Schweizerische
Fachschule
TEKO

Studiengang Informatik

Vertiefungsrichtung Applikationsentwicklung

Diplomarbeit

von **Paulo Ribeiro** mit dem Titel

Spieleentwicklung auf Website

Semester:	Sommersemester 2022
Betreuer:	Herr Benjamin Bäni, Dozent
Abgabedatum:	24. Oktober 2022

Version 1.0

Historie der Dokumentversionen

Version	Datum	Autor	Änderungsgrund / Bemerkungen
0.1	24.09.2022	Paulo Ribeiro	Ersterstellung
0.2	27.09.2022	Paulo Ribeiro	Titelblatt, Inhaltsverzeichnis erstellt
0.3	02.10.2022	Paulo Ribeiro	Dokumentation der programmierte Spiele
0.4	10.10.2022	Paulo Ribeiro	Dokumentation der Initialisierung, Planung und Realisierung
0.5	16.10.2022	Paulo Ribeiro	Erstellung Abbildungsverzeichnis, Tabellenverzeichnis
0.6	16.10.2022	Paulo Ribeiro	Dokumentation der Lessons Learned, Reflexion, Problemstellungen
1.0	19.10.2022	Paulo Ribeiro	Kontrolle, Anpassung, Korrektur und Fertigstellung der Dokumentation

Tabelle 1 Historie der Dokumentversionen

Management Summary

Als Abschluss der dreijährigen Ausbildung als Dipl. Techniker HF Informatik Fachrichtung Applikationsentwicklung erhielt ich den Auftrag von der TEKO Schweizerische Fachschule Olten eine praktische Diplomarbeit zu schreiben und implementieren. Für die Diplomarbeit wird das Erlernte der dreijährigen Ausbildung ausgeübt und überprüft. Dabei werden verschiedene Methoden angewendet, um die praktische Diplomarbeit auszuführen.

Die vorliegende Diplomarbeit befasst sich mit der Problemstellung der Programmierung von Videospielen, eine Website, ein Backend und diese miteinander zu verbinden. Während dieser Programmierung werden die entsprechenden verwendeten Methoden und Programmcodes dokumentiert.

Als Endresultat erhielt ich eine laufende HTML Website, die innerhalb von maximal fünf Sekunden Ladezeit geladen wird, einen strukturierten Aufbau und einfaches Design besitzt. Drei Videospielen, die fehlerfrei spielbar sind und mit der Programmiersprache JavaScript programmiert wurden. Ein Backend, der mittels Node JS und dem Hauptmodul Express gestartet werden kann, sowie eine ausführliche Dokumentation der Diplomarbeit.

Das Backend beinhaltet die Verbindung zu der Datenbank MongoDB und die Funktionen für das Aufstarten des lokalen Servers, die Kontaktaufnahme und die Anmeldung und Log-in des Benutzers. In der MongoDB Datenbank werden die Anmeldedaten der registrierten Benutzer gespeichert und sobald diese sich einloggen möchten, abgerufen. Nach erfolgreiches Log-in werden die Benutzer in einen Mitgliedsbereich umgeleitet.

Inhaltsverzeichnis

1 Einleitung

1.1 Paulo als Diplomat

Mein beruflicher Werdegang startete ich 2015 mit der Ausbildung zum Produktionsmechaniker EFZ, welchen ich 2018 abschloss. Bis 2019 arbeitete ich als Produktionsmechaniker in verschiedenen Unternehmen.

Im Oktober 2019 begann ich meine HF zum Techniker HF Informatik Applikationsentwicklung an der TEKO in Olten. Während dieser Zeit arbeitete ich in weiteren verschiedenen Unternehmen als temporärer Angestellter und wechselte in die Informatik.

Seit Januar 2022 bin ich von der Univativ Schweiz AG angestellt und im Universitätsspital Basel tätig, auch temporär.

1.2 Rollenverteilung

1.2.1 Diplomand

Der Diplomand und gleichzeitiger Ersteller dieser Arbeit ist **Paulo Ribeiro**.

Paulo Ribeiro
n.rib.paulo@gmail.com

1.2.2 Diplomlehrer

Der Diplomlehrer für diese Arbeit Seitens TEKO ist **Benjamin Bäni**.
Herr Bäni ist für die Betreuung von drei Studierenden zuständig.

