile abgespeichert. Ein formatierter Aufbau der JSON‑Datei zur besseren Übersicht befindet sich im Anhang. Dieser Zustand befindet sich

## Geschäftslogik

Der Ablauf des Spiels wurde linear aufgebaut, da die Bedienung ebenfalls linear erfolgt. Zunächst platzieren die Spieler ihre Schiffe. Hier wird durch die Funktion *feldauswahl* eine grundlegende Überprüfung der Gültigkeit der Auswahl der Felder vorgenommen. Es wird überprüft ob das Schiff gerade platziert werden soll und welche Länge das Schiff bekommt. Wenn die Überprüfung positiv ausfällt, wird die Länge, der Anfang des Schiffes und die Ausrichtung an die Funktion *platziereSchiff* mitübergeben. Diese prüft, ob das Schiff in der Größe vorhanden ist und ob auf einem Feld des Schiffes ein Hindernis liegt, wie etwa ein Sperrfeld oder anderes Schiff. Ist dies nicht der Fall, wird das Schiff platziert mit einem Sperrfeld um das Schiff rum. Wurden alle Schiffe platziert und der Spielername eingegeben, können die Spieler sich durch den Klick auf die Schaltfläche „Spiel beginnen“ in ein Spiel eintragen lassen.

Wenn Spieler 1 sich erfolgreich einträgt, wird ein Intervall gestartet, der alle 5 Sekunden die Funktion *serverAnfrage* aufruft. Diese fragt regelmäßig ein PHP-Skript ab, ob ein 2. Spieler sich eingetragen hat. Für eine Veranschaulichung wurde hierfür ein Programmablaufplan erstellt (Abbildung 2: PAP des Registrierungsvorgangs zum Spiel), das sich im Anhang befindet.

Sobald Spieler 1 eingetragen im Token ist, wird er durch die Serveranfrage erkannt, worauf dann eine neue Serveranfrage mit Intervall gestartet wird, die überprüft, ob der Spieler im Token eingetragen ist. Ist dies der Fall, kann der Spieler 1 den Spielzug tätigen. Ist Spieler 2 eingetragen, muss auf den gegnerischen Spielzug gewartet werden. Ist Spieler 1 dann im Token eingetragen, wird überprüft, ob ein Spielzug getätigt wurde. Bei einem Klick auf das gegnerische Spielfeld wird die Funktion *spielzug* ausgeführt, die dann prüft, ob das Feld bereits beschossen wurde und der Spieler gerade dran ist. Hier wird eine neue Serveranfrage gesendet, um den Spielzug durch das PHP-Skript in die Textdatei einzutragen. Jeder Spielzug wird ausgewertet und es wird überprüft, ob ein Schiff getroffen oder versenkt wurde. Wurden alle Schiffe versenkt ist das Spiel zu Ende und die Statistik wird durch die Funktion *statistikausgabe* unter der Schiffanzeige ausgegeben. Auch hierfür wurde ein sich im Anhang befindender Programmablaufplan erstellt. (Abbildung 3: PAP des Spielvorgangs)

## Maßnahmen zur Qualitätssicherung

Während der Entwicklung wurde nach jeder erstellten Funktion ein Unittest durchgeführt, um die ordnungsmäßige Lauffähigkeit sicherzustellen. Des Weiteren wurde nach Fertigstellung des Spiels ein White-Box Test zur ganzheitlichen Funktionsüberprüfung durchgeführt. Abschließend wurde noch ein praktischer Blackbox-Test durchgeführt, um unerwartete Fehlerquellen auszuschließen. Die Demoversion soll jedoch nach der Abnahme vor Ort ebenfalls durch die Ojde-Schule getestet werden, um Feedback zur Vollversion zu erhalten.

## Pflichtenheft

Erstellen… Auszug des Pflichtenheft im Anhang…

* + - Auszüge aus dem Pflichtenheft/Datenverarbeitungskonzept, wenn es im Rahmen des Projekts erstellt wurde.

**Beispiel**

Ein Beispiel für das auf dem Lastenheft (siehe Kapitel [3.6](#_bookmark27)) aufbauende Pflichtenheft ist im Anhang [A4](#_bookmark57) zu finden.

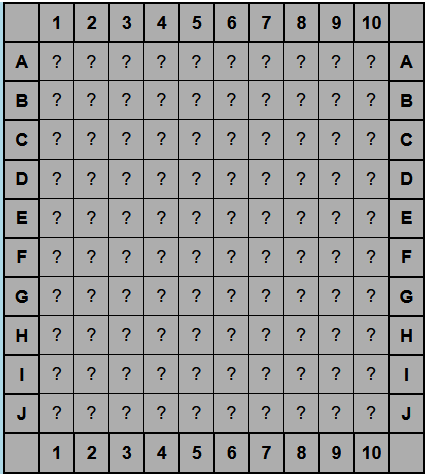
1. **Implementierungsphase**

## Implementierung der Datenstrukturen

Das in wie 4.4 Datenmodell geplante Modell wurde durch eine benötigte Information erweitert. Bei der Entwicklung ist aufgefallen, dass die Spieler ebenso die Information bei einem gemachten Spielzug benötigen, ob das gegnerische Schiff getroffen wurde oder sogar versenkt wurde. Somit wurde die Information des letzten gemachten Spielzuges im Token miteingetragen. Dazu wurde zur Abgrenzung noch ein Trennzeichen eingefügt, damit die Daten des Tokens durch eine Funktion getrennt werden.

## Implementierung der Weboberfläche

Es wurde für die Spielfelder des Spielers und des Gegners in HTML jeweils eine Tabelle erstellt, die mit CSS grafisch dargestellt wird. (siehe Abbildungen unten)

Links ist das eigene Spielfeld zu sehen, deren belegbaren Felder mit Wasser „W“ vorbelegt sind. Rechts befindet sich das noch unbekannte gegnerische Spielfeld, deren noch unbekannten Felder mit „?“ vorbelegt sind.

Jede ansprechbare Zelle wurde mit einer ID zugewiesen, die dem Aufbau eines Koordinatensystems folgt. Sie setzt sich auf Zeile als erste Ziffer und Spalte als folgende Zahl zusammen. Der Ursprung ist das Feld „A1“, wobei das „a“ die Zeile beschriftet und die „1“ die Spalte. Die Tabellen der Spielfelder sind quadratisch bestehend aus 10x10 Feldern. Somit reicht die Beschriftung bis „J10“.

