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

Identnummer: 1094000

Ausbildungsbetrieb:

SRH Berufsbildungswerk Neckargemünd GmbH

Im Spitzerfeld 25

69151 Neckargemünd

INHALTSVERZEICHNIS

Inhaltsverzeichnis

ABBILDUNGSVERZEICHNIS

QUELLCODEAUSSCHNITTSVERZEICHNIS

TABELLENVERZEICHNIS

QUELLCODEDOKUMENTATION

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

1 EINLEITUNG

Zunächst wird grob erläutert, worum es Im Abschlussprojekt überhaupt geht.

1.1 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.

1.2 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.

1.3 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.

1.4 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.

2 PROJEKTPLANUNG

Im folgendem wird die allgemeine Planung des Projektes erklärt.

2.1 Projektphasen

Hier befindet sich die Tabelle für die grobe Zeitplanung, wie auch in dem Projektantrag zu finden. Die Tabelle 3: 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

2.2 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.

2.3 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.

3 ANALYSEPHASE

Es erfolgt eine analytische Betrachtung des Projektes mit dem nötigen Hintergrund.

3.1 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.

3.2 Wirtschaftlichkeitsanalyse

Im Folgendem wird der wirtschaftliche Aspekt des Projektes betrachtet.

3.2.1 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.

3.2.2 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 €.

3.2.3 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.

3.3 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.

3.4 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 5: PAP des Spielvorgangs)

3.5 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.

3.6 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.

4 ENTWURFSPHASE

Im folgenden Abschnitt wird der allgemeine Entwurf beschrieben.

4.1 Zielplattform

Wie in der 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.

4.2 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.

4.3 Entwurf der Weboberfläche

Zuerst wurde eine Skizze für die webbasierte Benutzeroberfläche erstellt. (siehe Anhang Abbildung 3) 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.

4.4 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 **Fehler! Verweisquelle konnte nicht gefunden werden.**zur besseren Übersicht befindet sich im Anhang. Dieser Zustand befindet sich direkt nach Anmeldung beider Spieler.

4.5 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 4: 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 5: PAP des Spielvorgangs)

4.6 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.

4.7 Pflichtenheft

Im Rahmen des Projektes wurde ein Pflichtenheft erstellt. Deren Anforderungen wurden bei der Entwicklung berücksichtigt.

5 IMPLEMENTIERUNGSPHASE

Folgend wird die Entwicklung des Projektes beschrieben.

5.1 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.

5.2 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)

Abbildung 1: Eigenes Spielfeld

	1	2	3	4	5	6	7	8	9	10	
A	W	W	W	W	W	W	W	W	X	O	A
B	W	X	X	X	X	X	W	W	X	O	B
C	W	X	O	O	O	X	W	W	X	X	C
D	W	X	X	X	X	X	W	W	W	W	D
E	W	W	W	W	W	W	W	W	W		E
F	W	W	W	W	W	W	W	W	W	W	F
G	W	W	W	W	W	W	W	W	W	W	G
H	X	X	X	X	X	X	X	W	W	W	H
I	X	O	O	O	O	O	X	W	W	W	I
J	X	X	X	X	X	X	X	W	W	W	J
	1	2	3	4	5	6	7	8	9	10	

	1	2	3	4	5	6	7	8	9	10	
A	?	?	?	?	?	?	?	?	?	?	A
B	?	?	?	?	?	?	?	?	?	?	B
C	?	?	?	?	?	?	?	?	?	?	C
D	?	?	?	?	?	?	?	?	?	?	D
E	?	?	?	?	?	?	?	?	?	?	E
F	?	?	?	?	?	?	?	?	?	?	F
G	?	?	?	?	?	?	?	?	?	?	G
H	?	?	?	?	?	?	?	?	?	?	H
I	?	?	?	?	?	?	?	?	?	?	I
J	?	?	?	?	?	?	?	?	?	?	J
	1	2	3	4	5	6	7	8	9	10	

Abbildung 2: Gegnerisches Spielfeld

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.

5.3 Implementierung der Spiellogik

Folgend wird die Umsetzung der Spiellogik beschrieben. Diese unterteilt sich in vier Phasen, die nacheinander implementiert werden mussten, da diese aufeinander aufbauen.

5.3.1 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)

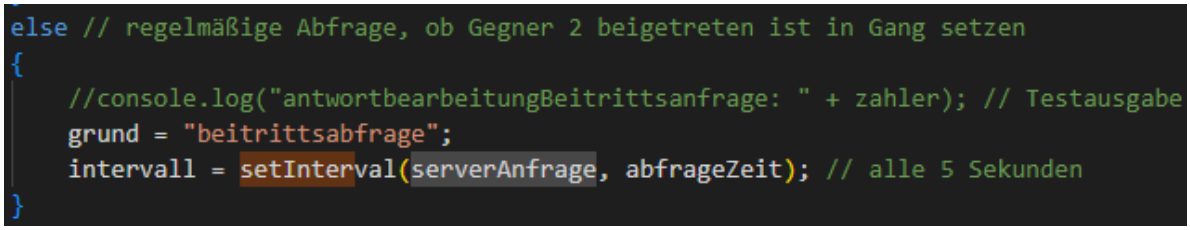
```
let check = true; // alle Felder müssen frei sein
for (let element of schiffsflache) // Prüfen ob Felder frei für Platzierung
{
    //console.log(element); // Testausgabe
    if (element.innerHTML === "X" || element.innerHTML === "O")
    {
        check = false; // unfreies Feld gefunden
        break; // Schleifenabbruch
    }
}
```

Quellcodeausschnitt 1: Suche nach belegten Flächen

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.

5.3.2 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 nächste Seite)



```
else // regelmäßige Abfrage, ob Gegner 2 beigetreten ist in Gang setzen
{
    //console.log("antwortbearbeitungBeitrittsanfrage: " + zahler); // Testausgabe
    grund = "beitrittsabfrage";
    intervall = setInterval(serverAnfrage, abfrageZeit); // alle 5 Sekunden
}
```

Quellcodeausschnitt 2: Intervall für Beitrittsabfrage

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)

```
else # Spieler1 will Spieler2 Namen
{
    $spieldaten = file_get_contents($datei); # Dateiauslesen
    $json = json_decode($spieldaten); # JSON String decodieren -> Objekt
    echo $json->spieler2; # Spieler2 zurückgeben (Namen)
}
```

Quellcodeausschnitt 3: PHP Rückgabe an Ajax

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

5.3.3 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 nächste Seite)

```
function sendeSpielzug()
{
    //console.log("Spielzug wird gesendet.");
    let formDaten = new FormData();

    formDaten.append("spielerID", spielerID); // übergeben ID
    formDaten.append("spielzug", eintrag); // übergeben Spielzug

    // Anfrage wird gesendet
    xhttp.send(formDaten);
}
```

Quellcodeausschnitt 4: Senden des Spielzugs an PHP

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)

```
foreach ($schiffsammlung as &$schiff) # & => Referenz
{
    foreach ($schiff as &$feld) # & => Referenz
    {
        # prüfen ob getroffen; übergebe Array-Index -> $treffer
        if (($treffer = array_search($spielzug, $schiff)) !== false)
        {
            #print_r($schiff); # Testausgabe
            #echo "Treffer: ".$treffer; # Testazsgabe
            unset($schiff[$treffer]); # Feld aus Schiff-Array nehmen
            $schiff = array_values($schiff); # Entfernen von Index (alle)
            $status = "T"; # (T)reffer

            if (count($schiff) === 0) # prüfen ob keine mehr beschießbaren Felder
            {
                $status = "V"; # (V)ersenkt
            }
            #print_r($schiff); # Testausgabe
        }
    }
    if ($leer) # solange leer ist überprüfe, ob Schiff leer
    {
        $leer = empty($schiff); # nicht leeres Schiff gefunden
    }
    #print_r($schiffsammlung); # Testausgabe
}
```

Quellcodeausschnitt 5: Trefferüberprüfung

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.

5.3.4 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)

```
function statistikausgabe() // Eintrag in Statistikbereich unter der Schiffanzeige
{
    let ausgabe = "<br><br>" + treffer + " Treffer von " + zuganzahl + " Schüssen<br>"; // x Treffer von y Schüssen
    ausgabe += "<br>Eigene Trefferquote: " + parseInt(100 * treffer / zuganzahl) + " %"; // ganze % ohne Kommastellen

    document.getElementById("statistik").innerHTML = ausgabe;
}
```

Quellcodeausschnitt 6: Statistikausgabe

6 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.

7 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.

8 FAZIT

Im Folgendem wird das Ergebnis verglichen und reflektiert.

8.1 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)

8.2 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. Eine *Beitrittsabfrage* wurde bei einer 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.

8.3 Ausblick

In der Demoversion wurden aufgrund der begrenzten Zeit das Design der Weboberfläche des Spiels einfach gehalten. Die Größe der Felder wurden zwar dynamisch gestaltet, jedoch sind diese nicht 100 % quadratisch. Ebenso ist die Platzierung des oberen Bereichs der Spielernamen noch zu zentrieren für die Vollversion.

Des Weiteren ist der Quellcode an manchen Stellen ähnlich aufgebaut und müsste noch strukturierter gestaltet werden. Auch ein Beispiel dafür ist der ähnliche Aufbau zur Feststellung der Gewinnbedingung, die Serverseitig verläuft und der Verlierbediengung, die clientseitig ermittelt wird. Hierbei wurde ursprünglich vorgesehen, so wenig Daten wie nötig in die Textdatei abzulegen. Für die Vollversion soll die Auswertung überwiegend serverseitig durch PHP erstellt werden.

Abschließend soll die Vollversion mehrere Spielmatches gleichzeitig ermöglichen. Für die Demoversion wurde dies einfach gehalten, in dem das Spiel auf 2 Spieler beschränkt wurden und jeder weitere Spieler, der sich registrieren möchte, bekommt eine Meldung, dass ein Spiel bereits im Gange ist und somit kein Spielbeitritt mehr möglich ist. Um ein erneutes Spielen zuzulassen muss jedoch die JSON-Datei „spiel.json“ mit den Spielinformationen wieder gelöscht werden. Dies muss in der Vollversion noch angepasst werden.

9 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
https://w1.pngwing.com/pngs/751/1019/png-transparent-butterfly-design-sinking-of-the-rms-titanic-ship-shipwrecking-maritime-disaster-boat-yellow-pollinator.png