Benjamin Bäni
b.baeni@gmx.ch

2 Aufgabe

Mit der Diplomarbeit werde ich das während des gesamten Studium aufgebaute Grundlagenwissen an einer praxisorientierten Aufgabe in meiner Studienrichtung anwenden und vertiefen. Dabei geht es für mich im Wesentlichen um die konsequente Zielorientierung, die Anwendung der Phasen des Projektmanagements sowie die konsequente Umsetzung von definierten Rahmenbedingungen.

Als Applikationsentwickler entschied ich mich, eine Website mit drei selbst programmierte Spiele zu entwickeln, ein Backend für die Website und diese mit einer funktionstüchtigen Datenbank verbinden.

Die drei selbst programmierten Spiele sind:

- Rock, Paper, Scissors (Schere, Stein, Papier)
- Tic-Tac-Toe (Drei gewinnt)
- Pong

Für die Programmierung der Spiele werden keine Frameworks verwendet, sie sollen in «Vanilla» JavaScript programmiert werden. Sobald die Spiele programmiert wurden, werden diese in die Website integriert.

Durch die Anwendung der Programmiersprache HTML wird die Website erstellt und mit CSS in einem passendem, modernes Design gestylt.

Zusätzlich wird ein Backend programmiert, mit einer Datenbank namens MongoDB. Dies wird mittels Node.js erreicht. Die Verbindung zwischen MongoDB und Node.js erlaubt es, Daten zu speichern und abzurufen, daher wird auch eine Anmeldung und Log-In für die Spieler programmiert.

Alle verwendeten Methoden, um die Ziele zu erreichen, werden gründlich in der Dokumentation dokumentiert, sowie die Erstellung der drei Spiele, die Datenbank MongoDB und das Backend.

3 Zielbestimmung

Für die Diplomarbeit bestimmte ich selbst meine eigenen Ziele und Erfolgskriterien. Um diese Ziele und Erfolgskriterien klar zu definieren, verwendete ich das erlernte vom Fach Projektmanagement. Ich erstellte die Musskriterien, Wunschkriterien, Abgrenzungskriterien und die Zielscheibe. Zusätzlich bekam jeder Diplomand von seinem Diplomlehrer eine Aufgabenstellung. In dieser Aufgabenstellung befinden sich weitere besondere Punkte, die beachtet werden müssen. Diese Punkte werden hier auch als Zielbestimmung aufgelistet.

- Die einzelnen Schritte, welche als Summe zum Ergebnis geführt haben, müssen beschrieben sein.
- Wurden im Rahmen der Lösungsfindung verschiedene Varianten entworfen und geprüft, muss die Evaluation der gewählten Variante mittels einer dazu geeigneten Methode dokumentiert und begründet sein.
- Bei der Dokumentation der Diplomarbeit ist auf einen logischen Aufbau, eine saubere Gliederung sowie auf gute Verständlichkeit zu achten. Details sind in den Richtlinien zur Diplomarbeit beschrieben.
- Der Value-add (Mehrwert) des Ergebnisses für den Auftraggeber muss aus der Dokumentation klar erkennbar und nachvollziehbar sein.
- Für die optimale Begleitung muss dem Diplomlehrer ein wöchentlicher Statusbericht (gemässe Vorlage Extranet TEKO) zugestellt werden. Dies ermöglicht dem Diplomlehrer den Verlauf unmittelbar mit- verfolgen zu können und bei Bedarf gezielt zu reagieren.
- Der Fachexperte muss die Diplomarbeit gemäss den Kriterien auf dem durch den Diplomlehrer abgegebenen Bewertungsformular beurteilen. Der Fachexperte sendet das Formular bis spätestens eine Woche nach Abgabe der Diplomarbeit an den Diplomlehrer.
- Die Präsentation der Diplomarbeit muss so ausgestaltet sein, dass eine aussenstehende, fachunabhängige Person innerhalb von 15 Minuten die folgenden Punkte erkennen und bewerten kann:
  - Auftrag sowie Sinn und Zweck
  - Endergebnisse und Erfolgskriterien
  - besondere Herausforderungen auf dem Weg zum Ziel inkl. gewählte Lösungsansätze mit Begründung, jedoch ohne sich in Details zu verlieren
  - Demo des Produktes, Prototyps etc.
  - Reflexion: Zielerreichung, Weg zum Ziel, lessons learnt
- Im Anschluss an die Präsentation des Ergebnisses muss innerhalb von 5 Minuten die Onlinepublikation der Diplomarbeit vorgestellt werden. Die Onlinepublikation wird bewertet und muss so ausgestaltet sein, dass Aussenstehende in einer attraktiven, kreativen und leicht verständlichen Form sich über die erarbeitete Diplomarbeit informieren können.