Wenn der Spieler mit dem Mauszeiger über eine Wasserfläche fährt, wird der Hintergrund dieser Fläche schwarz dargestellt. Dies dient zur Hilfe für die Orientierung, um eine genaue Auswahl der Spielfläche zu ermöglichen. Des Weiteren wurde die Feldgröße dynamisch auf die Fenstergröße angepasst. Bei größeren Bildschirmen ergeben sich beim Vollbild des Browserfensters somit größere Spielfelder. Ein größeres Feld kann unter Umständen auch von Vorteil für die motorisch eingeschränkten Schüler sein, da durch eine größere Fläche bei der Genauigkeit eine größere Toleranz besteht.

## Implementierung der Spiellogik

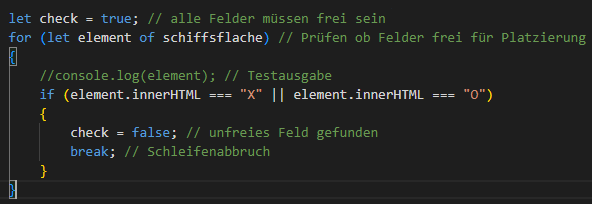


Folgend wird die Umsetzung der Spiellogik beschrieben.

### Platzierung der Schiffe

Nach dem das Design der Oberfläche der Webseite für das Spiel „Schiffe versenken“ fertiggestellt wurde, wurde zunächst die Platzierung der Schiffe ermöglicht. Dazu wurden die Aktionen des Spielers gespeichert, durch die ein Spieler Start- und Endkoordinaten des eigenen Spielfeldes des Schiffes durch einen Klick auf das gewünschte Spielfeld auswählen kann. Dabei wird einerseits überprüft, ob das Schiff gerade platziert wurde und andererseits wird die Schifflänge auf Gültigkeit überprüft. Diese muss einschließlich von 2 bis 5 reichen, da andere Längen für Schiffe laut Spielregeln nicht vorgesehen sind. (siehe Quellcodedokumentation von Ajax.js Zeile 104 – 124)

Hierbei werden die für die Platzierung benötigten Spielfeldflächen zunächst in eine Liste geschrieben. Es werden zunächst alle in der Liste enthaltenden Flächen überprüft, ob diese frei zur Platzierung des Schiffes sind. Dies ist nur der Fall, wenn keines der Flächen bereits durch ein Schiff belegt wurde oder es sich auch nicht um ein Sperrfeld handelt. (siehe Ausschnitt aus Quellcodedokumentation unten)

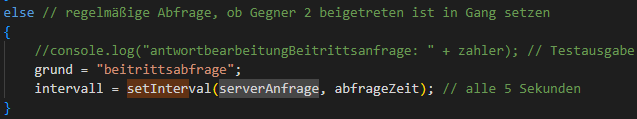


Sollte ein Feld des Schiffes doch nicht frei zur Platzierung eines Schiffes sein, wird eine Meldung an den Spieler ausgegeben, dass ihn diesbezüglich darauf hinweist. Eine Sperrfläche wird direkt um das Schiff nach der Platzierung erstellt, um zu verhindert, dass Schiffe aneinander platziert werden können. Dies dient zur Einhaltung der gewünschten Regeln für das Spiel.



### Anmeldung der Spieler

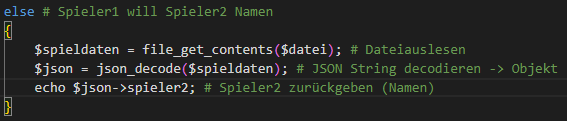
Nachdem alle Schiffe platziert wurden, können Spieler nach Eingabe des Spielernamens sich in ein Spiel eintragen lassen. Hierbei ist das Vorhandensein eines „;“ im Namen unzulässig, da die Informationen des Tokens durch dieses Zeichen getrennt werden. Dieser Eintrag erfolgt in eine Textdatei als JSON‑String. Beim Klick auf die Schaltfläche „Spiel beginnen“ wird der Spielername und die Aufstellung der Schiffe gespeichert in einem Array als JSON-String an PHP gesendet. Der erste Spieler, der sich einträgt, prüft nach einem erfolgreichen Eintrag in eine Textdatei regelmäßig, ob ein zweiter Spieler beigetreten ist. Hierbei wird der Grund der Serveranfrage auf „Beitrittsabfrage“ gesetzt und im Intervall von 5 Sekunden abgefragt. (siehe Ausschnitt aus Quellcodedokumentation unten)



Zusätzlich bekommt der erste Spieler, der sich einträgt, die ID „Spieler1“ übergeben, die nach Spielbeginn mit jedem getätigten Spielzug mitgesendet wird. Somit wird auch das Problem umgangen, dass beide Spieler mit denselben Spielernamen nicht gegeneinander spielen könnten.

Der zweite Spieler erhält nach erfolgreichem Eintrag den Namen des ersten Spielers und die Information ob er den ersten Spielzug hat. Dies wurde durch eine einfache Funktion zur Generierung einer Zufallszahl umgesetzt und wird dann als Token in die Textdatei miteingetragen. Der Token speichert sowohl Spieler-ID von dem Spieler, der gerade den Spielzug hat, als auch den zuletzt durchgeführten Spielzug. Dieser ist bei Spielbeginn nicht vorhanden und dient dazu, bei Tokenwechsel für den anderen Spieler die Information des gegnerischen Spielzuges beim eigenen Spielfeld einzutragen.

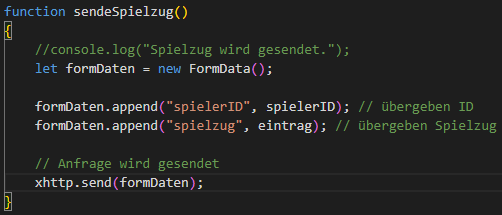
Nach dem Beitritt des zweiten Spielers wird dieser auch vom ersten Spieler durch die erwähnte PHP‑Abfrage nach Intervall der Textdatei entdeckt, wodurch dieser auch den Spielernamen seines Gegners erhält. (siehe Ausschnitt aus Quellcodedokumentation unten)



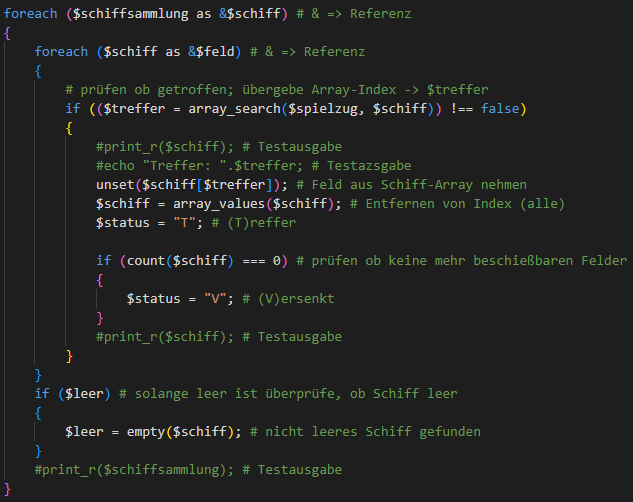
Danach überprüft er regelmäßig ob er mit dem Spielzug dran ist. Dies geschieht durch die Abfrage des Tokens in der Textdatei.