10 ANHANG

10.1 Tabellen

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

10.2 Abbildungen

Abbildung 3: Skizze der Webseite

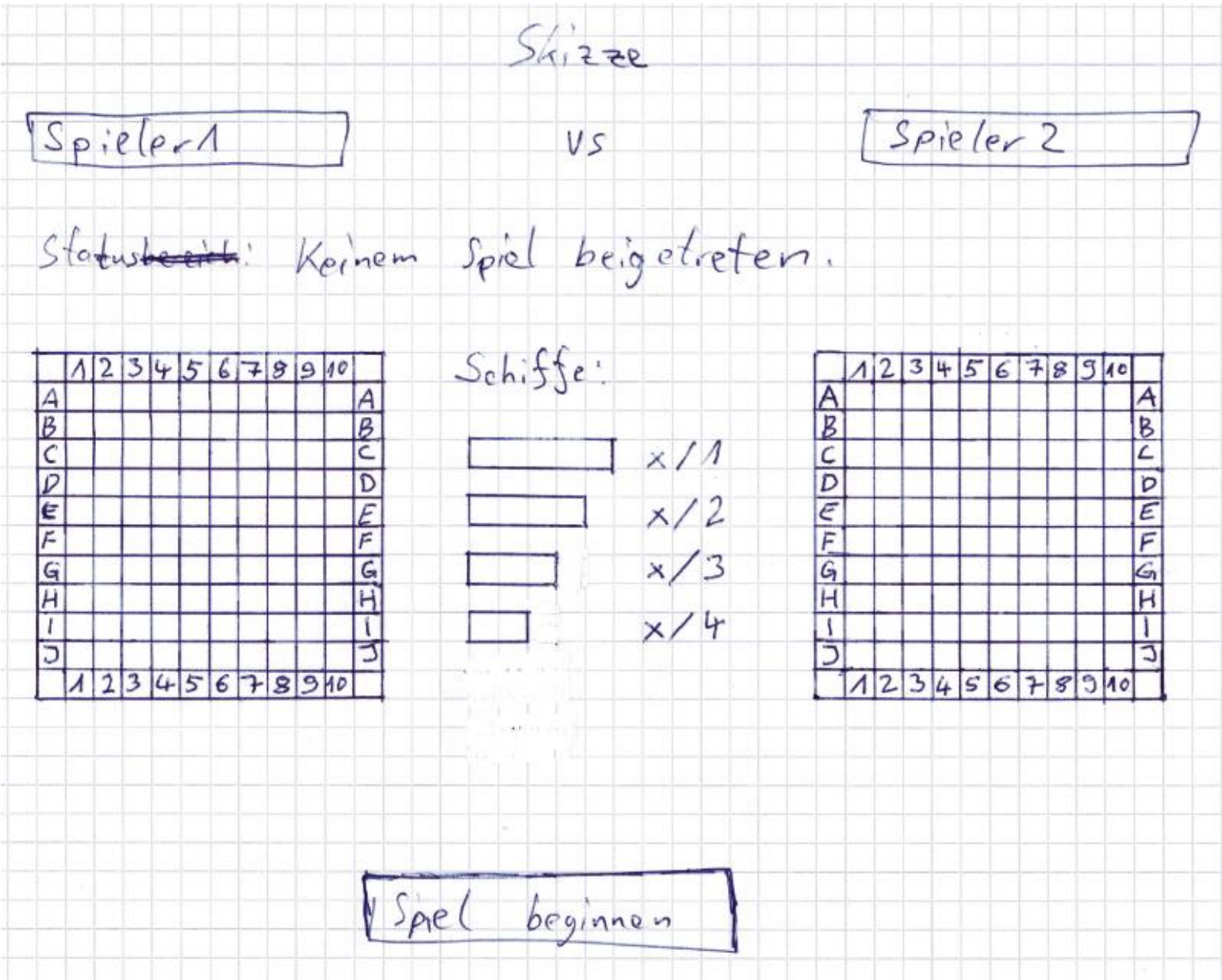


Abbildung 4: PAP des Registrierungsvorgangs zum Spiel

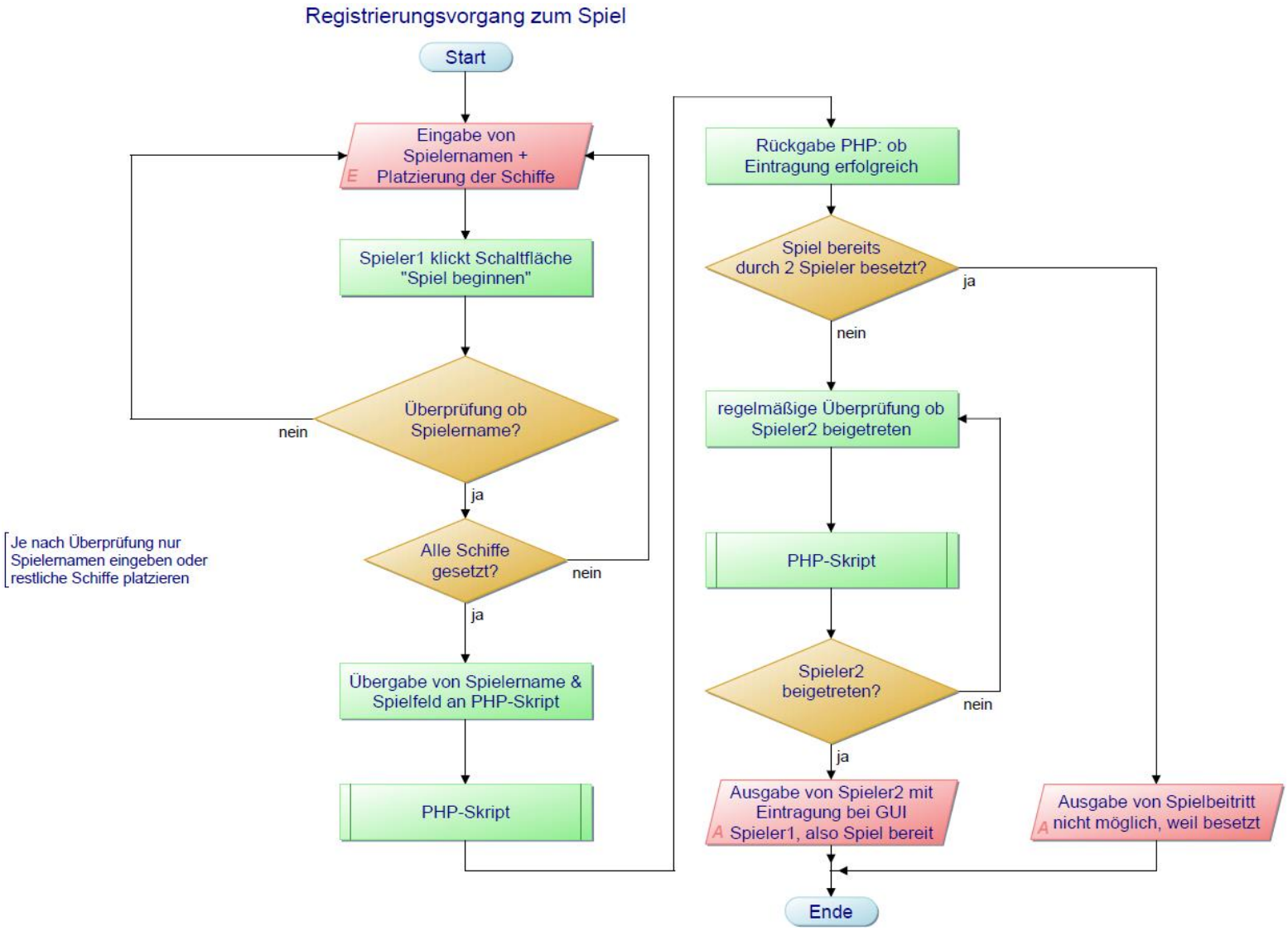


Abbildung 5: PAP des Spielvorgangs

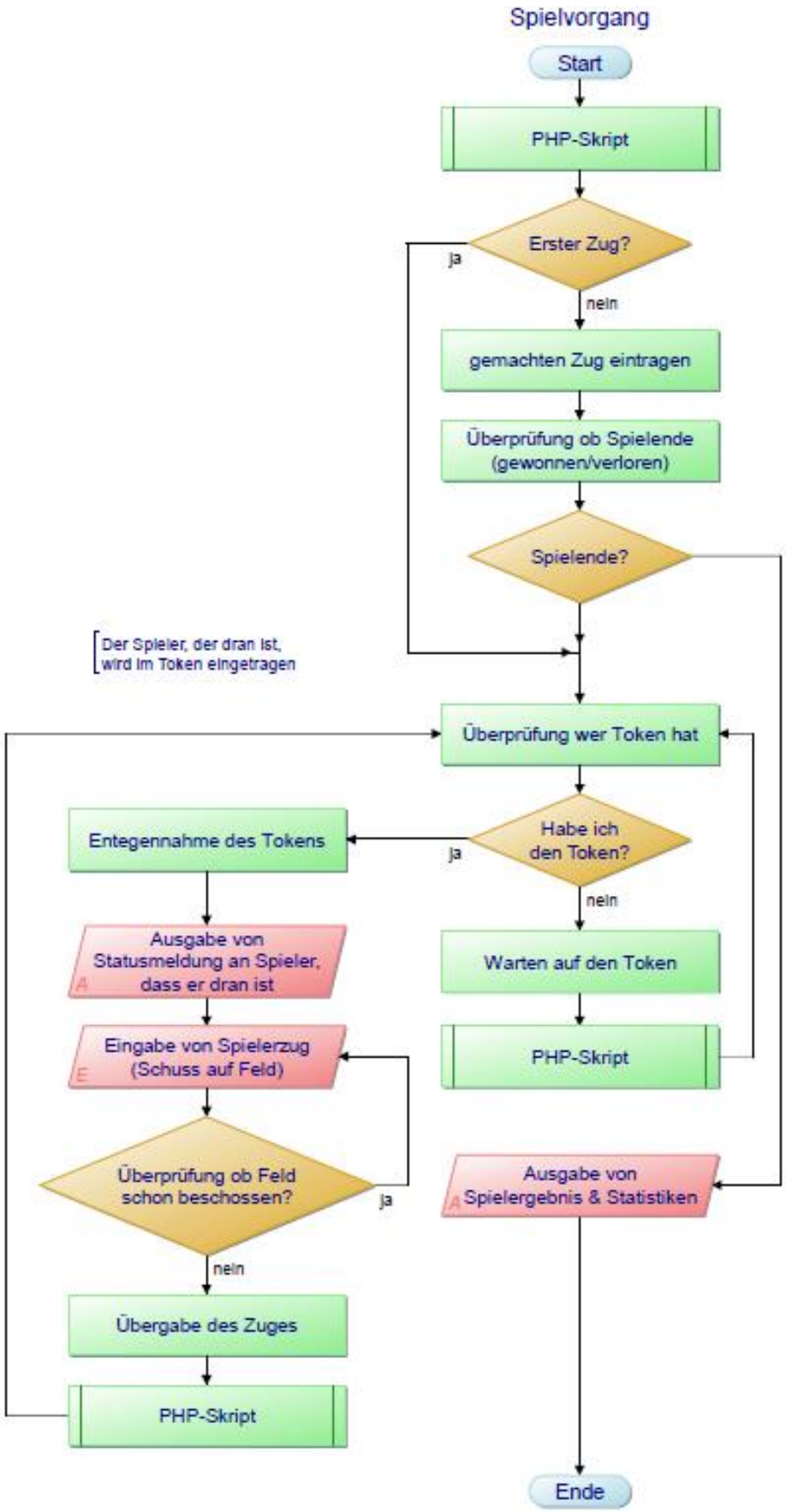


Abbildung 6: Oberflächenentwurf

Spielername vs

Schiffe platzieren.

	1	2	3	4	5	6	7	8	9	10	
A	W	W	W	W	W	W	W	W	W	W	A
B	W	W	W	W	W	W	W	W	W	W	B
C	W	W	W	W	W	W	W	W	W	W	C
D	W	W	W	W	W	W	W	W	W	W	D
E	W	W	W	W	W	W	W	W	W	W	E
F	W	W	W	W	W	W	W	W	W	W	F
G	W	W	W	W	W	W	W	W	W	W	G
H	W	W	W	W	W	W	W	W	W	W	H
I	W	W	W	W	W	W	W	W	W	W	I
J	W	W	W	W	W	W	W	W	W	W	J
	1	2	3	4	5	6	7	8	9	10	

Schiffe:

▪▪▪▪▪ 0 / 1
▪▪▪▪ 0 / 2
▪▪▪ 0 / 3
▪▪ 0 / 4

	1	2	3	4	5	6	7	8	9	10	
A	?	?	?	?	?	?	?	?	?	?	A
B	?	?	?	?	?	?	?	?	?	?	B
C	?	?	?	?	?	?	?	?	?	?	C
D	?	?	?	?	?	?	?	?	?	?	D
E	?	?	?	?	?	?	?	?	?	?	E
F	?	?	?	?	?	?	?	?	?	?	F
G	?	?	?	?	?	?	?	?	?	?	G
H	?	?	?	?	?	?	?	?	?	?	H
I	?	?	?	?	?	?	?	?	?	?	I
J	?	?	?	?	?	?	?	?	?	?	J
	1	2	3	4	5	6	7	8	9	10	

Spiel beginnen

10.3 Pflichtenheft

10.3.1 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.

10.3.2 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.