3.1 Musskriterien

- Der Benutzer
  - ❖ hat Zugriff auf die Website (Website abrufen).
  - ❖ kann die Inhalte der Website ansehen.
  - ❖ kann die Spiele spielen.

- Die Website
  - ❖ wird strukturiert aufgebaut.
  - ❖ funktioniert über Localhost (lokalen Host).
  - ❖ lädt innerhalb von maximal fünf Sekunden.
  - ❖ besitzt eine einfache, klare, gestaltende Navigation/Navigationsmenü.
  - ❖ besitzt ein Impressum, Datenschutz Erklärung und Kontaktformular.

- Die Spiele
  - ❖ sind: Rock, Paper, Scissors / Tic-Tac-Toe und Pong.
  - ❖ besitzen je eine Spielanleitung.
  - ❖ besitzen einen Computergegner.
  - ❖ wurden mittels Vanilla JavaScript programmiert.
  - ❖ zeigen den aktuellen Highscore (Punktestand)
  - ❖ funktionieren fehlerfrei.

- Sonstiges
  - ❖ Inhaltssprache der Website auf Deutsch.
  - ❖ Inhaltssprache der Spiele auf Englisch.

3.2 Wunschkriterien

- Die Website
  - ❖ läuft über Node.js und Express (lokalen Host).
  - ❖ ermöglicht es, dass Benutzer ein Konto eröffnen können. Die Daten für den Login werden über Node.js, Express in der MongoDB Datenbank gespeichert und abgerufen

- Die Spiele
  - ❖ (sofern der Benutzer ein Konto hat) speichert den Highscore des Benutzers auf der MongoDB Datenbank.

3.3 Abgrenzungskriterien

- Die Website läuft lokal auf dem Notebook und ist nicht öffentlich zugänglich.
- Die Spiele besitzen einen Computergegner. Es gibt keine Schwierigkeitsstufe.
- Website mittels HTML programmiert und CSS gestylt, die Spiele mittels JavaScript programmiert. Als «Framework» wird Node.js verwendet mit dem «Express» Modul. Für die Datenbank wird MongoDB verwendet.
- Es werden keine weiteren Inhaltssprachen hinzugefügt.

3.4 Zielscheibe

Richtziel:

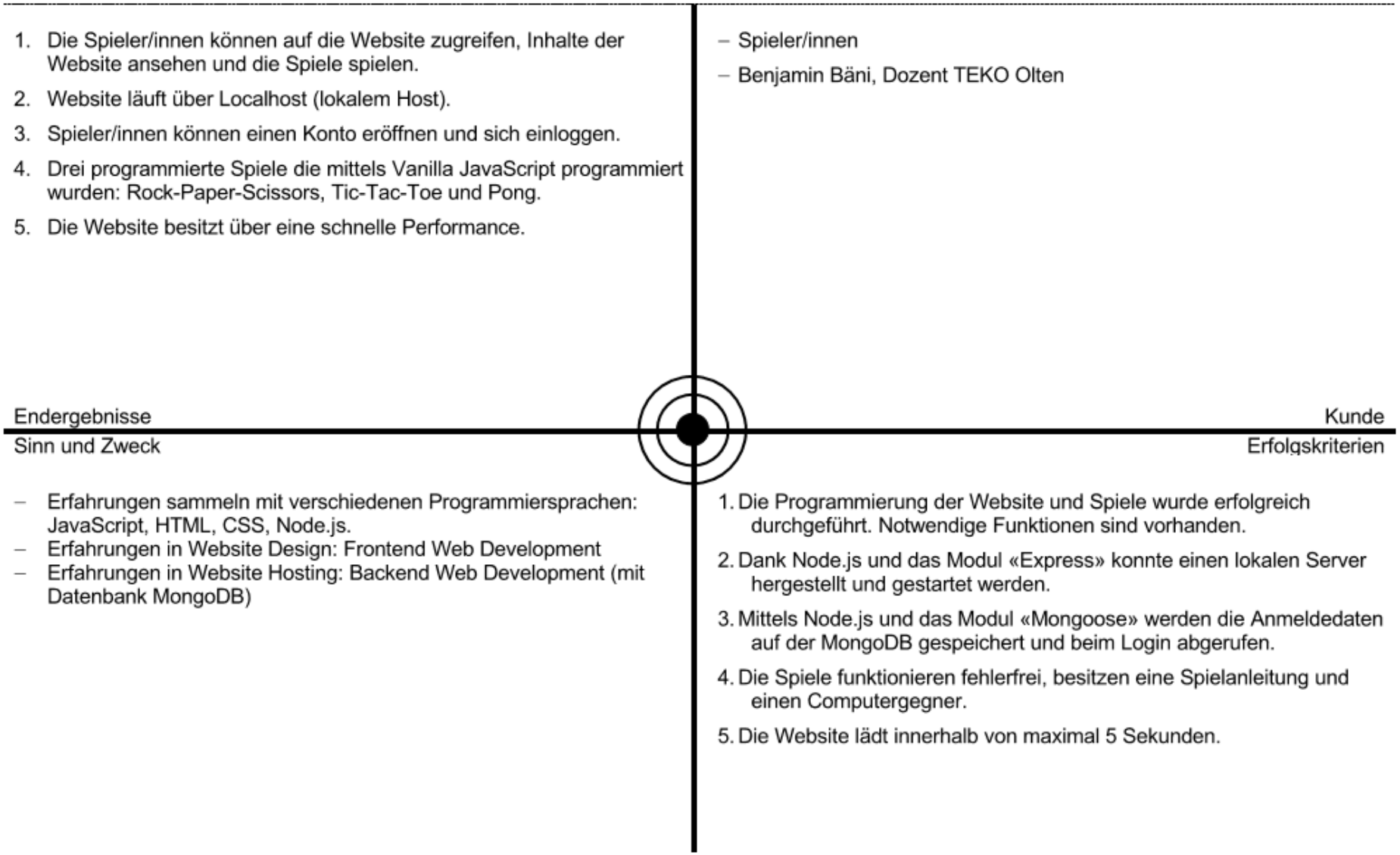


Abbildung 1 Zielscheibe

4 Planungsphase

Wie bei der Zielbestimmung, wird auch hier das erlernte vom Fach Projektmanagement angewendet. Für die Erstellung der Planungsphase, erstellte ich zwei Projektpläne, die Projektstrukturplanung und die Projektablaufplanung.

Die Projektstrukturplanung stellt die verschiedenen Phasen, von der Initialisierung bis zum Abschluss, dar.

Bei der Projektablaufplanung wurden wichtige Aufgaben/Tätigkeiten aufgelistet, die in einem bestimmtem Zeitraum erfüllt werden müssen. In der Projektablaufplanung sieht man den Fortschritt der einzelnen Tätigkeiten (Status), Beginn- und Enddatum, die Meilensteine, sowie weitere hilfreiche Angaben.