### Spielzüge

Je nachdem, welcher Spieler beim ersten Spielzug im Token eingetragen wurde, erhält dieser die Möglichkeit für den Spielzug. Beim ersten Spielzug enthält der Token noch keine Information zum letzten Spielzug, da noch kein Spielzug getätigt wurde. Der Spieler mit dem Spielzug kann dann auf ein Feld des gegnerischen Spielfelds klicken, das noch nicht beschossen wurde. Andererseits kommt eine Meldung an den Spieler, dass dieses Feld bereits beschossen wurde. Nach der Auswahl des Feldes im gegnerischen Spielfeld wird die ID des Feldes mit der Spieler ID durch eine Serverabfrage mit Grund „Spielzug“ an PHP gesendet, um diese einzutragen. (siehe Ausschnitt aus Quellcodedokumentation unten)



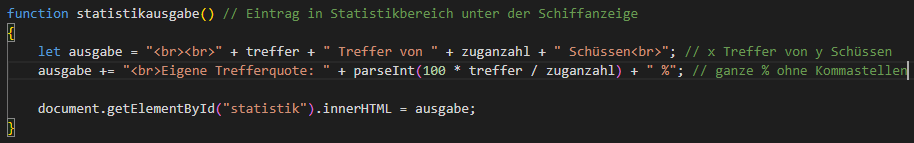
Nach dem Eintrag kommt die Rückmeldung vom PHP‑Skript mit der Information, ob es sich um einen Treffer handelt oder sogar Versenkt. Falls es sich beim Versenken um das letzte gegnerische Schiff handelt, kommt die Information, dass der Spieler gewonnen hat. Dies wird durch das PHP‑Skript festgestellt, in dem der Status in einer erstellten Funktion durch die ganze Schiffaufstellung jedes Schiff überprüft, ob ein Treffer vorliegt. (siehe Ausschnitt aus Quellcodedokumentation unten)



Nach dem Spielzug wechselt der Token und beinhaltet die neue Information des letzten Spielzuges. Dieser wird durch die regelmäßige Serveranfrage gelesen. Wird eine Änderung des Spielers festgestellt, wird der Spielzug des Gegners eingetragen, und es wird eine Statusmeldung für den Spieler ausgegeben, nach dem die Information des letzten Spielzuges verarbeitet wurde.

### Statistik

Für die Umsetzung der Statistik wurden zusätzliche Variablen benötigt, die die Spielzüge und Treffer abspeichern, woraus dann die Berechnung der Trefferquote in Prozent erfolgt. Die Ausgabe der Statistik erfolgt unter der Schiffanzeige am Ende eines Spielmatches für beide Spieler. Dies wurde in eine Funktion ausgelagert. (siehe Ausschnitt aus Quellcodedokumentation unten)



1. **Abnahmephase**

Die Abnahme erfolgte bei der Ojde-Schule vor Ort. Die Dateien inklusive der Benutzerdokumentation wurden mit einem USB-Stick der Schulleitung übergeben. Zusammen mit dem EDV-Lehrer wurde das Produkt auf den Schulserver kopiert, um in der Schule noch einen Funktionstest für die Demoversion vor der Schulleitung zu demonstrieren. Die Schulleitung war mit der Demoversion zufrieden. Die Ojde-Schule möchte diese mit ausgewählten Schülern mit motorischen Einschränkungen selbst testen lassen und von den Endbenutzern noch Feedback einholen. Diese Rückmeldungen sollen ausgewertet werden und dann in der Vollversion berücksichtigt werden.

1. **Dokumentation**

Die Dokumentation des Projektes „Schiffe versenken“ besteht aus zwei Bestandteilen. die Projektdokumentation und die Benutzerdokumentation.

Die Benutzerdokumentation ist für die Schulleitung beziehungsweise den verwaltenden EDV-Lehrer geschrieben. Diese soll einerseits dazu dienen, das Produkt in Betrieb zu nehmen und andererseits, die Anwender die Bedienung zu erklären. Hierbei lag der Fokus auf eine gute Veranschaulichung der Bedienungsanweisungen und eine verständliche Erklärung für die Inbetriebnahme der Demoversion.

Für die Projektdokumentation wurde während der Projektdurchführung fortlaufend dokumentiert. Dabei wurde auf eine gute Struktur und leichte Verständlichkeit geachtet.

1. **Fazit**

Text

## Soll-/Ist-Vergleich

Wie sich herausgestellt hat, benötigte die Implementierung der Features, dass Spieler gegeneinander spielen können mehr Zeit als die Implementierung der Regeln. Der Aufwand für die Anmeldung der Spieler, die Verarbeitung der Spielzüge für den Spielvorgang und die Auswertung der Spielzüge war im Vergleich zu den Regeln für die Platzierung der Schiffe deutlich höher. (siehe Tabelle unten)

Tabelle 2: Zeitvergleich

|  |  |  |  |
| --- | --- | --- | --- |
| **Projektphasen** | **geplante Zeit** | **benötigte Zeit** | **Differenz** |
| Kundengespräch | 1 h | 1 h | 0 h |
| Projektplanung | 4 h | 4 h | 0 h |
| GUI‑Entwurf | 4 h | 4 h | 0 h |
| Erstellung der GUI | 6 h | 5 h | -1 h |
| Implementierung der Features | 12 h | 20 h | +8 h |
| Implementierung der Regeln | 16 h | 9 h | -7 h |
| Funktionsprüfung & Qualitätskontrolle | 4 h | 4 h | 0 h |
| Soll‑Ist‑Vergleich | 1 h | 1 h | 0 h |
| Abnahme | 2 h | 2 h | 0 h |
| Dokumentation | 30 h | 30 h | 0 h |
| Summe | 80 h | 80 h | 0 h |

Trotz dieser Fehleinschätzung wurden alle Features und alle Regeln umgesetzt und die Demoversion fristgerecht fertiggestellt und ist voll funktionsfähig. Eine detaillierte Tabelle befindet sich im Anhang. (Tabelle 4: Detaillierter Zeitvergleich)

## Gelerntes

Bei der Durchführung des Projektes habe ich mehr über Webentwicklung gelernt. Hierbei sind einige Fehler aufgetreten, die behoben werden mussten. Diese Fehler wurden generell recht schnell korrigiert und haben nur wenig Verzögerungen verursacht.