10.4 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"]
	]
}
```

10.5 Benutzerdokumentation

Kundendokumentation - Abbildung 1: Deckblattbild

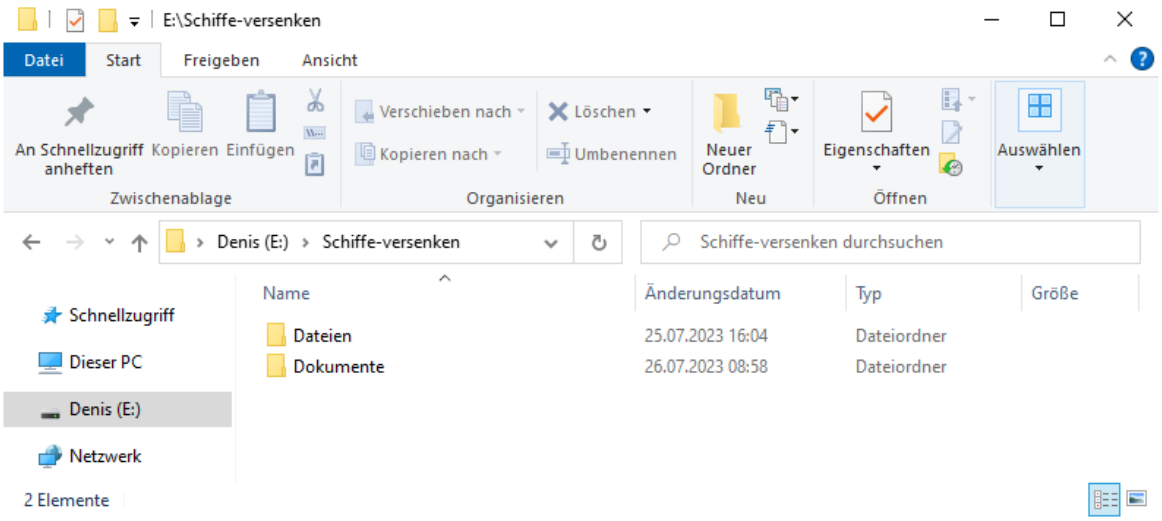
10.5.1 Einleitung

Das Spiel „Schiffe versenken“ ist ein webbasiertes Spiel, dass motorisch eingeschränkten Schülern den Umgang mit der Computermaus zu trainieren. Diese Dokumentation ist für die Lehrer gedacht, die das Spiel im Betrieb nehmen und den Schülern den Umgang damit erklären sollen. Die Demoversion wurde auf den Browsern Mozilla Firefox und Google Chrome getestet, somit wird empfohlen auch diese für das Spiel zu benutzen.

10.5.2 Installation

Die Installation der Anwendung ist grundsätzlich einfach, es müssen jedoch einige Aspekte berücksichtigt werden.

Alle ausgelieferten Projektdaten befinden sich in einer Verzeichnisstruktur (siehe Abbildung nächste Seite). Alle Daten (index.html, style.css, ajax.js, script.php) im Ordner „Dateien“ müssen in das für Webseiten vorgesehene Verzeichnis kopiert werden. Bei einem Apache Webserver trägt dieses Verzeichnis üblicherweise die Bezeichnung „htdocs“.



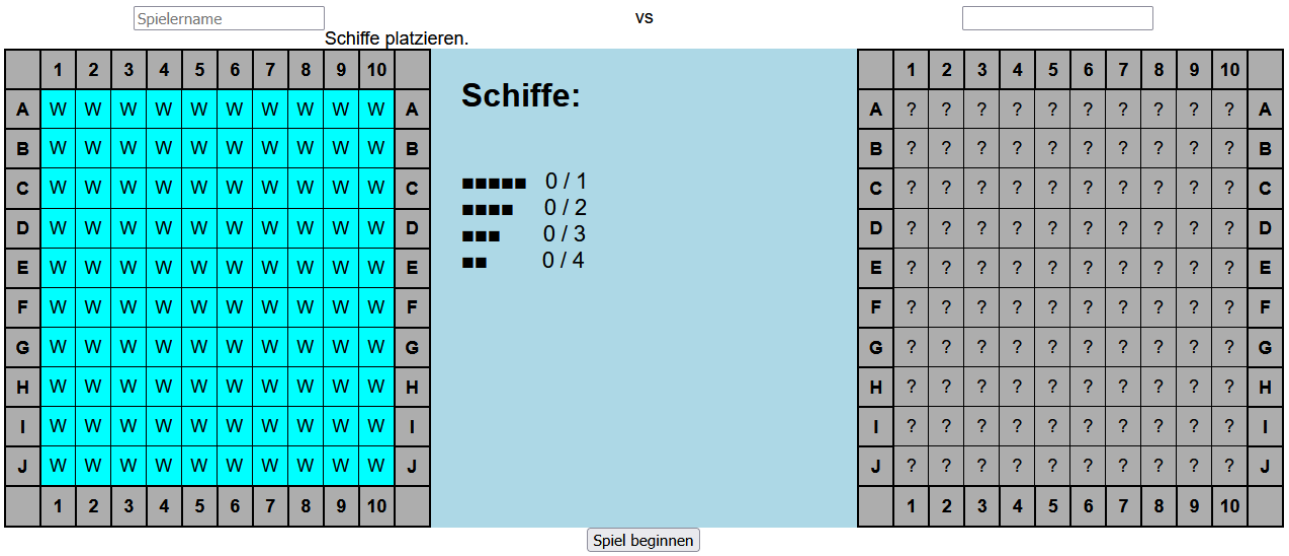
Kundendokumentation - Abbildung 2: Verzeichnisstruktur

Im Ordner „Dokumente“ befindet sich die Kundendokumentation mit allen nötigen Informationen bezüglich des Spiels „Schiffe versenken“.

Aufgrund der Limitierung der Demoversion ist es außerdem nötig, nach jedem Spiel die erzeugt „spiel.json“-Datei zu löschen, um ein neues Spiel zu ermöglichen.

10.5.3 Spielanleitung

In der unteren Abbildung ist das Spielfeld direkt beim Erstaufruf zu sehen



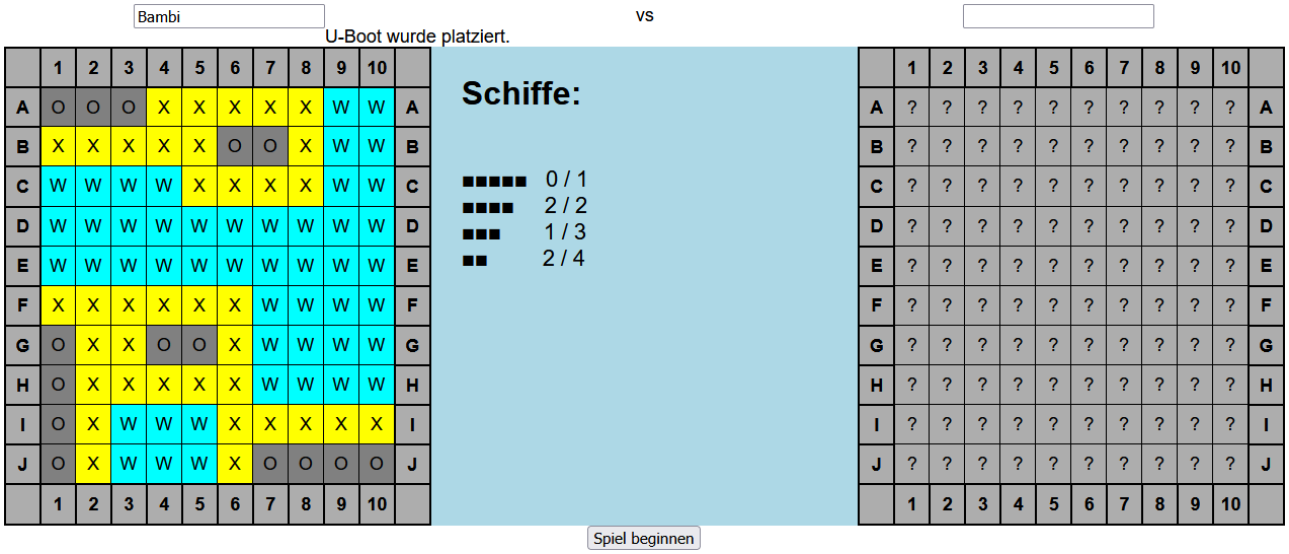
Kundendokumentation - Abbildung 3: Spielfeld beim Start

Hier wird durch Mausklicks auf Start- und Endkoordinate die Platzierung durchgeführt. Welches Schiff platziert wird, ist abhängig der Entfernung des Start- und Endpunktes der Auswahl.

Es existieren folgende Schiffe:

- Ein Schlachtschiff mit der Länge von 5 Feldern
- Zwei Kreuzer mit der Länge von 4 Feldern
- Drei Zerstörer mit der Länge von 3 Feldern
- Vier U-Boote mit der Länge von 2 Feldern

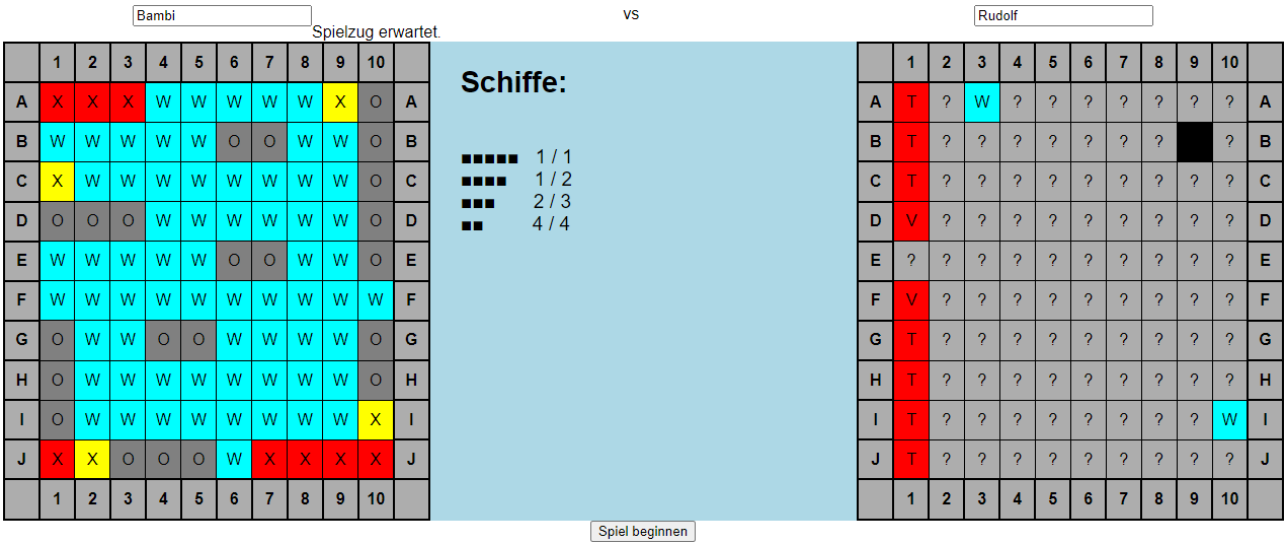
Die Gesamtanzahl ist in der Schiffanzeige in der Mitte der beiden Spielfelder vermerkt. Ebenso vor dem Spielmatch die bisher platzierten Schiffe, als auch später während dem Spiel, die noch nicht versenkten Schiffe. Für die bessere Orientierung wird das mit der Maus anvisierte Spielfeld sowohl auf dem eigenen Spielfeld bei der Platzierung, als auch später bei den Spielzügen auf dem gegnerischen Spielfeld schwarz hervorgehoben.



Kundendokumentation - Abbildung 4: Platzierung der Schiffe

Damit ein Spieler sich registrieren kann, muss er alle 10 Schiffe gerade platzieren, die nicht aneinanderstoßen dürfen und seinen Spielernamen in der linken Textbox über seinem Spielfeld eintragen. Um ein Schiff (graues O) wird während der Platzierungsphase ein Sperrfeld (gelbes X) errichtet, dass dies verhindert. (siehe Abbildung oben)

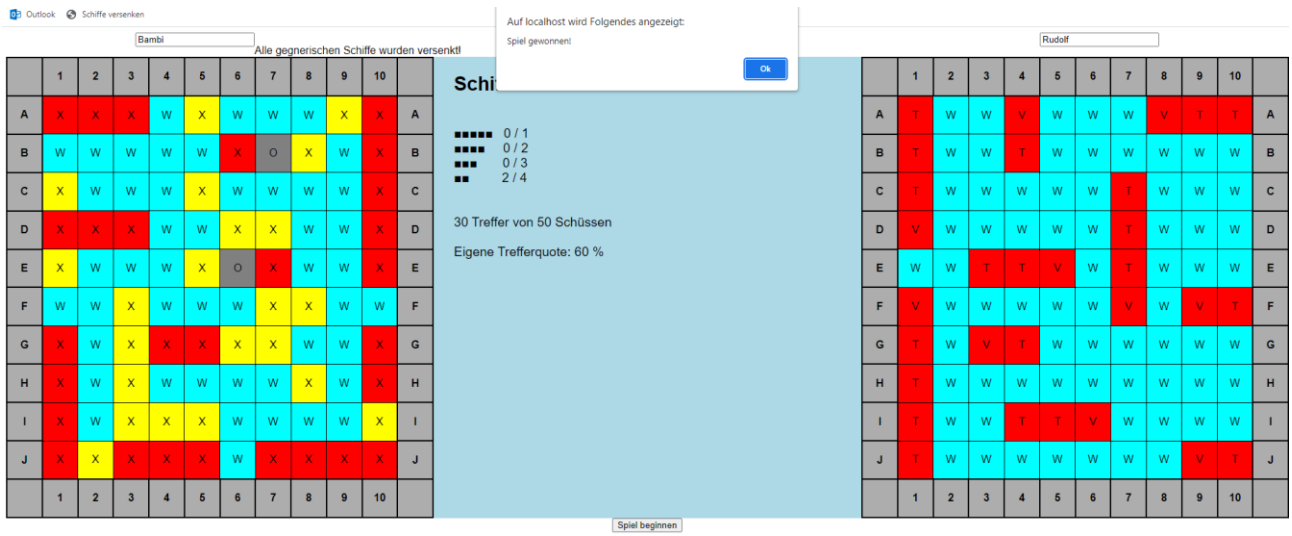
Anschließend wird durch ein Mausklick auf die sich mittig unter dem Bereich der Schiffanzeige befindende Schaltfläche „Spiel beginnen“ der Spieler in ein Spiel eingetragen. Sind beide Spieler eingetragen, wird der Name des Gegners in der rechten Textbox über das gegnerische Spielfeld angezeigt. Danach können die Spieler abwechselnd ihre Spielzüge tätigen. Hierbei muss auf dem gegnerischen Spielfeld (rechts) ein noch nicht beschossenes Feld (graues ?) mit der Maus angeklickt werden. (siehe Abbildung nächste Seite)



Kundendokumentation - Abbildung 5: Spielvorgang

Das Ergebnis des Spielzuges wird dann im gegnerischen Spielfeld eingetragen. Ein rotes „T“ deutet auf einen Treffer des gegnerischen Schiffes hin. Bei einem roten „V“ bedeutet dies, dass das gegnerische Schiff versenkt wurde. Ein blaues „W“ deutet auf einen Schuss ins Wasser hin. Ebenso wird nach Spielzug des Gegners dieser im eigenen Feld eingetragen. Ein rotes „X“ deutet auf einen Treffer hin und ein „gelbes X“ deutet später auf einen Schuss ins Wasser hin.

Wurden alle Schiffe eines Spielers versenkt, wird am Ende das Spielergebnis mit Statistiken unter der Schiffanzeige ausgegeben. (siehe Abbildung unten)



Kundendokumentation - Abbildung 6: Spielende mit Statistik

10.6 Quellcodedokumentation

10.6.1 index.html

```
1 <!DOCTYPE html>
2 <html lang="de">
3
4 <head>
5 <meta charset="utf-8">
6 <title>
7 Schiffe versenken
8 </title>
9 <link rel="stylesheet" href="style.css">
10 <script src="ajax.js"></script>
11 </head>
12
13 <body>
14 <!-- Spieler 1 vs Spieler 2 -->
15 <div id="spielernamen">
16 <input type="text" id="spieler" name="spieler" class="spielernamensfeld"
placeholder="Spielername">
17 <span class="vs">vs</span>
18 <input type="text" id="gegner" name="gegner" class="gegnernamensfeld">
19 </div>
20
21 <div id="status">
22 Status
23 </div>
24
25 <div class="spielbereich">
26 <!-- eigenes Spielfeld -->
27 <table id="tabelle">
28 <tr>
29 <th> </th>
30 <th>1</th>
31 <th>2</th>
32 <th>3</th>
33 <th>4</th>
34 <th>5</th>
35 <th>6</th>
36 <th>7</th>
37 <th>8</th>
38 <th>9</th>
39 <th>10</th>
40 <th> </th>
41 </tr>
42 <tr>
43 <th>A</th>
44 <td id="spieler.a1" class="eigenesSpielfeld">a1</td>
45 <td id="spieler.a2" class="eigenesSpielfeld">a2</td>
46 <td id="spieler.a3" class="eigenesSpielfeld">a3</td>
```