4.1 Projektstrukturplanung

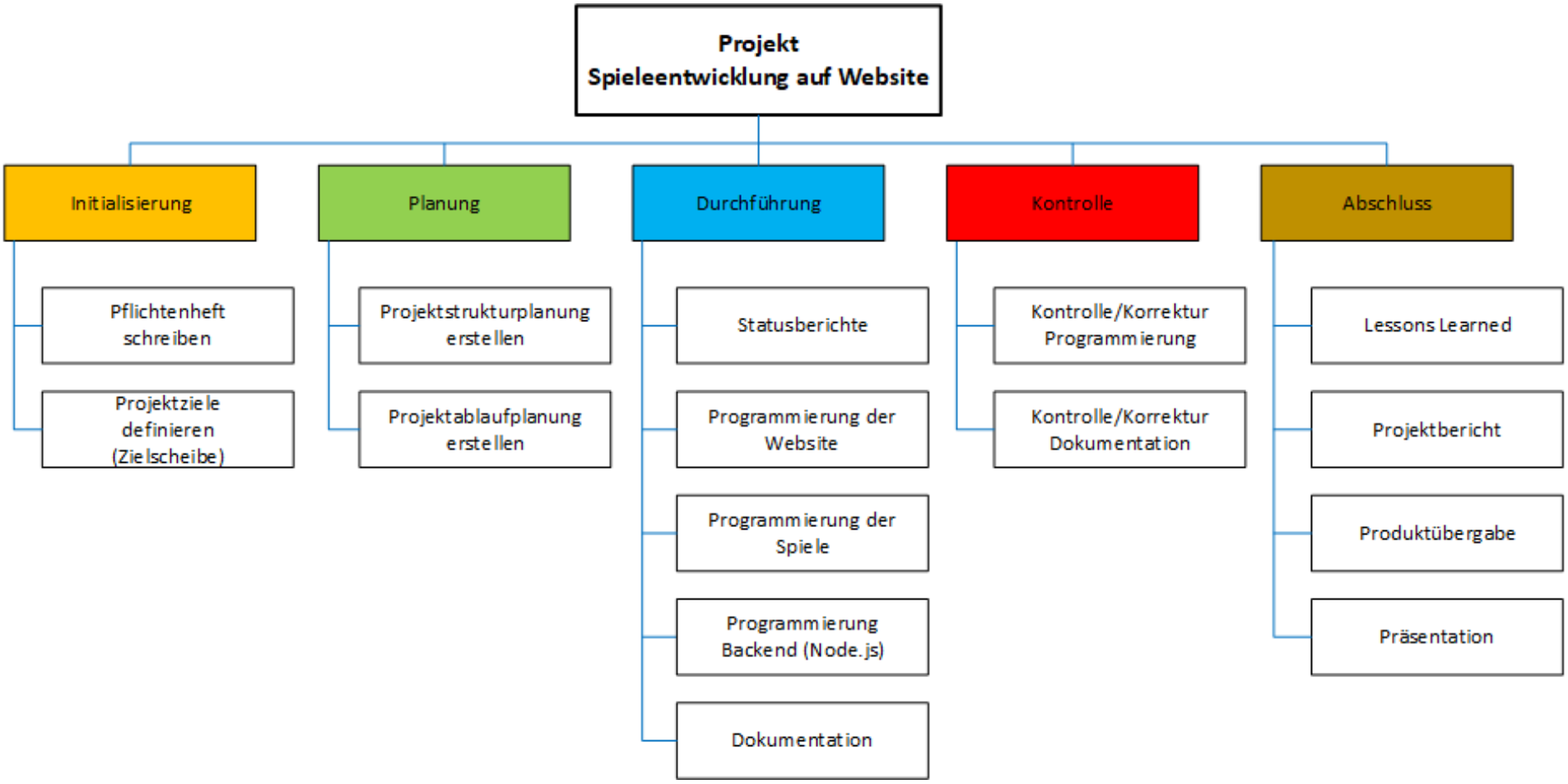


Abbildung 2 Projektstrukturplanung

4.2 Projektablaufplanung

Projektablaufplanung

Projekt	Spieleentwicklung auf Website
Projektnummer	DA_P1
Projektleiter/in	Paulo Ribeiro
Datum	12.09.2022
Projektinformationen	Programmierung von Website (HTML, CSS, JS), drei Spiele (Vanilla JavaScript) & Backend (Node.js und Modul "Express")
Stand:	*18.10.2022*