Zum einen ist im Koordinatensystem des Spielfelds bei der 10er Spalte das Problem aufgetreten, dass die 10er Felder bei der Erkennung der Länge hinter der 1er Spalte gelegen hatte. Dies hatte zur Folge, dass die Schifflänge falsch berechnet wurde und somit die Schiffe als zu lang angesehen wurden. Dies wurde gelöst, in dem die Spalte und Zeile voneinander getrennt verarbeitet wurden, um das Problem der einfachen Sortierung zu umgehen.

Des Weiteren wurden die Sperrflächen bei der Platzierung der Schiffe falsch erstellt. Dieser Fehler hatte den Grund, dass die Sperrflächen nicht fest auf die Schiffsfläche bezogen wurden, sondern voneinander abhängig waren. Dies war am Rand der Spielfläche zu erkennen, wo die Sperrfelder dann abgeknickt nach oben beziehungsweise unten im Gegensatz zum Spielfeldrand lagen. Damit die Sperrfelder vor und nach dem Schiff richtig platziert werden, musste diese vollständig abhängig der Schiffsfläche gemacht werden.

Ein weiterer Fehler war, dass eine ungewollte Rekursion erzeugt wurde. Aufgrund mangelnder Konzentration nach einem Feueralarm wurde in der *Beitrittsabfrage* eine weitere *Serveranfrage* ausgeführt. Dies war unerwünscht, da die *Serverabfrage* die *Beitrittsabfrage* aufruft und diese mehrfache Aufrufung dann zu einem JavaScript-Fehler führte, der auf eine Rekursion hinwies. Am nächsten Tag wurde das Problem jedoch schnell behoben, in dem die ursprünglich vorgesehene Logik, das Senden eines Forms an das PHP‑Skript, stattdessen in die Funktion der *Beitrittsabfrage* implementiert wurde.

Auch für die Planung von Projekten muss der Aufwand der einzelnen benötigten Schritte in Zukunft präziser geschätzt werden. Wie schon in 8.1 Soll-/Ist-Vergleich beschrieben, wurde der Aufwand für die Erstellung des Anmeldevorgangs und der Spielzüge unterschätzt.

## Ausblick

In der Demoversion wurden aufgrund der begrenzten Zeit das Design der Weboberfläche des Spiels einfach gehalten. Die Größe der Felder wurde zwar dynamisch gestaltet, jedoch

(max. halbe Seite)

* + - Wie wird sich das Projekt in Zukunft weiterentwickeln (z.B. geplante Erweiterungen)?

**Quellenverzeichnis**

1. Quellenverzeichnis

**Internetrecherchen**

https://www.w3schools.com/jsref/jsref\_includes.asp

https://www.w3schools.com/jsref/jsref\_includes\_array.asp

https://www.w3schools.com/html/html\_table\_borders.asp

https://www.w3schools.com/js/js\_timing.asp

https://www.w3schools.com/html/html\_tables.asp

https://www.w3schools.com/html/html\_css.asp

https://www.w3schools.com/xml/ajax\_xmlhttprequest\_response.asp

https://www.w3schools.com/php/php\_oop\_constructor.asp

https://www.w3schools.com/jsref/jsref\_split.asp

https://www.w3schools.com/jsref/jsref\_slice\_array.asp

https://www.w3schools.com/jsref/jsref\_charat.asp

https://www.w3schools.com/jsref/jsref\_parseint.asp

https://www.w3schools.com/js/js\_switch.asp

https://www.w3schools.com/jsref/jsref\_push.asp

https://www.w3schools.com/howto/howto\_css\_three\_columns.asp

https://www.w3schools.com/cssref/pr\_dim\_width.php

https://www.w3schools.com/howto/howto\_css\_aspect\_ratio.asp

https://www.w3schools.com/css/css\_dimension.asp

https://www.w3schools.com/jsref/jsref\_keys.asp

https://www.w3schools.com/js/js\_json\_arrays.asp

https://www.w3schools.com/js/js\_json.asp

https://www.w3schools.com/php/php\_oop\_classes\_objects.asp

https://www.w3schools.com/php/php\_json.asp

https://www.w3schools.com/php/func\_math\_rand.asp

https://www.w3schools.com/jsref/met\_win\_clearinterval.asp

https://www.w3schools.com/php/func\_var\_empty.asp

https://www.w3schools.com/php/func\_string\_explode.asp

https://www.w3schools.com/php/php\_looping\_foreach.asp

https://www.w3schools.com/php/func\_array\_replace.asp

https://www.w3schools.com/php/func\_array\_filter.asp

https://www.w3schools.com/jsref/jsref\_foreach.asp

https://www.w3schools.com/jsref/jsref\_filter.asp

https://www.w3schools.com/jsref/jsref\_obj\_array.asp

https://www.w3schools.com/js/js\_arrays.asp

https://www.php.net/manual/en/function.empty.php

https://www.php.net/manual/en/function.array-filter.php

https://www.php.net/manual/en/function.getrandmax.php

https://www.php.net/manual/en/function.rand.php

https://www.php.net/manual/en/functions.returning-values.php

https://www.php.net/manual/en/function.file-put-contents.php

https://www.php.net/manual/en/function.json-encode.php

https://www.php.net/manual/en/function.json-decode.php

https://www.php.net/manual/de/function.isset.php

https://www.php.net/manual/en/language.types.object.php

https://www.php.net/manual/en/function.json-encode.php

https://www.php.net/manual/en/function.json-decode.php

https://www.php.net/manual/en/function.file-exists.php

https://www.php.net/manual/en/language.oop5.magic.php

https://www.php.net/manual/en/language.oop5.visibility.php

https://www.php.net/manual/de/function.explode.php

https://www.php.net/manual/de/control-structures.foreach.php

https://www.php.net/manual/en/control-structures.for.php

https://www.php.net/manual/en/function.array-replace.php

https://www.geeksforgeeks.org/how-to-create-equal-width-table-cell-using-css/

https://www.geeksforgeeks.org/javascript-array-values-method/

https://linuxhint.com/javascript-remove-index-from-array/

https://linuxhint.com/enable-disable-input-fields-using-javascript/

https://stackoverflow.com/questions/72321480/how-to-loop-through-and-reference-with-an-array-in-an-array