```
47 <td id="spieler.a4" class="eigenesSpielfeld">a4</td>
48 <td id="spieler.a5" class="eigenesSpielfeld">a5</td>
49 <td id="spieler.a6" class="eigenesSpielfeld">a6</td>
50 <td id="spieler.a7" class="eigenesSpielfeld">a7</td>
51 <td id="spieler.a8" class="eigenesSpielfeld">a8</td>
52 <td id="spieler.a9" class="eigenesSpielfeld">a9</td>
53 <td id="spieler.a10" class="eigenesSpielfeld">a10</td>
54 <th>A</th>
55 </tr>
56 <tr>
57 <th>B</th>
58 <td id="spieler.b1" class="eigenesSpielfeld">b1</td>
59 <td id="spieler.b2" class="eigenesSpielfeld">b2</td>
60 <td id="spieler.b3" class="eigenesSpielfeld">b3</td>
61 <td id="spieler.b4" class="eigenesSpielfeld">b4</td>
62 <td id="spieler.b5" class="eigenesSpielfeld">b5</td>
63 <td id="spieler.b6" class="eigenesSpielfeld">b6</td>
64 <td id="spieler.b7" class="eigenesSpielfeld">b7</td>
65 <td id="spieler.b8" class="eigenesSpielfeld">b8</td>
66 <td id="spieler.b9" class="eigenesSpielfeld">b9</td>
67 <td id="spieler.b10" class="eigenesSpielfeld">b10</td>
68 <th>B</th>
69 </tr>
70 <tr>
71 <th>C</th>
72 <td id="spieler.c1" class="eigenesSpielfeld">c1</td>
73 <td id="spieler.c2" class="eigenesSpielfeld">c2</td>
74 <td id="spieler.c3" class="eigenesSpielfeld">c3</td>
75 <td id="spieler.c4" class="eigenesSpielfeld">c4</td>
76 <td id="spieler.c5" class="eigenesSpielfeld">c5</td>
77 <td id="spieler.c6" class="eigenesSpielfeld">c6</td>
78 <td id="spieler.c7" class="eigenesSpielfeld">c7</td>
79 <td id="spieler.c8" class="eigenesSpielfeld">c8</td>
80 <td id="spieler.c9" class="eigenesSpielfeld">c9</td>
81 <td id="spieler.c10" class="eigenesSpielfeld">c10</td>
82 <th>C</th>
83 </tr>
84 <tr>
85 <th>D</th>
86 <td id="spieler.d1" class="eigenesSpielfeld">d1</td>
87 <td id="spieler.d2" class="eigenesSpielfeld">d2</td>
88 <td id="spieler.d3" class="eigenesSpielfeld">d3</td>
89 <td id="spieler.d4" class="eigenesSpielfeld">d4</td>
90 <td id="spieler.d5" class="eigenesSpielfeld">d5</td>
91 <td id="spieler.d6" class="eigenesSpielfeld">d6</td>
92 <td id="spieler.d7" class="eigenesSpielfeld">d7</td>
93 <td id="spieler.d8" class="eigenesSpielfeld">d8</td>
94 <td id="spieler.d9" class="eigenesSpielfeld">d9</td>
95 <td id="spieler.d10" class="eigenesSpielfeld">d10</td>
96 <th>D</th>
97 </tr>
98 <tr>
99 <th>E</th>
```

```
100 <td id="spieler.e1" class="eigenesSpielfeld">e1</td>
101 <td id="spieler.e2" class="eigenesSpielfeld">e2</td>
102 <td id="spieler.e3" class="eigenesSpielfeld">e3</td>
103 <td id="spieler.e4" class="eigenesSpielfeld">e4</td>
104 <td id="spieler.e5" class="eigenesSpielfeld">e5</td>
105 <td id="spieler.e6" class="eigenesSpielfeld">e6</td>
106 <td id="spieler.e7" class="eigenesSpielfeld">e7</td>
107 <td id="spieler.e8" class="eigenesSpielfeld">e8</td>
108 <td id="spieler.e9" class="eigenesSpielfeld">e9</td>
109 <td id="spieler.e10" class="eigenesSpielfeld">e10</td>
110 <th>E</th>
111 </tr>
112 <tr>
113 <th>F</th>
114 <td id="spieler.f1" class="eigenesSpielfeld">f1</td>
115 <td id="spieler.f2" class="eigenesSpielfeld">f2</td>
116 <td id="spieler.f3" class="eigenesSpielfeld">f3</td>
117 <td id="spieler.f4" class="eigenesSpielfeld">f4</td>
118 <td id="spieler.f5" class="eigenesSpielfeld">f5</td>
119 <td id="spieler.f6" class="eigenesSpielfeld">f6</td>
120 <td id="spieler.f7" class="eigenesSpielfeld">f7</td>
121 <td id="spieler.f8" class="eigenesSpielfeld">f8</td>
122 <td id="spieler.f9" class="eigenesSpielfeld">f9</td>
123 <td id="spieler.f10" class="eigenesSpielfeld">f10</td>
124 <th>F</th>
125 </tr>
126 <tr>
127 <th>G</th>
128 <td id="spieler.g1" class="eigenesSpielfeld">g1</td>
129 <td id="spieler.g2" class="eigenesSpielfeld">g2</td>
130 <td id="spieler.g3" class="eigenesSpielfeld">g3</td>
131 <td id="spieler.g4" class="eigenesSpielfeld">g4</td>
132 <td id="spieler.g5" class="eigenesSpielfeld">g5</td>
133 <td id="spieler.g6" class="eigenesSpielfeld">g6</td>
134 <td id="spieler.g7" class="eigenesSpielfeld">g7</td>
135 <td id="spieler.g8" class="eigenesSpielfeld">g8</td>
136 <td id="spieler.g9" class="eigenesSpielfeld">g9</td>
137 <td id="spieler.g10" class="eigenesSpielfeld">g10</td>
138 <th>G</th>
139 </tr>
140 <tr>
141 <th>H</th>
142 <td id="spieler.h1" class="eigenesSpielfeld">h1</td>
143 <td id="spieler.h2" class="eigenesSpielfeld">h2</td>
144 <td id="spieler.h3" class="eigenesSpielfeld">h3</td>
145 <td id="spieler.h4" class="eigenesSpielfeld">h4</td>
146 <td id="spieler.h5" class="eigenesSpielfeld">h5</td>
147 <td id="spieler.h6" class="eigenesSpielfeld">h6</td>
148 <td id="spieler.h7" class="eigenesSpielfeld">h7</td>
149 <td id="spieler.h8" class="eigenesSpielfeld">h8</td>
150 <td id="spieler.h9" class="eigenesSpielfeld">h9</td>
151 <td id="spieler.h10" class="eigenesSpielfeld">h10</td>
152 <th>H</th>
```

```
153 </tr>
154 <tr>
155 <th>I</th>
156 <td id="spieler.i1" class="eigenesSpielfeld">i1</td>
157 <td id="spieler.i2" class="eigenesSpielfeld">i2</td>
158 <td id="spieler.i3" class="eigenesSpielfeld">i3</td>
159 <td id="spieler.i4" class="eigenesSpielfeld">i4</td>
160 <td id="spieler.i5" class="eigenesSpielfeld">i5</td>
161 <td id="spieler.i6" class="eigenesSpielfeld">i6</td>
162 <td id="spieler.i7" class="eigenesSpielfeld">i7</td>
163 <td id="spieler.i8" class="eigenesSpielfeld">i8</td>
164 <td id="spieler.i9" class="eigenesSpielfeld">i9</td>
165 <td id="spieler.i10" class="eigenesSpielfeld">i10</td>
166 <th>I</th>
167 </tr>
168 <tr>
169 <th>J</th>
170 <td id="spieler.j1" class="eigenesSpielfeld">j1</td>
171 <td id="spieler.j2" class="eigenesSpielfeld">j2</td>
172 <td id="spieler.j3" class="eigenesSpielfeld">j3</td>
173 <td id="spieler.j4" class="eigenesSpielfeld">j4</td>
174 <td id="spieler.j5" class="eigenesSpielfeld">j5</td>
175 <td id="spieler.j6" class="eigenesSpielfeld">j6</td>
176 <td id="spieler.j7" class="eigenesSpielfeld">j7</td>
177 <td id="spieler.j8" class="eigenesSpielfeld">j8</td>
178 <td id="spieler.j9" class="eigenesSpielfeld">j9</td>
179 <td id="spieler.j10" class="eigenesSpielfeld">j10</td>
180 <th>J</th>
181 </tr>
182 <tr>
183 <th> </th>
184 <th>1</th>
185 <th>2</th>
186 <th>3</th>
187 <th>4</th>
188 <th>5</th>
189 <th>6</th>
190 <th>7</th>
191 <th>8</th>
192 <th>9</th>
193 <th>10</th>
194 <th> </th>
195 </tr>
196 </table>
197
198 <!-- Schiffanzeige -->
199 <div class="inhalt">
200 <h2>Schiffe:</h2>
201 <br>
202 ■■■■■   <span id="schiff5">0</span> / 1
203 <br>
204 ■■■■     <span id="schiff4">0</span> / 2
205 <br>
```

```
206 ■■■       <span id="schiff3">0</span> / 3
207 <br>
208 ■■         <span id="schiff2">0</span> / 4
209 <div id="statistik"></div>
210 </div>
211
212 <!-- gegnerisches Spielfeld -->
213 <table>
214 <tr>
215 <th> </th>
216 <th>1</th>
217 <th>2</th>
218 <th>3</th>
219 <th>4</th>
220 <th>5</th>
221 <th>6</th>
222 <th>7</th>
223 <th>8</th>
224 <th>9</th>
225 <th>10</th>
226 <th> </th>
227 </tr>
228 <tr>
229 <th>A</th>
230 <td id="gegner.a1" class="gegnerischesSpielfeld">a1</td>
231 <td id="gegner.a2" class="gegnerischesSpielfeld">a2</td>
232 <td id="gegner.a3" class="gegnerischesSpielfeld">a3</td>
233 <td id="gegner.a4" class="gegnerischesSpielfeld">a4</td>
234 <td id="gegner.a5" class="gegnerischesSpielfeld">a5</td>
235 <td id="gegner.a6" class="gegnerischesSpielfeld">a6</td>
236 <td id="gegner.a7" class="gegnerischesSpielfeld">a7</td>
237 <td id="gegner.a8" class="gegnerischesSpielfeld">a8</td>
238 <td id="gegner.a9" class="gegnerischesSpielfeld">a9</td>
239 <td id="gegner.a10" class="gegnerischesSpielfeld">a10</td>
240 <th>A</th>
241 </tr>
242 <tr>
243 <th>B</th>
244 <td id="gegner.b1" class="gegnerischesSpielfeld">b1</td>
245 <td id="gegner.b2" class="gegnerischesSpielfeld">b2</td>
246 <td id="gegner.b3" class="gegnerischesSpielfeld">b3</td>
247 <td id="gegner.b4" class="gegnerischesSpielfeld">b4</td>
248 <td id="gegner.b5" class="gegnerischesSpielfeld">b5</td>
249 <td id="gegner.b6" class="gegnerischesSpielfeld">b6</td>
250 <td id="gegner.b7" class="gegnerischesSpielfeld">b7</td>
251 <td id="gegner.b8" class="gegnerischesSpielfeld">b8</td>
252 <td id="gegner.b9" class="gegnerischesSpielfeld">b9</td>
253 <td id="gegner.b10" class="gegnerischesSpielfeld">b10</td>
254 <th>B</th>
255 </tr>
256 <tr>
257 <th>C</th>
258 <td id="gegner.c1" class="gegnerischesSpielfeld">c1</td>
```

```
259 <td id="gegner.c2" class="gegnerischesSpielfeld">c2</td>
260 <td id="gegner.c3" class="gegnerischesSpielfeld">c3</td>
261 <td id="gegner.c4" class="gegnerischesSpielfeld">c4</td>
262 <td id="gegner.c5" class="gegnerischesSpielfeld">c5</td>
263 <td id="gegner.c6" class="gegnerischesSpielfeld">c6</td>
264 <td id="gegner.c7" class="gegnerischesSpielfeld">c7</td>
265 <td id="gegner.c8" class="gegnerischesSpielfeld">c8</td>
266 <td id="gegner.c9" class="gegnerischesSpielfeld">c9</td>
267 <td id="gegner.c10" class="gegnerischesSpielfeld">c10</td>
268 <th>C</th>
269 </tr>
270 <tr>
271 <th>D</th>
272 <td id="gegner.d1" class=„gegnerischesSpielfeld“>d1</td>
273 <td id="gegner.d2" class=„gegnerischesSpielfeld“>d2</td>
274 <td id="gegner.d3" class=„gegnerischesSpielfeld“>d3</td>
275 <td id="gegner.d4" class=„gegnerischesSpielfeld“>d4</td>
276 <td id="gegner.d5" class=„gegnerischesSpielfeld“>d5</td>
277 <td id="gegner.d6" class=„gegnerischesSpielfeld“>d6</td>
278 <td id="gegner.d7" class=„gegnerischesSpielfeld“>d7</td>
279 <td id="gegner.d8" class=„gegnerischesSpielfeld“>d8</td>
280 <td id="gegner.d9" class=„gegnerischesSpielfeld“>d9</td>
281 <td id="gegner.d10" class=„gegnerischesSpielfeld“>d10</td>
282 <th>D</th>
283 </tr>
284 <tr>
285 <th>E</th>
286 <td id="gegner.e1" class=„gegnerischesSpielfeld“>e1</td>
287 <td id="gegner.e2" class=„gegnerischesSpielfeld“>e2</td>
288 <td id="gegner.e3" class=„gegnerischesSpielfeld“>e3</td>
289 <td id="gegner.e4" class=„gegnerischesSpielfeld“>e4</td>
290 <td id="gegner.e5" class=„gegnerischesSpielfeld“>e5</td>
291 <td id="gegner.e6" class=„gegnerischesSpielfeld“>e6</td>
292 <td id="gegner.e7" class=„gegnerischesSpielfeld“>e7</td>
293 <td id="gegner.e8" class=„gegnerischesSpielfeld“>e8</td>
294 <td id="gegner.e9" class=„gegnerischesSpielfeld“>e9</td>
295 <td id="gegner.e10" class=„gegnerischesSpielfeld“>e10</td>
296 <th>E</th>
297 </tr>
298 <tr>
299 <th>F</th>
300 <td id="gegner.f1" class=„gegnerischesSpielfeld“>f1</td>
301 <td id="gegner.f2" class=„gegnerischesSpielfeld“>f2</td>
302 <td id="gegner.f3" class=„gegnerischesSpielfeld“>f3</td>
303 <td id="gegner.f4" class=„gegnerischesSpielfeld“>f4</td>
304 <td id="gegner.f5" class=„gegnerischesSpielfeld“>f5</td>
305 <td id="gegner.f6" class=„gegnerischesSpielfeld“>f6</td>
306 <td id="gegner.f7" class=„gegnerischesSpielfeld“>f7</td>
307 <td id="gegner.f8" class=„gegnerischesSpielfeld“>f8</td>
308 <td id="gegner.f9" class=„gegnerischesSpielfeld“>f9</td>
309 <td id="gegner.f10" class=„gegnerischesSpielfeld“>f10</td>
310 <th>F</th>
311 </tr>
```