Meilenstein

Pos.	Aufgabe/Tätigkeit	Beginn [Datum]	Ende nach [Tage]	Ende [Datum]	Status [0 bis 100]	Meilenstein [Datum]	Verantwortlich [Name]	Unterstützung [Name]
1	Projektziele definieren (Zielscheibe)	12.09.2022	2	13.09.2022	0%		Paulo Ribeiro	Benjamin Bäni
2	Projektstrukturplanung erstellen	12.09.2022	2	13.09.2022	0%		Paulo Ribeiro	Benjamin Bäni
3	Projektablaufplanung erstellen	12.09.2022	2	13.09.2022	0%		Paulo Ribeiro	Benjamin Bäni
4	Software installieren (VSCode, Node.js, etc.)	12.09.2022	2	13.09.2022	0%		Paulo Ribeiro	
5	Node.js und Modul: Express recherchieren	14.09.2022	4	17.09.2022	0%		Paulo Ribeiro	
6	Statusbericht 1	18.09.2022	1	18.09.2022	0%		Paulo Ribeiro	Benjamin Bäni
7	Lokalhost Programmierung mit Node.js und Modul Express	18.09.2022	2	19.09.2022	0%		Paulo Ribeiro	
8	Frontpage der Website programmieren	19.09.2022	2	20.09.2022	0%		Paulo Ribeiro	
9	Frontpage Styling	19.09.2022	2	20.09.2022	0%		Paulo Ribeiro	
10	Spiele: Pong, Tic-Tac-Toe und Rock, Paper, Scissors Webseiten programmieren und styling	21.09.2022	1	21.09.2022	0%		Paulo Ribeiro	
11	1. Vorzeigetermin	22.09.2022	1	22.09.2022	0%		Paulo Ribeiro	Benjamin Bäni
12	Spiel Pong programmieren	22.09.2022	3	24.09.2022	0%		Paulo Ribeiro	
13	Statusbericht 2	25.09.2022	1	25.09.2022	0%		Paulo Ribeiro	Benjamin Bäni
14	Spiel Tic-Tac-Toe programmieren	25.09.2022	3	27.09.2022	0%		Paulo Ribeiro	
15	Spiel Rock, Paper, Scissors programmieren	28.09.2022	3	30.09.2022	0%		Paulo Ribeiro	
16	Dokumentation Titelblatt, Inhaltsverzeichnis	01.10.2022	1	01.10.2022	0%		Paulo Ribeiro	Benjamin Bäni
17	Impressum, Datenschutz Webpages programmieren	01.10.2022	1	01.10.2022	0%		Paulo Ribeiro	
18	Kontaktformular programmieren	01.10.2022	1	01.10.2022	0%		Paulo Ribeiro	
19	Statusbericht 3	02.10.2022	1	02.10.2022	0%		Paulo Ribeiro	Benjamin Bäni
20	Dokumentation der Spiele	02.10.2022	2	03.10.2022	0%		Paulo Ribeiro	
21	MongoDB recherchieren	04.10.2022	2	05.10.2022	0%		Paulo Ribeiro	
22	2. Vorzeigetermin	06.10.2022	1	06.10.2022	0%		Paulo Ribeiro	Benjamin Bäni
23	Programmierung, Verbindung der MongoDB Datenbank	06.10.2022	3	08.10.2022	0%		Paulo Ribeiro	
24	Statusbericht 4	09.10.2022	1	09.10.2022	0%		Paulo Ribeiro	Benjamin Bäni
25	Anmeldung, Login programmieren und mit MongoDB Datenbank verbinden	09.10.2022	2	10.10.2022	0%		Paulo Ribeiro	
26	Dokumentation der Initialisierung, Planung und Realisierung	10.10.2022	4	13.10.2022	0%		Paulo Ribeiro	
27	Programmcode: Kontrolle, Anpassungen Korrekturen	14.10.2022	2	15.10.2022	0%		Paulo Ribeiro	
28	Statusbericht 5	16.10.2022	1	16.10.2022	0%		Paulo Ribeiro	Benjamin Bäni
29	Dokumentation: Lessons Learned, Reflexion, Problemstellungen, Management Summary	16.10.2022	3	18.10.2022	0%		Paulo Ribeiro	
30	Dokumentation: Kontrolle, Anpassungen Korrekturen, Fertigstellung	19.10.2022	3	21.10.2022	0%		Paulo Ribeiro	
31	Qualifikationsprofil erstellen	22.10.2022	1	22.10.2022	0%			
32	Statusbericht 6	23.10.2022	1	23.10.2022	0%		Paulo Ribeiro	Benjamin Bäni
33	Produktübergabe	24.10.2022	1	24.10.2022	0%		Paulo Ribeiro	