https://stackoverflow.com/questions/52163894/javascript-reference-in-for-of-loop

https://stackoverflow.com/questions/17511273/how-to-replace-elements-in-array-with-elements-of-another-array

https://stackoverflow.com/questions/3954438/how-to-remove-item-from-array-by-value

https://stackoverflow.com/questions/41877868/php-a-way-to-get-property-without-getter

https://stackoverflow.com/questions/2156712/how-to-float-3-divs-side-by-side-using-css

https://stackoverflow.com/questions/58202591/three-divs-with-different-width-side-by-side- those-left-and-right-fixed

https://stackoverflow.com/questions/5445491/height-equal-to-dynamic-width-css-fluid-layout

https://stackoverflow.com/questions/21799852/css-width-same-as-height

https://stackoverflow.com/questions/11243075/css-scale-height-to-match-width-possibly-with-a-formfactor

https://stackoverflow.com/questions/22893866/css-dynamically-calculate-width

https://stackoverflow.com/questions/59696160/calc-pixel-height-dynamically-based-on-https://stackoverflow.com/questions/25990938/how-to-remove-keys-from-php-array

https://stackoverflow.com/questions/2295496/convert-array-to-json

https://stackoverflow.com/questions/10525744/css-table-cell-equal-width

https://stackoverflow.com/questions/10525744/css-table-cell-equal-width

https://stackoverflow.com/questions/7693224/how-do-i-right-align-div-elements

https://stackoverflow.com/questions/50936764/storing-key-value-pairs-in-an-array-in-javascript

https://stackoverflow.com/questions/1144705/best-way-to-store-a-key-value-array-in-javascript

https://stackoverflow.com/questions/42526032/how-to-find-if-element-with-specific-id-exists-or-not

https://stackoverflow.com/questions/9329446/loop-for-each-over-an-array-in-javascript

https://stackoverflow.com/questions/12504042/what-is-a-method-that-can-be-used-to-increment-letters

https://dev.to/dillionmegida/arraysplice-for-removing-replacing-or-adding-values-to-an-array-1k6c

https://www.tutorialrepublic.com/faq/how-to-delete-php-array-element-by-value-not-key.php

https://www.reddit.com/r/PHP/comments/kkrzks/getterssetters\_vs\_public\_properties/

https://www.beberlei.de/post/building\_an\_object\_model\_\_no\_setters\_allowed

https://php.budgegeria.de/frgcebc

https://coding-champ.com/tutorials/php/getters-and-setters

https://www.daniweb.com/programming/web-development/threads/100792/pass-a-2d-array-with-ajax

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Object

https://softwareengineering.stackexchange.com/questions/321534/what-json-structure-to-use-for-key-value-pairs

https://www.blogseite.com/html-div-nebeneinander-anordnen/

https://www.computerhilfen.de/info/css-tricks-3-divs-zentriert-nebeneinander-anzeigen-ohne-float.html

https://www.thesitewizard.com/css/make-table-cells-same-size.shtml

https://blog.hubspot.com/website/center-div-css

https://codedamn.com/news/frontend/use-css-to-put-div-side-by-side

dynamic-width-and-ratio-with-css

https://support.pega.com/question/how-calculate-dynamic-width-css-code-based-screen-resolution

https://mademyday.de/height-equals-width-with-pure-css/

https://www.cssportal.com/css-properties/size.php

https://www.helpster.de/tabulator-in-html-erzeugen-so-gelingt-s\_123496

https://www.lima-city.de/thread/html-text-mit-tabulator-abstand

https://www.home.unix-ag.org/juergen/selfhtml/absatz.html

https://sentry.io/answers/how-do-i-add-a-tab-space-instead-of-multiple-non-breaking-spaces/

https://forum.freecodecamp.org/t/hover-effect-not-working/592288

**Anhang**

1. Anhang

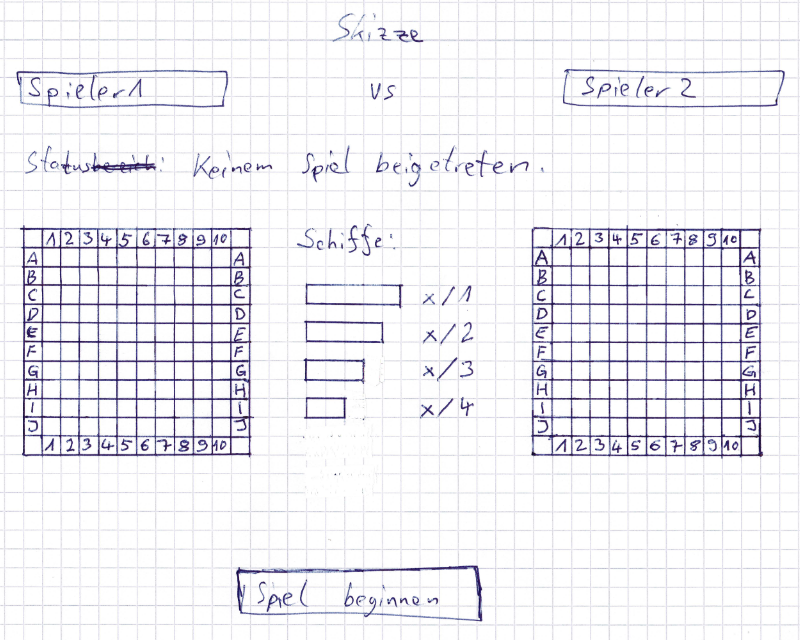
## Tabelle 3: Detaillierte Ablaufplanung

|  |  |  |
| --- | --- | --- |
| **Projektphase** | **geplante Zeit** | |
| Kundengespräch |  | 1 h |
| Projektplanung | 4 h |  |
| Analyse |  | 2 h |
| Erstellung der Programmablaufpläne |  | 2 h |
| GUI Entwurf |  | 4 h |
| Erstellung der GUI |  | 6 h |
| Implementierung der Features | 12 h |  |
| Registrierung der Spieler |  | 5 h |
| Verarbeitung der Spielzüge |  | 5 h |
| Ausgabe der Statistik |  | 2 h |
| Implementierung der Regeln | 16 h |  |
| Festlegung der Schiffanzahl |  | 2 h |
| gerade Platzierung der Schiffe |  | 5 h |
| Abstand der Schiffe |  | 7 h |
| Zufälliger Spielerstart |  | 2 h |
| Funktionsüberprüfung & Qualitätskontrolle |  | 4 h |
| Soll-Ist-Vergleich |  | 1 h |
| Abnahme |  | 2 h |
| Dokumentation |  | 30 h |
| Insgesamt |  | 80 h |