```
312 <tr>
313 <th>G</th>
314 <td id="gegner.g1" class=„gegnerischesSpielfeld“>g1</td>
315 <td id="gegner.g2" class=„gegnerischesSpielfeld“>g2</td>
316 <td id="gegner.g3" class=„gegnerischesSpielfeld“>g3</td>
317 <td id="gegner.g4" class=„gegnerischesSpielfeld“>g4</td>
318 <td id="gegner.g5" class=„gegnerischesSpielfeld“>g5</td>
319 <td id="gegner.g6" class=„gegnerischesSpielfeld“>g6</td>
320 <td id="gegner.g7" class=„gegnerischesSpielfeld“>g7</td>
321 <td id="gegner.g8" class=„gegnerischesSpielfeld“>g8</td>
322 <td id="gegner.g9" class=„gegnerischesSpielfeld“>g9</td>
323 <td id="gegner.g10" class=„gegnerischesSpielfeld“>g10</td>
324 <th>G</th>
325 </tr>
326 <tr>
327 <th>H</th>
328 <td id="gegner.h1">h1</td>
329 <td id="gegner.h2">h2</td>
330 <td id="gegner.h3">h3</td>
331 <td id="gegner.h4">h4</td>
332 <td id="gegner.h5">h5</td>
333 <td id="gegner.h6">h6</td>
334 <td id="gegner.h7">h7</td>
335 <td id="gegner.h8">h8</td>
336 <td id="gegner.h9">h9</td>
337 <td id="gegner.h10">h10</td>
338 <th>H</th>
339 </tr>
340 <tr>
341 <th>I</th>
342 <td id="gegner.i1" class=„gegnerischesSpielfeld“>i1</td>
343 <td id="gegner.i2" class=„gegnerischesSpielfeld“>i2</td>
344 <td id="gegner.i3" class=„gegnerischesSpielfeld“>i3</td>
345 <td id="gegner.i4" class=„gegnerischesSpielfeld“>i4</td>
346 <td id="gegner.i5" class=„gegnerischesSpielfeld“>i5</td>
347 <td id="gegner.i6" class=„gegnerischesSpielfeld“>i6</td>
348 <td id="gegner.i7" class=„gegnerischesSpielfeld“>i7</td>
349 <td id="gegner.i8" class=„gegnerischesSpielfeld“>i8</td>
350 <td id="gegner.i9" class=„gegnerischesSpielfeld“>i9</td>
351 <td id="gegner.i10" class=„gegnerischesSpielfeld“>i10</td>
352 <th>I</th>
353 </tr>
354 <tr>
355 <th>J</th>
356 <td id="gegner.j1" class=„gegnerischesSpielfeld“>j1</td>
357 <td id="gegner.j2" class=„gegnerischesSpielfeld“>j2</td>
358 <td id="gegner.j3" class=„gegnerischesSpielfeld“>j3</td>
359 <td id="gegner.j4" class=„gegnerischesSpielfeld“>j4</td>
360 <td id="gegner.j5" class=„gegnerischesSpielfeld“>j5</td>
361 <td id="gegner.j6" class=„gegnerischesSpielfeld“>j6</td>
362 <td id="gegner.j7" class=„gegnerischesSpielfeld“>j7</td>
363 <td id="gegner.j8" class=„gegnerischesSpielfeld“>j8</td>
364 <td id="gegner.j9" class=„gegnerischesSpielfeld“>j9</td>
```

```
365 <td id="gegner.j10" class=„gegnerischesSpielfeld“>j10</td>
366 <th>J</th>
367 </tr>
368 <tr>
369 <th> </th>
370 <th>1</th>
371 <th>2</th>
372 <th>3</th>
373 <th>4</th>
374 <th>5</th>
375 <th>6</th>
376 <th>7</th>
377 <th>8</th>
378 <th>9</th>
379 <th>10</th>
380 <th> </th>
381 </tr>
382 </table>
383
384 </div>
385
386 <div class="mittig">
387 <input type="button" id="start" name="start" value="Spiel beginnen">
388 </div>
389 </body>
390
391 </html>
392
```

10.6.2 style.css

```
1 body /* Schriftauswahl */
2 {
3 font-family: Arial, Helvetica, sans-serif;
4 }
5
6 .mittig /* zentrieren */
7 {
8 text-align: center;
9 }
10
11 /* Zentrierung und Rand aller Spielfelder */
12
13 table,
14 td
15 {
16 border: 1px solid black;
17 border-collapse: collapse;
18 text-align: center;
19 }
20
21 th /* Beschriftung am Rand */
22 {
23 border: 2px solid black;
24 width: calc(100% / 12);
25 }
26
27 th, td /* Höhe anpassen, damit quadratisch */
28 {
29 height: calc(calc(calc(100vw - 15px) / 3) / 12);
30 }
31
32 td:hover /* Selektion */
33 {
34 background-color: black;
35 }
36
37 /* Gleichmäßiges Gitter für 3 gleichgroße Kontainer */
38
39 .spielbereich
40 {
41 display: grid;
42 grid-template-columns: 1fr 1fr 1fr;
43 background-color: rgb(173, 173, 173);
44 width: calc(100vw - 15px);
45 }
46
47 .eigenesSpielfeld /* Starthintergrundfarbe */
48 {
49 background-color: aqua;
50 }
```

```
51
52 .inhalt /* Anpassung der Schiffanzeige */
53 {
54 font-size: larger;
55 padding-left: 30px;
56 background-color: lightblue;
57 }
58
59 .spielernamensfeld /* linksbündig mit Abstand zu links */
60 {
61 float: left;
62 margin-left: 10vw;
63 }
64
65 .gegnernamensfeld /* rechtsbündig mit Abstand zu rechts */
66 {
67 float: right;
68 margin-right: 10vw;
69 }
70
71 .vs /* Zentrierung durch Abstand zu links */
72 {
73 margin-left: 24vw;
74 }
75
```

10.6.3 ajax.js

```
1 // Variablen
2 //let zahler = 0; // Serveranfragen-Zähler für Tests
3 let anzahl = 10; // für Zeilen & Spalten -> 10 x 10 = 100 Felder
4 let index = 'a'; // Zeilenanfang für Schleifen
5 let spieleraktion = ""; // Klick des Spielers (ID des angeklicktgen Elements)
6 let spielerID = ""; // vom Server zugewiesen
7 let grund = ""; // Zweck der Abfrage an den Server:
"beitritt","beitrittsabfrage","token","spielzug"
8 let intervall = null; // zur Speicherung der ID des Timers
9 let abfrageZeit = 5000; // alle 5 Sekunden -> Timer
10 let eintrag = ""; // Feld ID vom Spielzug
11
12 // für Statistik
13 let zuganzahl = 0; // Gesamanzahl der eigenen gemachten Spielzüge
14 let treffer = 0; // Anzahl der Treffer der eigenen Spielzüge
15
16 // Anzahl der Schiffe (mit Feldgröße)
17 let schiff2 = 4; // 4 U-Boote
18 let schiff3 = 3; // 3 Zerstörer
19 let schiff4 = 2; // 2 Kreuzer
20 let schiff5 = 1; // 1 Schlachtschiff
21
22 let schiffe = []; // Schiffsammlung (Array)
23 let schifflange = []; // Schifflänge für die Schiffanzeige -> Zuordnung zu "schiffe"
24
25 // Startzustand herstellen nach Laden der Seite
26 window.addEventListener("load", ladenErfolgreich);
27 function ladenErfolgreich()
28 {
29 document.getElementById("start").addEventListener("click", meldungPlatzierung); //
Meldung an Spieler, da Schiffe nicht platziert
30 vorbereiten("spieler", feldauswahl); // was soll vorbereitet werden? "feldauswahl" /
"spielzug" + welches Feld
31 //vorbereiten("gegner", spielzug); // Testaufruf zur Überprüfung
32 vorbereitenFelder();
33 document.getElementById("status").innerHTML = "Schiffe platzieren.";
34 }
35
36 function meldungPlatzierung() // Information für Spielstart
37 {
38 alert("Es wurden noch nicht alle Schiffe platziert."); // Meldung an Spieler
39 }
40
41 function timerStarten() // Verbindungsaufbau für Spiel starten
42 {
43 if (document.getElementById("spieler").value !== "") // Spielernamen eigegeben?
44 {
45 if(document.getElementById("spieler").value.includes(";"))
46 {
47 alert("Ungültiges Zeichen ';' im Spielernamen entdeckt!");
```

```
48 }
49 else
50 {
51 document.getElementById("spieler").readOnly = true; // Feld für Spielernamen
sperren
52 document.getElementById("start").removeEventListener("click", timerStarten); //
nur 1x Spiel beginnen
53 grund = "beitritt";
54 serverAnfrage(); // Spielbeitritt
55 }
56 }
57 else
58 {
59 alert("Spielernamen eingeben!"); // Meldung, dies nachzuholen
60 }
61 }
62
63 // Spalzierung der Schiffe (feld="spieler"; auswahl=feldauswahl) / Spielzüge (feld="gegner";
auswahl=spielzug)
64 function vorbereiten(feld, auswahl) // Parameter
65 {
66 //console.log("Plazierung wird vorbereitet..."); // Testausgabe
67 index = 'a'; // Zurücksetzen des Indexes
68
69 // Für jede Reihe ( A - J )
70 for (let zahlReihe = 1; zahlReihe <= anzahl; zahlReihe++)
71 {
72 // jeweilige Reihe
73 for (let zahl = 1; zahl <= anzahl; zahl++)
74 {
75 //console.log("Index: "+index); // Testausgabe
76 let idFeld = feld + "." + index + zahl;
77 //console.log("idFeld add: " + idFeld); // id-Ausgabe zum Test
78 document.getElementById(idFeld).addEventListener("click", auswahl);
79 //console.log(auswahl); // Testausgabe
80 }
81 index = String.fromCharCode(index.charCodeAt(0) + 1); // nächster Buchstabe
82 }
83 }
84
85 // Logik der Platzierung mit Einhaltung der Regeln
86
87 function feldauswahl()
88 {
89 //console.log(this.id); // id des Elements (Testausgabe)
90 let id = this.id;
91 const schiffsflache = id.split("."); // ID aufteilen
92 let aktion = schiffsflache[1]; // Fläche-Koordinaten zuweisen
93 //console.log(aktion); // Testausgabe für das reine Feld
94
95 if (spieleraktion === "") // Speicherung des ersten Klicks
96 {
97 spieleraktion = aktion;
```

```
98 }
99 else // Logik für 2. Auswahl
100 {
101 //console.log(spieleraktion + "." + aktion);
102
103 // Überprüfung ob gerade
104 if (spieleraktion.charAt(0) === aktion.charAt(0) || spieleraktion.slice(1) === aktion.
slice(1))
105 {
106 //console.log("Schiff gerade"); // Testausgabe
107 let lang = 0;
108 let anfang = "";
109 let richtung = "";
110
111 if (spieleraktion.charAt(0) === aktion.charAt(0)) // Zeile
112 {
113 lang = aktion.slice(1) - spieleraktion.slice(1); // Berechnung
114 richtung = "h"; // Anpassung der Ausrichtung
115 }
116 else // Spalte
117 {
118 lang = spieleraktion.charCodeAt(0) - aktion.charCodeAt(0); // Berechnung
119 richtung = "v"; // Anpassung der Ausrichtung
120 }
121 lang = korrigiereLang(lang); // Schifflänge korrigieren wegen Rechnung +
Vorzeichen
122 //console.log("Länge: "+lang); // Testausgabe
123 //console.log("Richtung: "+richtung); // Testausgabe
124 if (6 > lang && lang > 1) // Überprüfung ob gültige Länge
125 {
126 //let temp = null; // temporäres Zahl oder Buchstabe zur Überprüfung (Zeile /
Spalte)
127
128 /* Testausgaben zur Fehlerüberprüfung der Logik
129 //console.log("Spieleraktion: " + spieleraktion + ", Aktion: " + aktion);
130 //console.log("Character: ");
131 //console.log(spieleraktion.charAt(0) <= aktion.charAt(0));
132 //console.log("Integer: ");
133 //console.log(parseInt(spieleraktion.slice(1)) <= parseInt(aktion.slice(1)));
134 */
135 if (spieleraktion.charAt(0) <= aktion.charAt(0) && (parseInt(spieleraktion.
slice(1)) <= parseInt(aktion.slice(1))))
136 {
137 //console.log("if"); // Testausgabe
138 anfang = spieleraktion; // erste Aktion
139 }
140 else
141 {
142 //console.log("else"); // Testausgabe
143 anfang = aktion; // zweite Aktion
144 }
145
146 //console.log("Anfang: " + anfang); // Testausgabe
```

```
147 platziereSchiff(anfang, lang, richtung); // Übergabe der benötigten Werte
148 }
149 else
150 {
151 alert("Gewählte Schiffgröße existiert nicht!\nLänge: " + lang); // Meldung an
Spieler
152 }
153 }
154 else
155 {
156 alert("Schiffe können nicht diagonal platziert werden!"); // Meldung an Spieler
157 }
158
159 spieleraktion = ""; // Spieleraktion wird zurückgesetzt (1. Klick)
160 }
161 }
162
163 // Korrektur der berechneten Schiffslänge
164
165 function korrigiereLang(wert)
166 {
167 if (wert < 0) // Wert negativ
168 {
169 wert /= -1; // Vorzeichenumkehr zu +
170 }
171 return ++wert; // +1 für Korrektur
172 }
173
174 // Platzierung der Schiffe auf dem Spielfeld mit Sperrflächen
175
176 function platziereSchiff(start, wert, ausrichtung)
177 {
178 // start = Schleifenanfang (links oben), wert = Länge, ausrichtung vertikal / horizontal
(v/h)
179 //console.log("Start: " + start + ", Wert: " + wert + ", Ausrichtung: " + ausrichtung);
// Testausgabe zur Überprüfung der Werte
180
181 let schifftyp = "";
182 let schiffVorhanden = false;
183 let schiffsflache = []; // Vorlage zur Reservierung des benötigten Bereichs für das Schiff
184 let sperrbereich = []; // Vorlage zur Sperrung der Bereiche nebenan
185 //console.log("Ausrichtung: " + ausrichtung); // Testausgabe
186
187 switch (wert) // Schifftyp festlegen + Prüfung ob vorhanden (Bestand > 0)
188 {
189 case 5:
190 schifftyp = "Schlachtschiff";
191 if (schiff5 > 0)
192 {
193 schiffVorhanden = true;
194 }
195 break;
196 case 4:
```