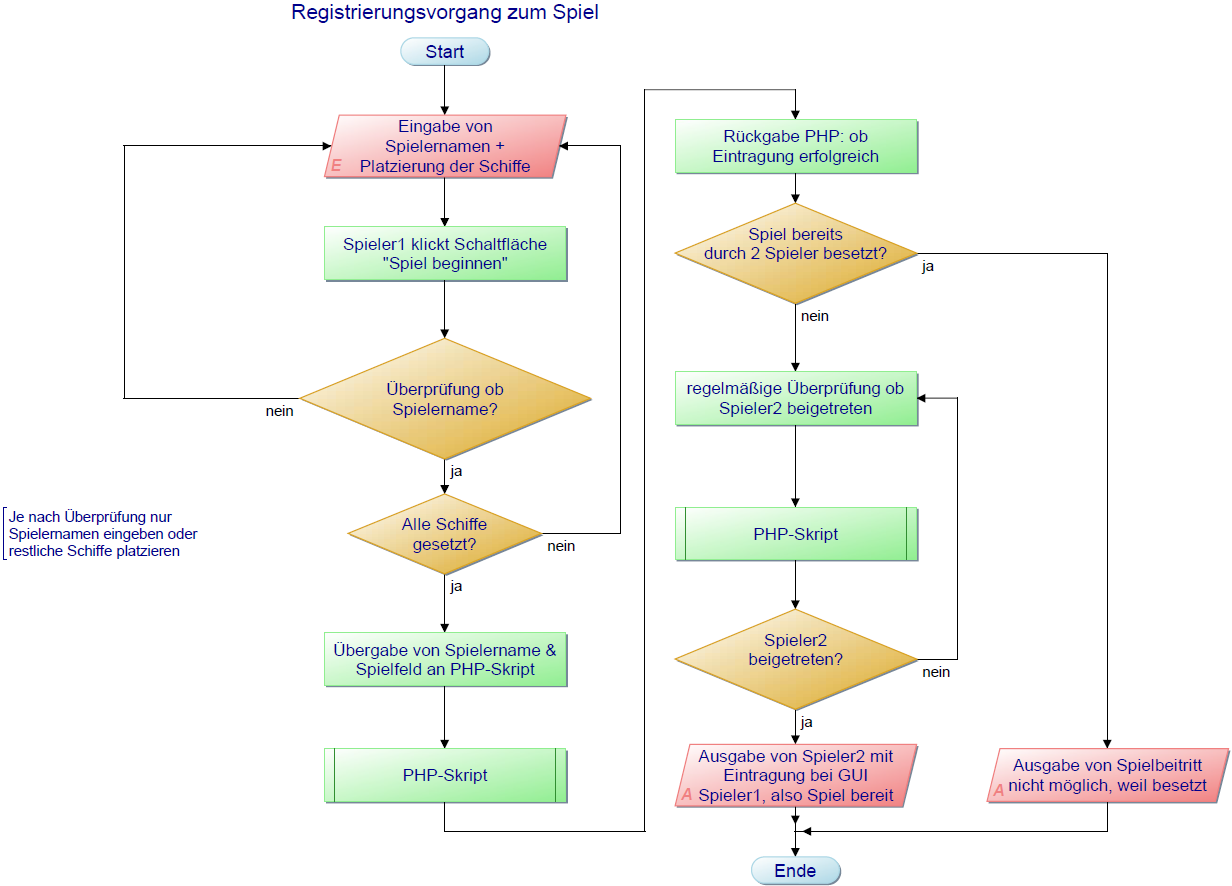
## Tabelle 4: Detaillierter Zeitvergleich

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Projektphase** | **geplante Zeit** | | **tatsächliche Zeit** | | **Differenz** |
| Kundengespräch |  | 1 h |  | 1 h | 0 h |
| Projektplanung | 4 h |  | 4 h |  |  |
| Analyse |  | 2 h |  | 2 h | 0 h |
| Erstellung der Programmablaufpläne |  | 2 h |  | 2 h | 0 h |
| GUI Entwurf |  | 4 h |  | 4 h | 0 h |
| Erstellung der GUI |  | 6 h |  | 5 h | -1 h |
| Implementierung der Features | 12 h |  | 20 h |  |  |
| Registrierung der Spieler |  | 5 h |  | 8 h | 3 h |
| Verarbeitung der Spielzüge |  | 5 h |  | 11 h | 6 h |
| Ausgabe der Statistik |  | 2 h |  | 1 h | -1 h |
| Implementierung der Regeln | 16 h |  | 9 h |  |  |
| Festlegung der Schiffanzahl |  | 2 h |  | 1 h | -1 h |
| gerade Platzierung der Schiffe |  | 5 h |  | 2 h | -3 h |
| Abstand der Schiffe |  | 7 h |  | 4 h | -3 h |
| Zufälliger Spielerstart |  | 2 h |  | 2 h | 0 h |
| Funktionsüberprüfung & Qualitätskontrolle |  | 4 h |  | 4 h | 0 h |
| Soll-Ist-Vergleich |  | 1 h |  | 1 h | 0 h |
| Abnahme |  | 2 h |  | 2 h | 0 h |
| Dokumentation |  | 30 h |  | 30 h | 0 h |
| Insgesamt |  | 80 h |  | 80 h | 0 h |