```
197 schifftyp = "Kreuzer";
198 if (schiff4 > 0)
199 {
200 schiffVorhanden = true;
201 }
202 break;
203 case 3:
204 schifftyp = "Zerstörer";
205 if (schiff3 > 0)
206 {
207 schiffVorhanden = true;
208 }
209 break;
210 case 2:
211 schifftyp = "U-Boot";
212 if (schiff2 > 0)
213 {
214 schiffVorhanden = true;
215 }
216 }
217
218 if (schiffVorhanden) // wenn Schifftyp Anzahl > 0
219 {
220 // benötigte Elemente in schiffsflache laden + sperrbereich neben Schiff befüllen
221 for (let zeichne = 0; wert > zeichne; zeichne++)
222 {
223 let IDsperreDavor = "";
224 let IDsperreDanach = "";
225 //console.log("Start: " + start); // Testausgabe
226 let IDzusatz = "";
227 if (ausrichtung === "h") // horizontal
228 {
229 let temp = zeichne + parseInt(start.slice(1)); // zur vorbereitung der
variablen Zahl der ID
230 //console.log("Temp: " + temp); // Testausgabe
231 IDzusatz = start.charAt(0) + temp; // für die Fläche des Schiffs
232 //console.log("IDzusatz: " + IDzusatz);
233 IDsperreDavor = String.fromCharCode(start.charCodeAt(0) - 1) + temp;
234 IDsperreDanach = String.fromCharCode(start.charCodeAt(0) + 1) + temp;
235 //console.log("von " + IDsperreDavor + " bis " + IDsperreDanach); //
Testausgabe für davor und danach sperren
236 }
237 else // vertikal
238 {
239 let temp = String.fromCharCode(zeichne + start.charCodeAt(0)); // zur
vorbereitung des variablen Buchstabens der ID
240 //console.log("Temp: " + temp); // Testausgabe
241 IDzusatz = temp + start.slice(1); // für die Fläche des Schiffs
242 //console.log("IDzusatz: " + IDzusatz);
243 IDsperreDavor = temp + (start.slice(1) - 1);
244 IDsperreDanach = temp + (parseInt(start.slice(1)) + 1);
245 //console.log("von " + IDsperreDavor + " bis " + IDsperreDanach); //
Testausgabe für davor und danach sperren
```

```
246 }
247 //console.log("schiffsflache..."); // Testausgabe
248
249 // nur existierende Elemente in Array schreiben
250 let pruefen = document.getElementById("spieler." + IDsperreDavor);
251 if (pruefen !== null)
252 {
253 sperrbereich.push(pruefen);
254 }
255 pruefen = document.getElementById("spieler." + IDsperreDanach);
256 if (pruefen !== null)
257 {
258 sperrbereich.push(pruefen);
259 }
260 schiffsflache.push(document.getElementById("spieler." + IDzusatz));
261 }
262
263 let schiffanfang = schiffsflache[0].id.split("."); // Sperrelement für Start
264 let schiffende = schiffsflache[schiffsflache.length - 1].id.split("."); //
Sperrelement für Ende
265
266 // Sperrbereich erweitern: eins nach vorne + eins nach hinten (Schiff), falls möglich
(document.getElementById(x) == null, wenn nicht da)
267
268 if (ausrichtung === "h") // horizontale Ausrichtung
269 {
270 // Verschiebung nach vorne bzw. hinten (vom Schiff aus) -> Anpassung des fixen
Wertes (Zahl)
271 let zusatzsperreAnfang = schiffanfang[1].slice(1) - 1; // Zahl von
272 let zusatzsperreEnde = parseInt(schiffende[1].slice(1)) + 1; // Zahl bis
273 //console.log("ZusatzsperreAnfang: " + zusatzsperreAnfang); // Testausgabe
274 //console.log("ZusatzsperreEnde: " + zusatzsperreEnde); // Testausgabe
275
276 for (let zusatz = -1; zusatz < 2; zusatz++) // Verschiebung nach oben -> -1
277 {
278 // linke Seite sperren
279 let temp = String.fromCharCode(schiffanfang[1].charCodeAt(0) + zusatz) +
zusatzsperreAnfang; // Erzeugung der ID links
280 //console.log("ID: " + "spieler." + temp); // Testausgabe
281 let pruefen = document.getElementById("spieler." + temp);
282 if (pruefen !== null)
283 {
284 sperrbereich.push(pruefen);
285 //console.log("Test: "+pruefen); // Testausgabe
286 }
287 pruefen = null; // zurücksetzen
288
289 // rechte Seite sperren
290 temp = String.fromCharCode(schiffende[1].charCodeAt(0) + zusatz) +
zusatzsperreEnde; // Erzeugung der ID rechts
291 pruefen = document.getElementById("spieler." + temp);
292 //console.log("neu Temp: " + temp); // Testausgabe
293 if (pruefen !== null)
```

```
294 {
295 sperrbereich.push(pruefen);
296 //console.log("Test: " + pruefen.id); // Testausgabe
297 }
298 }
299 }
300 else // vertikale Ausrichtung
301 {
302 // Verschiebung nach vorne bzw. hinten (vom Schiff aus) -> Anpassung des fixen
Wertes (Buchstabe)
303 let zusatzsperreAnfang = String.fromCharCode(schiffanfang[1].charCodeAt(0) - 1);
// Buchstabe von
304 let zusatzsperreEnde = String.fromCharCode(schiffende[1].charCodeAt(0) + 1); //
Buchstabe bis
305 //console.log("ZusatzsperreAnfang: " + zusatzsperreAnfang); // Testausgabe
306 //console.log("ZusatzsperreEnde: " + zusatzsperreEnde); // Testausgabe
307
308 for (let zusatz = -1; zusatz < 2; zusatz++) // Verschiebung nach links -> -1
309 {
310 // linke Seite sperren
311 let temp = zusatzsperreAnfang + (parseInt(schiffanfang[1].slice(1)) + zusatz);
// Erzeugung der ID oben
312 //console.log("ID: " + "spieler." + temp); // Testausgabe
313 let pruefen = document.getElementById("spieler." + temp);
314 if (pruefen !== null)
315 {
316 sperrbereich.push(pruefen);
317 //console.log("Test: "+pruefen); // Testausgabe
318 }
319 pruefen = null; // zurücksetzen
320
321 // rechte Seite sperren
322 temp = zusatzsperreEnde + (parseInt(schiffende[1].slice(1)) + zusatz); //
Erzeugung der ID unten
323 pruefen = document.getElementById("spieler." + temp);
324 //console.log("neu Temp: " + temp); // Testausgabe
325 if (pruefen !== null)
326 {
327 sperrbereich.push(pruefen);
328 //console.log("Test: " + pruefen.id); // Testausgabe
329 }
330 //console.log("----------"); // Testausgabe (Trennung der
Schleifendurchläufen)
331 }
332 }
333
334 let check = true; // alle Felder müssen frei sein
335 for (let element of schiffsflache) // Prüfen ob Felder frei für Platzierung
336 {
337 //console.log(element); // Testausgabe
338 if (element.innerHTML === "X" || element.innerHTML === "O")
339 {
340 check = false; // unfreies Feld gefunden
```

```
341 break; // Schleifenabbruch
342 }
343 }
344
345 if (check) // wenn frei
346 {
347 switch (wert) // Schiff entnehmen für Platzierung & Schiffanzeige aktualisieren
348 {
349 case 5:
350 schiff5--;
351 document.getElementById("schiff5").innerHTML = parseInt(document.
getElementById("schiff5").innerHTML) + 1;
352 break;
353 case 4:
354 schiff4--;
355 document.getElementById("schiff4").innerHTML = parseInt(document.
getElementById("schiff4").innerHTML) + 1;
356 break;
357 case 3:
358 schiff3--;
359 document.getElementById("schiff3").innerHTML = parseInt(document.
getElementById("schiff3").innerHTML) + 1;
360 break;
361 case 2:
362 schiff2--;
363 document.getElementById("schiff2").innerHTML = parseInt(document.
getElementById("schiff2").innerHTML) + 1;
364 }
365
366 let array = []; // für IDs aller Flächen eines Schiffes ohne "spieler." Zusatz
367
368 for (const element of schiffsflache) // Schiff platzeren
369 {
370 element.innerHTML = "O";
371 element.style.backgroundColor = "grey";
372 let temp = element.id.split("."); // Spieler und Flächenkoordinaten trennen
373 //console.log(temp[1]); // Testausgabe
374 array.push(temp[1]);
375 }
376 document.getElementById("status").innerHTML = schifftyp + " wurde platziert."; //
Statusausgabe
377 schiffe.push(array); // Schiffe im Array sammeln
378 schifflange.push(wert);
379 //console.log(schiffe); // Testausgabe
380
381 // Bereich drumherum sperren
382 for (const element of sperrbereich) // Sperrfläche um das Schiff drum herum
platzieren
383 {
384 element.innerHTML = "X";
385 element.style.backgroundColor = "yellow";
386 }
387 //console.log(sperrbereich); // Testausgabe des Sperrbereichs
```

```
388
389 if (schiff5 === 0 && schiff4 === 0 && schiff3 === 0 && schiff2 === 0) // Abfrage
ob Schiffe übrig zum platzieren
390 {
391 wasser(); // alle Schiffe wurden platziert
392 }
393
394 }
395 else // mindestens ein Feld belegt
396 {
397 document.getElementById("status").innerHTML = schifftyp + " konnte aufgrund einer
Kollision nicht platziert werden!";
398 }
399
400 //console.log(check); // Testausgabe
401 }
402 else
403 {
404 document.getElementById("status").innerHTML = "Kein " + schifftyp + " mehr verfügbar!"
;
405 }
406 }
407
408 function wasser() // wandelt Sperrflächen zu Wasserflächen um & entfernt die Evenlistener für
die Platzierung der Schiffe
409 {
410 //console.log("Wasser wird verschüttet..."); // Testausgabe
411 index = 'a'; // Zurücksetzen des Indexes
412
413 // Für jede Reihe ( A - J )
414 for (let zahlReihe = 1; zahlReihe <= anzahl; zahlReihe++)
415 {
416 // jeweilige Reihe
417 for (let zahl = 1; zahl <= anzahl; zahl++)
418 {
419 //console.log("Index: "+index); // Testausgabe
420 let idFeld = "spieler." + index + zahl;
421 //console.log(idFeld); // id-Ausgabe zum Test
422 if (document.getElementById(idFeld).innerHTML !== "O") // kein Schiff platziert
423 {
424 document.getElementById(idFeld).innerHTML = "W"; // Wasserzeichen
425 document.getElementById(idFeld).style.backgroundColor = "aqua"; //
Wasserhintergrundfarbe
426 }
427 //console.log("idFeld remove: " + idFeld);
428 document.getElementById(idFeld).removeEventListener("click", feldauswahl); //
entferne die Auswahlmöglichkeit auf dem eigenen Spielfeld
429 }
430 index = String.fromCharCode(index.charCodeAt(0) + 1); // nächster Buchstabe
431 }
432 setTimeout(meldungNachPlatzierung, 200); // damit Meldung erst nach den platzieren des
letzten Schiffs auftaucht (200ms Verzögerung)
433 }
```

```
434
435 function meldungNachPlatzierung() // gibt die Bestätigung für den Spieler aus, dass alle
Schiffe platziert wurden.
436 {
437 alert("Es wurden alle Schiffe platziert.");
438 document.getElementById("start").removeEventListener("click", meldungPlatzierung); //
Hinweis zur Platzierung der Schiffe entfernen
439 document.getElementById("start").addEventListener("click", timerStarten); // Spiel bereit
zum Starten -> Verbinden mit Server
440 }
441
442 function vorbereitenFelder() // alle Spielflächen beider Spielfelder mit Zeichen vorbelegen
443 {
444 //console.log("Wasser wird verschüttet..."); // Testausgabe
445 index = 'a'; // Zurücksetzen des Indexes
446
447 // Für jede Reihe ( A - J )
448 for (let zahlReihe = 1; zahlReihe <= anzahl; zahlReihe++)
449 {
450 // jeweilige Reihe
451 for (let zahl = 1; zahl <= anzahl; zahl++)
452 {
453 //console.log("Index: "+index); // Testausgabe
454 let idFeldSpieler = "spieler." + index + zahl;
455 let idFeldGegner = "gegner." + index + zahl;
456 //console.log(idFeld); // id-Ausgabe zum Test
457 document.getElementById(idFeldSpieler).innerHTML = "W"; // Wasserzeichen
458 document.getElementById(idFeldGegner).innerHTML = "?"; // unbekannte Gegnerflächen
459 //console.log("idFeld remove: " + idFeld); // Testausgabe
460 }
461 index = String.fromCharCode(index.charCodeAt(0) + 1); // nächster Buchstabe
462 }
463 }
464
465 // Spielersuche zum Registrieren in Datei zum Aufbau des Spiels
466 function serverAnfrage()
467 {
468 //zahler++; // AnfrageNr für Tests
469 //console.log("Serveranfrage: " + zahler); // Testausgabe
470
471 // Anfang - Browserweiche
472 try
473 {
474 // Eintrag von Verbindungsdaten in xhttp
475 xhttp = new XMLHttpRequest(); // Firefox
476 }
477 catch (error)
478 {
479 try
480 {
481 xhttp = new ActiveXObject("Msxml2.XMLHTTP"); // Microsoft
482 }
483 catch (error)
```