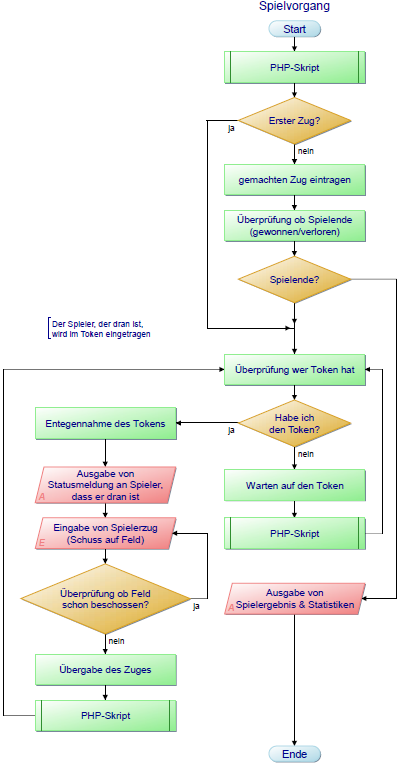
## Abbildung 1: Skizze der Webseite



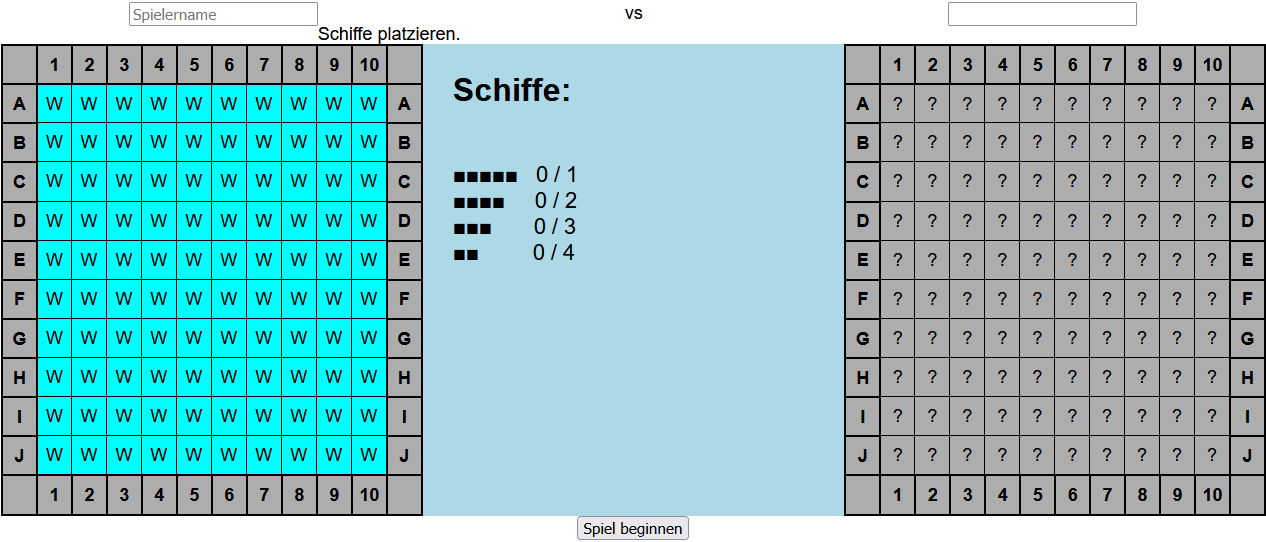
## Abbildung 2: PAP des Registrierungsvorgangs zum Spiel



## Abbildung 3: PAP des Spielvorgangs



## Abbildung 4: Oberflächenentwurf

****

## Pflichtenheft

**Zielbestimmung**

Zunächst muss die Webseite erstellt werden mit den benötigten Spielfeldern als Tabellen und den Namensfeldern für beide Spieler, sowie eine Schiffanzeige mit Statistiken und eine Befehlsschaltfläche zum Starten des Spiels. Danach muss durch eine Funktion die Platzierung der eigenen Schiffe unter Einhaltung von Regeln ermöglicht werden. Schiffe müssen gerade platziert werden, vertikal bzw. horizontal, und dürfen nicht aneinanderstoßen. Anschließend muss der Registrierungsvorgang implementiert werden, damit zwei Spieler gegeneinander antreten können. Dazu wird eine Funktion erstellt, die ein PHP Skript aufruft, die dann die benötigten Daten in eine Textdatei als JSON-String ablegt. Hierzu muss noch eine Hilfsklasse erstellt werden, die dann die benötigten Daten als Attribute abspeichert um aus dem generierten Objekt den JSON-String mit PHP generieren zu können. Darauffolgend muss die Verarbeitung der Spielzüge umgesetzt werden. Hierbei wird eine Funktion für die Verarbeitung und Auswertung erstellt. Zum Schluss muss eine Funktion zur Auswertung und Ausgabe der Statistik erstellt werden.

**Produkteinsatz**

Das Produkt soll zunächst als Demoversion im Informatikunterricht der Ojde-Schule zum Einsatz kommen. Die Anwender des Produktes sind motorisch eingeschränkte Schüler, die unter anderem mit Hilfe des Spiels lernen sollen, mit ihren Einschränkungen im späteren Leben zu Recht zu kommen. Das Produkt wird nach der Abnahme auf dem schuleigenen Webserver in Betrieb genommen, um diese vor Ort nochmal zu testen.

## JSON‑Datei

{

"spieler1":"Rudolf",

"spieler2":"Bambi",

"token":"Spieler1;",

"aufstellung1":[

["a1", "b1"],

["f1", "g1", "h1", "i1", "j1"],

["d1", "d2", "d3", "d4"],

["d6", "d7", "d8"],

["c10", "d10", "e10"],

["a4", "a5", "a6"],

["g4", "g5"],

["f7", "g7", "h7", "i7"],

["j9", "j10"],

["i4", "j4"]

],

"aufstellung2":[

["a10", "b10", "c10", "d10", "e10"],

["g10", "h10", "i10", "j10"],

["i1", "i2", "i3", "i4"],

["a1", "a2", "a3"],

["f5", "f6", "f7"],

["h7", "i7"],

["a7", "b7"],

["d2", "d3", "d4"],

["f2", "g2"],

["d7", "d8"]

]