```
484 {
485 try
486 {
487 xhttp = new ActiveXObject("Microsoft.XMLHTTP");
488 }
489 catch (error)
490 {
491 return;
492 }
493 }
494 }
495 // Ende - Browserweiche
496
497 // liest ständig die Rückmeldungen
498 xhttp.onreadystatechange = ajax;
499
500 // Verbindung wird geöffnet mit method="POST"
501 xhttp.open("POST", "script.php");
502
503 //console.log("Grund der Serveranfrage: " + grund); // Testausgabe
504
505 if (grund === "beitritt") // Abfrage für Grund
506 {
507 verbinden(); // Versuch sich für Spiel zu registrieren
508 }
509 else if (grund === "beitrittsabfrage")
510 {
511 beitrittsabfrage(); // Prüfen ob Spieler 2 beigetreten
512 }
513 else if (grund === "token")
514 {
515 //console.log("Form für Token???"); // Testausgabe
516 tokenAbfrage();
517 }
518 else if (grund === "spielzug") // bei Schuss auf Feld
519 {
520 sendeSpielzug();
521 }
522 }
523
524 function ajax() // Antwort des php-Skripts
525 {
526 // readyState = 4 -> Rückmeldung vollständig ; status = 200 -> Rückmeldung fehlerfrei
527 if (this.readyState == 4 && this.status == 200)
528 {
529 let antwort = this.responseText;
530 //console.log("Testantwort: " + antwort); // Testausgabe
531 // function für weiteren Ablauf
532 if (grund === "beitritt")
533 {
534 antwortbearbeitungBeitrittsanfrage(antwort); // Spieler will sich registrieren
535 }
536 else if (grund === "beitrittsabfrage")
```

```
537 {
538 anwortbearbeitungBeitrittsabfrage(antwort); // Spieler1 fragt nach Spieler2
539 }
540 else if (grund === "token")
541 {
542 antwortbearbeitungTokenAbfrage(antwort);
543 //console.log("Hab ich Token?"); // Testausgabe
544 }
545 else if (grund === "spielzug") // Rückmeldung des Schusses auf ein Feld
546 {
547 antwortbearbeitungSendeSpielzug(antwort);
548 }
549 }
550 }
551
552 function antwortbearbeitungBeitrittsanfrage(antwort) // Anfrageverarbeitung Beitritt
553 {
554 // gesplittetes Array. Übergabe: Spieler1 registriert. oder Spieler2 registriert. Gegner:
[Spieler2 Name]
555 let check = antwort.split(";"); // Trennzeichen ";" -> als Spielername unzulässig
556 spielerID = check[0];
557 //console.log(spielerID); // Testausgabe
558 //console.log("antwortbearbeitungBeitrittsanfrage"); // Testausgabe
559 if (check[1] === "registriert.") // Registrierung für Spiel erfolgreich
560 {
561 vorbereiten("gegner", spielzug); // Spielzug bereit machen
562 //console.log("Check: "+check); // Testausgabe
563
564 if (check.length > 2) // Registrierung für Spieler2 erfolgt
565 {
566 //console.log("Check: " + check); // Testausgabe
567 //console.log("Checkelement: " + check[check.length - 3]); // Testausgabe
568 document.getElementById("gegner").value = check[check.length - 3]; // Spieler2
eintragen
569 alert("Spieler " + check[check.length - 3] + " ist dem Spiel beigetreten."); //
Statusmeldung
570 // direkt Abfrage, ob er dran ist..... -> token
571 //console.log("Token: " + check[check.length - 2]); // Testausgabe des Tokens, ob
Spieler1 oder Spieler2
572 if (spielerID === check[check.length - 2]) // ich (Spieler2) hab Token
573 {
574 //console.log("Ich hab Token."); // Testausgabe
575 grund = "spielzug";
576 document.getElementById("status").innerHTML = "Spielzug erwartet."; //
Statusmeldung -> Spieler ist dran
577 }
578 else // Timer für Abfrage ob Token
579 {
580 grund = "token";
581 intervall = setInterval(serverAnfrage, abfrageZeit);
582 document.getElementById("status").innerHTML = "Warten auf gegnerischen
Spielzug..."; // Statusmeldung -> Spieler ist nicht dran
583 }
```

```
584
585 }
586 else // regelmäßige Abfrage, ob Gegner 2 beigetreten ist in Gang setzen
587 {
588 //console.log("antwortbearbeitungBeitrittsanfrage: " + zahler); // Testausgabe
589 grund = "beitrittsabfrage";
590 intervall = setInterval(serverAnfrage, abfrageZeit); // alle 5 Sekunden
591 }
592 }
593 else // Registrierung fehlgeschlagen
594 {
595 document.getElementById("status").innerHTML = "Spielbeitritt nicht möglich.";
596 alert(antwort); // Meldung: Spiel bereits im Gange.
597 }
598 }
599
600 function verbinden() // Spielbeitrittsversuch
601 {
602 document.getElementById("status").innerHTML = "Spielbeitritt erfolgt...";
603
604 // Vorlage für Formular erstellen + Daten mit "append" anheften
605 let formDaten = new FormData();
606
607 let jsonString = JSON.stringify(schiffe); // JSON String generieren
608 formDaten.append("spieler", document.getElementById("spieler").value);
609 formDaten.append("aufstellung", jsonString); // JSON String mitschicken
610 //console.log(jsonString); // Testausgabe
611
612 // Anfrage wird gesendet
613 xhttp.send(formDaten);
614 }
615
616 function beitrittsabfrage() // alle 5 Sekunden bis Spieler2 beigetreten
617 {
618 // Anfrage wird gesendet
619 xhttp.send(); // keine Form angehängt
620 }
621
622 function anwortbearbeitungBeitrittsabfrage(antwort) // Überprüfung ob Spieler2 beigetreten ist
623 {
624 //console.log("Beitritt: " + antwort); // Testausgabe
625 if (antwort !== "") // nur wenn Spieler2 eingetragen ist
626 {
627 document.getElementById("status").innerHTML = "Warten auf Spielzug...";
628 document.getElementById("gegner").value = antwort; // Gegner für Spieler1 eintragen
629 grund = "token"; // ob Spieler dran ist
630 //clearInterval(intervall); // nur wenn Spieler dran ist....
631 }
632 }
633
634 function tokenAbfrage() // Sende Anfrage an Token
635 {
636 let formDaten = new FormData();
```

```
637
638 formDaten.append("spielerID", spielerID); // übergeben ID
639
640 // Anfrage wird gesendet
641 xhttp.send(formDaten);
642 }
643
644 function antwortbearbeitungTokenAbfrage(antwort) // Überprüfung, ob eigener Token
645 {
646 let temp = antwort.split(";"); // Seperatisieren von Token & letzter Spielzug
647 //console.log("Token: " + temp[0]); // Testausgabe
648 //console.log("ID Feld: " + temp[1]); // Testausgabe
649 if (temp[0] === spielerID) // Eigener Token
650 {
651 clearInterval(intervall); // Intervall entfernen, da Spieler mit Spielzug dran ist
652 grund = "spielzug"; // nächste serverAnfrage hat den grund für spielzug
653
654 //console.log("A"+temp[1]+"B"); // Testausgabe
655 if (temp[1] !== "") // Abfangen vom Erstzug von Spieler1
656 {
657 let position = -1; // für das versunkene Schiff im Array der "schifflange"
658
659 //console.log("Schiffe: " + schiffe); // Testausgabe
660 for (let schiff in schiffe) // foreach JS Version
661 {
662 // "schiff" wird mit Koordinate gefiltert, falls vorhanden
663 //console.log("Schiffindex: "+schiffe.indexOf(schiff) + " hat die Länge " +
schiffe[schiff].length); // Testausgabe
664 if (schiffe[schiff].includes(temp[1])) // Überprüfung ob gegnerischer
Spielzug trifft
665 {
666 schiffe[schiff] = schiffe[schiff].filter(e => e !== temp[1]); //
entfernen der beschossenen Fläche
667 if (schiffe[schiff].length === 0) // keine beschießbaren Flächen mehr
668 {
669 // Treffer merken welches Schiff !!!
670 position = schiff; // setzen der Position des versunkenen Schiffs
671 //console.log("Eigenes Schiff verloren."); // Testausgabe
672 }
673 //console.log("Schiff: "); // Testausgabe
674 //console.log(schiffe[schiff]); // Testausgabe
675 }
676 }
677
678 /* Testausgaben
679 //console.log("Schiffe: ");
680 //console.log(schiffe);
681 //console.log("Position (Array): " + position);
682 //console.log("Länge des Schiffes: " + schifflange[position]);
683 */
684
685 if (-1 < position) // wenn Position gesetzt (Schiff versunken)
686 {
```

```
687 switch (schifflange[position]) // Meldung über Art des Schiffes +
Aktualisierung der Schiffanzeige
688 {
689 case 2:
690 document.getElementById("status").innerHTML = "Eigenes U-Boot wurde
versenkt!";
691 document.getElementById("schiff2").innerHTML = parseInt(document.
getElementById("schiff2").innerHTML) - 1;
692 break;
693 case 3:
694 document.getElementById("status").innerHTML = "Eigener Zerstörer
wurde versenkt!";
695 document.getElementById("schiff3").innerHTML = parseInt(document.
getElementById("schiff3").innerHTML) - 1;
696 break;
697 case 4:
698 document.getElementById("status").innerHTML = "Eigener Kreuzer wurde
versenkt!";
699 document.getElementById("schiff4").innerHTML = parseInt(document.
getElementById("schiff4").innerHTML) - 1;
700 break;
701 case 5:
702 document.getElementById("status").innerHTML = "Eigenes Schlachtschiff
wurde versenkt!";
703 document.getElementById("schiff5").innerHTML = parseInt(document.
getElementById("schiff5").innerHTML) - 1;
704 }
705 }
706
707 //console.log("Schifflänge: " + schifflange); // Testausgabe
708 //console.log("Schiff verloren: " + schifflange[index]); // Testausgabe
709
710 // gegnerischer Spielzug verarbeiten
711 if (document.getElementById("spieler." + temp[1]).innerHTML === "W")
712 {
713 document.getElementById("spieler." + temp[1]).style.backgroundColor = "yellow"
; // wenn Schuss auf Wasser, dann Gelb
714 document.getElementById("status").innerHTML = "Gegnerischer Schuss ging ins
Wasser."; // Meldung, dass Schuss daneben
715 }
716 else if (document.getElementById("spieler." + temp[1]).innerHTML === "O")
717 {
718 document.getElementById("spieler." + temp[1]).style.backgroundColor = "red";
// wenn Treffer eines Schiffes, dann Rot
719 }
720 document.getElementById("spieler." + temp[1]).innerHTML = "X"; // Treffer
markieren auf eigenem Spielfeld
721 }
722 if (document.getElementById("schiff2").innerHTML === "0" && document.getElementById(
"schiff3").innerHTML === "0" && document.getElementById("schiff4").innerHTML === "0"
&& document.getElementById("schiff5").innerHTML === "0")
723 {
724 aktionEntfernen(); // Entfernen des weiteren Beschusses
```

```
725 statistikausgabe(); // Statistik unter Schiffanzeige platzieren
726 setTimeout(verlorenMeldung, 200); // Meldung über Niederlage (200 ms Verzögerung)
727 }
728 else
729 {
730 setTimeout(erwarten, abfrageZeit); // verzögerte Statusmeldung, dass Spielzug
erfolgen soll (5 Sekunden Verzögerung)
731 }
732 }
733 }
734
735 function verlorenMeldung() // verzögerte Meldung, dass verloren
736 {
737 document.getElementById("status").innerHTML = "Alle Schiffe verloren!";
738 alert("Spiel verloren!");
739 }
740
741 function erwarten() // Statusmeldung
742 {
743 document.getElementById("status").innerHTML = "Spielzug erwartet."; // Statusmeldung,
dass Spieler Zug tätigen soll
744 }
745
746 // Logik der Spielzüge
747
748 function spielzug() // beim Klick auf gegnerisches Spielfeld
749 {
750 if (document.getElementById(this.id).innerHTML === "?") // unbekanntes Feld
751 {
752 let id = this.id;
753 //console.log(id); // Testausgabe
754 if (grund === "spielzug") // nur wenn Spieler dran ist
755 {
756 zuganzahl++;
757 let temp = id.split("."); // gegnerID aufteilen
758 eintrag = temp[1]; // Feldkoordinate des gegnerischen Spielfeldes (ID)
759 document.getElementById(id).innerHTML = "X";
760 serverAnfrage();
761 intervall = setInterval(serverAnfrage, abfrageZeit);
762 }
763 else // wartet noch auf eigenen Token
764 {
765 alert("Gegner ist dran!"); // Meldung an Spieler
766 }
767 }
768 else
769 {
770 alert("Feld bereits beschossen!"); // Meldung an Spieler
771 }
772 }
773
774 function sendeSpielzug()
775 {
```

```
776 //console.log("Spielzug wird gesendet."); // Testausgabe
777 let formDaten = new FormData();
778
779 formDaten.append("spielerID", spielerID); // übergeben ID
780 formDaten.append("spielzug", eintrag); // übergeben Spielzug
781
782 // Anfrage wird gesendet
783 xhttp.send(formDaten);
784 }
785
786 function antwortbearbeitungSendeSpielzug(antwort) // Rückmeldung, ob (W)asser, (T)reffer,
(V)ersenkt oder sogar (G)ewonnen
787 {
788 treffer++; // Treffer erhöhen -> Code sparen
789 switch (antwort)
790 {
791 case "W":
792 document.getElementById("status").innerHTML = "Schuss ging ins Wasser."; //
Statusmeldung -> (W)asser!
793 document.getElementById("gegner." + eintrag).innerHTML = "W"; // Wasserzeichen
794 document.getElementById("gegner." + eintrag).style.backgroundColor = "aqua"; //
Wasserfarbe
795 treffer--; // Treffer reduzieren, weil daneben
796 break;
797 case "T":
798 document.getElementById("status").innerHTML = "Gegnerisches Schiff getroffen!";
// Statusmeldung -> (T)reffer!
799 document.getElementById("gegner." + eintrag).innerHTML = "T"; // Trefferzeichen
800 document.getElementById("gegner." + eintrag).style.backgroundColor = "red"; //
Treffermarkierung
801 break;
802 case "V":
803 document.getElementById("status").innerHTML = "Gegnerisches Schiff versenkt!"; //
Statusmeldung -> (V)ersenkt!
804 document.getElementById("gegner." + eintrag).innerHTML = "V"; // Versenktzeichen
805 document.getElementById("gegner." + eintrag).style.backgroundColor = "red"; //
Treffermarkierung
806 break;
807 case "G":
808 document.getElementById("status").innerHTML = "Alle gegnerischen Schiffe wurden
versenkt!"; // Statusmeldung -> (G)ewonnen!
809 document.getElementById("gegner." + eintrag).innerHTML = "V"; // Versenktzeichen
810 document.getElementById("gegner." + eintrag).style.backgroundColor = "red"; //
Treffermarkierung
811 aufdecken(); // restliche Felder mit Wasser befüllen
812 aktionEntfernen(); // Eventlistener entfernen vom gegnerischen Spielfeld
813 statistikausgabe(); // Statistik unter Schiffanzeige platzieren
814 clearInterval(intervall); // Stop der Anfragen, da Spiel vorbei.
815 setTimeout(meldungSieg, 200); // damit Meldung erst nach der Eintragung auftritt
(200ms Verzögerung)
816 //console.log("Spielende."); // Testausgabe
817 }
818 if (antwort !== "G") // wenn Spiel weitergeht
```

```
819 {
820 // Rückmeldung des Servers bezüglich (W)asser / (T)reffer / (V)ersenkt / (G)ewonnen
verarbeiten
821 grund = "token"; // danach die nächste serverAnfrage nach token
822 setTimeout(meldungWarten, abfrageZeit); // damit Statuseldung erst nach der
Statusmeldung des Spielzuges auftritt (5000ms Verzögerung)
823
824 //console.log("antwortbearbeitungSendeSpielzug: " + antwort); // Testausgabe
825 }
826 }
827
828 function meldungSieg() // Verzögerte Meldung des Sieges damit Markierung zuerst erfolgt
829 {
830 alert("Spiel gewonnen!");
831 }
832
833 function meldungWarten()
834 {
835 document.getElementById("status").innerHTML = "Warten auf gegnerischen Spielzug..."; //
Statusmeldung -> Spieler ist nicht dran
836 }
837
838 function aufdecken() // alle restlichen Wasserfelder anzeigen (Spielfeld -> Gegner)
839 {
840 //console.log("Wasser wird verschüttet..."); // Testausgabe
841 index = 'a'; // Zurücksetzen des Indexes
842
843 // Für jede Reihe ( A - J )
844 for (let zahlReihe = 1; zahlReihe <= anzahl; zahlReihe++)
845 {
846 // jeweilige Reihe
847 for (let zahl = 1; zahl <= anzahl; zahl++)
848 {
849 //console.log("Index: "+index); // Testausgabe
850 let idFeld = "gegner." + index + zahl;
851 //console.log(idFeld); // id-Ausgabe zum Test
852 if (document.getElementById(idFeld).innerHTML === "?")
853 {
854 document.getElementById(idFeld).innerHTML = "W"; // Wasserzeichen
855 document.getElementById(idFeld).style.backgroundColor = "aqua";
856 }
857 }
858 index = String.fromCharCode(index.charCodeAt(0) + 1); // nächster Buchstabe
859 }
860 }
861
862 function aktionEntfernen() // Eventlistener für gegnerisches Spielfeld entfernen
863 {
864 //console.log("Spieleraktionen beenden..."); // Testausgabe
865 index = 'a'; // Zurücksetzen des Indexes
866
867 // Für jede Reihe ( A - J )
868 for (let zahlReihe = 1; zahlReihe <= anzahl; zahlReihe++)
```

```
869 {
870 // jeweilige Reihe
871 for (let zahl = 1; zahl <= anzahl; zahl++)
872 {
873 //console.log("Index: "+index); // Testausgabe
874 let idFeld = "gegner." + index + zahl;
875 //console.log("idFeld remove: " + idFeld); // id-Ausgabe zum Test
876 document.getElementById(idFeld).removeEventListener("click", spielzug);
877 //console.log(auswahl); // Testausgabe
878 }
879 index = String.fromCharCode(index.charCodeAt(0) + 1); // nächster Buchstabe
880 }
881 }
882
883 function statistikausgabe() // Eintrag in Statistikbereich unter der Schiffanzeige
884 {
885 let ausgabe = "<br><br>" + treffer + " Treffer von " + zuganzahl + " Schüssen<br>"; // x
Treffer von y Schüssen
886 ausgabe += "<br>Eigene Trefferquote: " + parseInt(100 * treffer / zuganzahl) + " %"; //
ganze % ohne Kommastellen
887
888 document.getElementById("statistik").innerHTML = ausgabe;
889 }
890
```

10.6.4 script.php

```
1 <?php
2 #print_r($_POST); # Testausgabe der übergebenen Werte
3
4 $datei = "spiel.json";
5
6 /* for($i = 0; $i<100; $i++)
7 {
8 echo rand(0,1); # Testversuch für Zufallszahl
9 } */
10
11 if (isset($_POST["aufstellung"])) # Spieler will sich registrieren
12 {
13 registrierung($datei);
14 }
15 else if (isset($_POST["spielzug"])) # Spielzug setzen
16 {
17 #echo "Spielzug: ".$_POST["spielzug"]."\n"; # Testausgabe
18 $spieldaten = file_get_contents($datei); # Dateiauslesen
19 $json = json_decode($spieldaten); # JSON String decodieren -> Objekt
20 $spiel = new Spiel($json->spieler1, $json->spieler2, $json->token, $json->aufstellung1
, $json->aufstellung2); # Spielobjekt neu erzeugt mit Parametern
21 $check = explode(";", $spiel->token);
22
23 if ($check[0] === "Spieler1") # Spieler1 dran -> Tokenwechsel
24 {
25 #echo $_POST["spielzug"].$_POST["spielerID"]."???\n"; # Testausgabe
26 #print_r($array); # Testausgabe
27 #echo $_POST["spielzug"]; # Testausgabe
28 $spiel->token = "Spieler2"; # Wechsel des Tokens an ID 2. Spieler
29 }
30 else if ($check[0] === "Spieler2") # Spieler2 dran -> Tokenwechsel
31 {
32 #echo $_POST["spielzug"].$_POST["spielerID"]."!!!\n"; # Testausgabe
33 #print_r($array); # Testausgabe
34 #echo $_POST["spielzug"]; # Testausgabe
35 $spiel->token = "Spieler1"; # Wechsel des Tokens an ID 1. Spieler
36 }
37
38 $auswertung = auswerten($_POST["spielerID"], $_POST["spielzug"], $spiel, $datei); #
Arraybearbeitung
39 $spiel->token .= ";" . $_POST["spielzug"];
40 echo $auswertung;
41 #echo "Spielzug erfolgt."; # Testausgabe
42 # Auswertung ob (W)asser / (T)reffer / (V)ersenkt
43
44 $json = json_encode($spiel); # JSON String generieren
45 file_put_contents($datei, $json); # in Datei schreiben
46 }
47 else if (isset($_POST["spielerID"])) # Tokenabfrage
48 {
```

```
49 $spieldaten = file_get_contents($datei); # Dateiauslesen
50 $json = json_decode($spieldaten); # JSON String decodieren -> Objekt
51 echo $json->token; # Token zurückgeben (SpielerID;Spielzug)
52 }
53 else # Spieler1 will Spieler2 Namen
54 {
55 $spieldaten = file_get_contents($datei); # Dateiauslesen
56 $json = json_decode($spieldaten); # JSON String decodieren -> Objekt
57 echo $json->spieler2; # Spieler2 zurückgeben (Namen)
58 }
59
60 function registrierung($datei) # Registrierung in spiel.json - Datei
61 {
62 if (!file_exists($datei)) # Datei existiert nicht -> Spieler1 mit Aufstellung1
reinschreiben
63 {
64 $json = json_decode($_POST["aufstellung"]); # JSON String decodieren -> Array
65 $spiel = new Spiel($_POST["spieler"], null, null, $json, null); # Objekt Spiel
erzeugen mit Parametern
66 $json = json_encode($spiel); # JSON String generieren
67 #print_r($spiel); # Testausgabe
68 file_put_contents($datei, $json); # in Datei schreiben
69 echo "Spieler1;registriert."; # Ausgabe für JS
70 }
71 else # Spieler2 will sich registrieren
72 {
73 $spieldaten = file_get_contents($datei); # Dateiauslesen
74 $json = json_decode($spieldaten); # JSON String decodieren -> Objekt
75
76 # echo "Spieler1: ".$json->spieler1."\n"; # Wertzugriff mit Schlüssel ->
Testausgabe
77 $spiel = new Spiel($json->spieler1, $json->spieler2, $json->token, $json->
aufstellung1, $json->aufstellung2); # Spielobjekt neu erzeugt mit Parametern
78
79 if ($json->spieler2 !== null)
80 {
81 echo "Spiel bereits im Gange."; # Ausgabe für JS
82 }
83 else
84 {
85 $spiel->spieler2 = $_POST["spieler"];
86 $json = json_decode($_POST["aufstellung"]); # String in Array umwandeln
87 $spiel->aufstellung2 = $json; # Array eintragen
88 #print_r($spiel); # Testausgabe
89 echo "Spieler2;registriert.;Gegner;"; # Ausgabe für JS, getrennt durch ";"
90 echo $spiel->spieler1 . ";"; # Spieler 1 Namen zurückgeben
91 #$spiel->token = "test123"; # Testeintrag in JSON
92
93 if (rand(0, 1) === 0) # Zufallsentscheid, wer anfängt von 0 bis 1 (Ganzzahl)
94 {
95 $spiel->token = "Spieler1"; # ID 1. Spieler
96 }
97 else
```

```
98 {
99 $spiel->token = "Spieler2"; # ID 2. Spieler
100 }
101 $spiel->token .= ";"; # danach kommt letzter gemachter Zug, bei Erstzug leer
102
103 echo $spiel->token; # Rückgabe des Tokens
104 $json = json_encode($spiel); # JSON String generieren
105 file_put_contents($datei, $json); # in Datei schreiben
106 #echo rand(0,1); # Spieler bestimmen der anfängt
107 }
108 }
109 }
110
111 class Spiel # Vorlage für die Speicherung in JSON-Format
112 {
113 public $spieler1;
114 public $spieler2;
115 public $token; # 1. Stelle: Spieler der dran ist; 2. Stelle: letzter Spielzug
("Spieler1" oder "Spieler2" -> IDs)
116 public $aufstellung1;
117 public $aufstellung2;
118
119 function __construct($spieler1, $spieler2, $token, $aufstellung1, $aufstellung2)
120 {
121 $this->spieler1 = $spieler1; # Spielername
122 $this->spieler2 = $spieler2; # Spielername
123 $this->token = $token;
124 $this->aufstellung1 = $aufstellung1; # Schiffe
125 $this->aufstellung2 = $aufstellung2; # Schiffe
126 }
127 }
128
129 function auswerten($SID, $spielzug, $spiel, $datei) # SID = spielerID
130 {
131 # Logik mit return der Meldung
132 #echo "Spielzug: ".$spielzug; # Testausgabe
133 #print_r($schiffsammlung); # Testausgabe
134 $status = "W"; # (W)asser
135 $schiffsammlung;
136
137 if ($SID === "Spieler1") # Spieler1 schießt auf Aufstellung von Spieler2
138 {
139 $schiffsammlung = $spiel->aufstellung2;
140 }
141 else # Spieler2 schießt auf Aufstellung von Spieler1
142 {
143 $schiffsammlung = $spiel->aufstellung1;
144 }
145
146 $leer = true;
147
148 foreach ($schiffsammlung as &$schiff) # & => Referenz
149 {
```

```
150 foreach ($schiff as &$feld) # & => Referenz
151 {
152 # prüfen ob getroffen; übergebe Array-Index -> $treffer
153 if (($treffer = array_search($spielzug, $schiff)) !== false)
154 {
155 #print_r($schiff); # Testausgabe
156 #echo "Treffer: ".$treffer; # Testazsgabe
157 unset($schiff[$treffer]); # Feld aus Schiff-Array nehmen
158 $schiff = array_values($schiff); # Entfernen von Index (alle)
159 $status = "T"; # (T)reffer
160
161 if (count($schiff) === 0) # prüfen ob keine mehr beschießbaren Felder
162 {
163 $status = "V"; # (V)ersenkt
164 }
165 #print_r($schiff); # Testausgabe
166 }
167 }
168 if ($leer) # solange leer ist überprüfe, ob Schiff leer
169 {
170 $leer = empty($schiff); # nicht leeres Schiff gefunden
171 }
172 #print_r($schiffsammlung); # Testausgabe
173 }
174
175 if ($leer)
176 {
177 $status = "G"; # (G)ewonnen
178 }
179
180 if ($SID === "Spieler1") # Aktualisierung der Schiffsammlung Spieler2
181 {
182 $spiel->aufstellung2 = $schiffsammlung;
183 }
184 else # Aktualisierung der Schiffsammlung Spieler1
185 {
186 $spiel->aufstellung1 = $schiffsammlung;
187 }
188
189 $json = json_encode($spiel); # JSON String generieren
190 file_put_contents($datei, $json); # in Datei schreiben
191
192 return $status;
193 }
194 ?>
195
```

10.7 Abnahmeprotokoll

IHK Abschlussprüfung Abnahme-Protokoll Denis Ojdanic

1.	Webseite zum Spiel wird geladen	☑
2.	Spieler gegen Spieler möglich	☑
3.	Über einen Webserver spielbar	☑
4.	Statistik für Züge, Trefferquoten & verlorene Schiffe	☑
5.	Spielfeldgröße 10 x 10	☑
6.	Zufällige Spielerauswahl für Spielstart	☑
7.	Schiffe dürfen nicht aneinanderstoßen	☑
8.	Schiffe müssen gerade verlaufen	☑
9.	Platzierung der Schiffe am Spielfeldrand möglich	☑
10.	Feldgröße der Spielfelder dynamisch	☑
11.	Bestimmte Anzahl der Schiffe	☑
12.	Aktualisierung der Schiffanzeige	☑

27.07.23
Datum

Ojdanic
Auszubildender

T. Müller
Ausbilder
~~Tim~~ Klaus Müller

-> 10. Auflösung des Fenster beeinflusst das Spielfeld
-> 12. Anzahl der Schiffe wird angezeigt
-> 3. auch auf remote Server (extern gehostet)