}

## A8 Screenshots der Anwendung

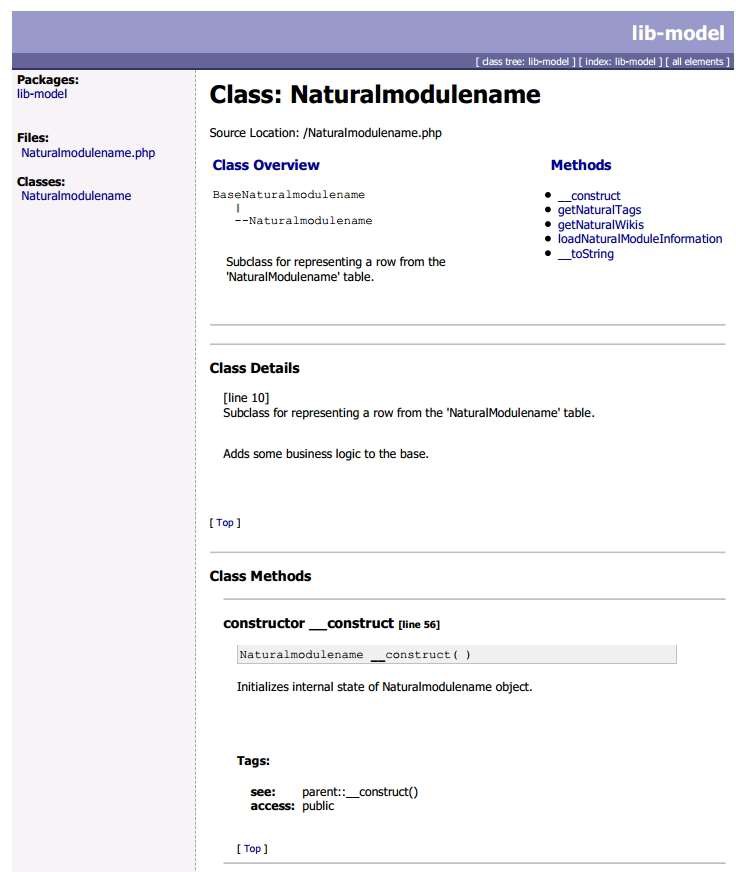


**Abbildung 7: Anzeige und Filterung der Module nach Tags**



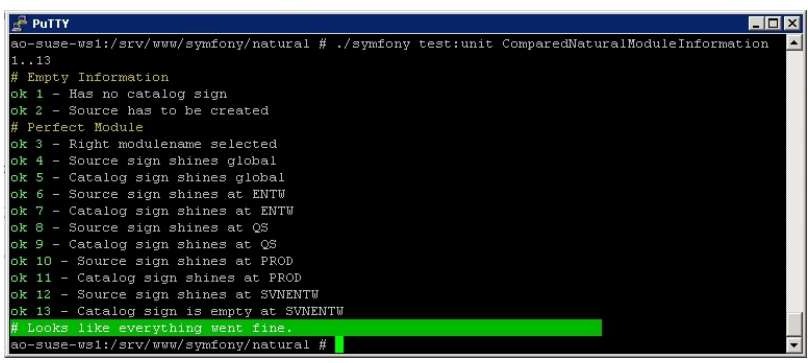
**Abbildung 8: Liste der Module mit Filtermöglichkeiten**

## A9 Entwicklerdokumentation (Auszug)



**Abbildung 9: Auszug aus der Entwicklerdokumentation mit *PHPDoc***

## A10 Testfall und sein Aufruf auf der Konsole



**Abbildung 10: Aufruf des Testfalls auf der Konsole**

$t->comment('Empty Information');

$emptyComparedInformation = new ComparedNaturalModuleInformation(array());

$t->is($emptyComparedInformation->getCatalogSign(), ComparedNaturalModuleInformation::EMPTY\_SIGN, 'Has no catalog sign');

$t->is($emptyComparedInformation->getSourceSign(), ComparedNaturalModuleInformation::SIGN\_CREATE, 'Source has to be created');

$t->comment('Perfect Module');

$criteria = new Criteria();

$criteria->add(NaturalmodulenamePeer::NAME, 'SMTAB');

$moduleName = NaturalmodulenamePeer::doSelectOne($criteria);

$t->is($moduleName->getName(), 'SMTAB', 'Right modulename selected');

$comparedInformation = $moduleName->loadNaturalModuleInformation();

$t->is($comparedInformation->getSourceSign(), ComparedNaturalModuleInformation::SIGN\_OK, 'Source sign shines global');

$t->is($comparedInformation->getCatalogSign(), ComparedNaturalModuleInformation::SIGN\_OK, 'Catalog sign shines global');

$infos = $comparedInformation->getNaturalModuleInformations(); foreach($infos as $info) {

$env = $info->getEnvironmentName();

$t->is($info->getSourceSign(), ComparedNaturalModuleInformation::SIGN\_OK, 'Source sign shines at ' . $env);

if($env != 'SVNENTW') {

$t->is($info->getCatalogSign(), ComparedNaturalModuleInformation::SIGN\_OK, 'Catalog sign shines at ' . $info->getEnvironmentName());

} else {

$t->is($info->getCatalogSign(), ComparedNaturalModuleInformation::EMPTY\_SIGN, 'Catalog sign is empty at ' . $info->getEnvironmentName());

}

}

**Listing 1: Testklasse**

## A11 Klasse: ComparedNaturalModuleInformation

Kommentare und simple Getter/Setter werden nicht gezeigt.

class ComparedNaturalModuleInformation { const EMPTY\_SIGN = 0;

...

const SIGN\_ERROR = 5;

private $naturalModuleInformations = array(); public static function environments() {

return array("ENTW", "SVNENTW", "QS", "PROD");

}

public static function signOrder() {

return array(self::SIGN\_ERROR, self::SIGN\_NEXT\_STEP, self::SIGN\_CREATE\_AND\_NEXT\_STEP, self::SIGN\_CREATE,

self::SIGN\_OK);

}

public function construct(array $naturalInformations) {

$this->allocateModulesToEnvironments($naturalInformations);

$this->allocateEmptyModulesToMissingEnvironments();

$this->determineSourceSignsForAllEnvironments();

private function allocateModulesToEnvironments(array

$naturalInformations) {

foreach ($naturalInformations as $naturalInformation) {

$env = $naturalInformation->getEnvironmentName(); if(in\_array($env, self::environments())) {

$this->naturalModuleInformations[array\_search($env, self::environments())] = $naturalInformation;

}

}

}

private function allocateEmptyModulesToMissingEnvironments() { if(array\_key\_exists(0, $this->naturalModuleInformations)) {

$this->naturalModuleInformations[0]-

>setSourceSign(self::SIGN\_OK);

}

for($i = 0;$i < count(self::environments());$i++) { if(!array\_key\_exists($i, $this-

>naturalModuleInformations)) {

$environments = self::environments();

$this->naturalModuleInformations[$i] = new EmptyNaturalModuleInformation($environments[$i]);

$this->naturalModuleInformations[$i]-

>setSourceSign(self::SIGN\_CREATE);

}

}

}

private function containsSourceSign($sign) {

foreach($this->naturalModuleInformations as $information) { if($information->getSourceSign() == $sign) {

return true;

}

}

return false;

private function containsCatalogSign($sign) {

foreach($this->naturalModuleInformations as $information) { if($information->getCatalogSign() == $sign) {

return true;

}

}

return false;

}

}

**Listing 2: Klasse ComparedNaturalModuleInformation**