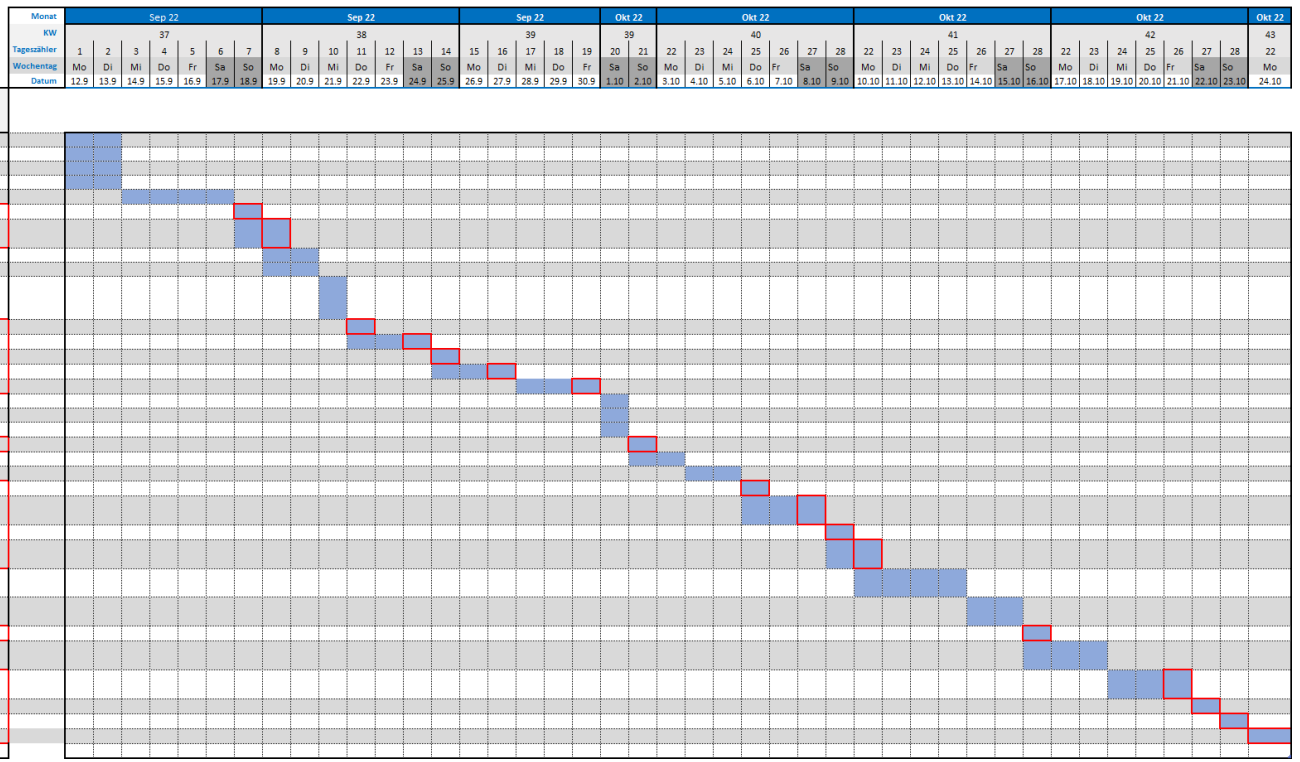


Abbildung 3 Projektablaufplanung

5 Realisierungsphase

In der Realisierungsphase erfolgt die Programmierung und Dokumentation der Diplomarbeit.

5.1 Programmiersprachen

Die verwendeten Programmiersprachen für die Diplomarbeit sind:

- HTML – Die Hypertext Markup Language ist eine textbasierte Auszeichnungssprache zur Strukturierung elektronischer Dokumente wie Texte mit Hyperlinks, Bildern und anderen Inhalten. HTML-Dokumente sind die Grundlage des World Wide Web und werden von Webbrowsern dargestellt. HTML wird für die Darstellung meiner Website verwendet.
- CSS – Cascading Style Sheets ist eine Stylesheet-Sprache für elektronische Dokumente und zusammen mit HTML und JavaScript eine der Kernsprachen des World Wide Webs. Sie ist ein sogenannter «living standard» ‘lebendiger Standard’ und wird vom World Wide Web Consortium beständig weiterentwickelt. CSS wird für das Design meiner Website und Spiele verwendet.
- JavaScript – JavaScript ist eine Skriptsprache, die ursprünglich 1995 von Netscape für dynamisches HTML in Webbrowsern entwickelt wurde, um Benutzerinteraktionen auszuwerten, Inhalte zu verändern, nachzuladen oder zu generieren und so die Möglichkeiten von HTML zu erweitern. Einfach ausgedrückt: In den meisten Fällen wird JavaScript verwendet, um responsive sowie interaktive Elemente für Webseiten zu erstellen. Meine drei Spiele werden in «Vanilla» JavaScript programmiert. «Vanilla» JavaScript bezeichnet, dass man auf einem Verzicht einem Framework verzichtet.

Um die Website lokal zu hosten und einen Backend zu entwickeln, verwende ich Node.js. Node.js ist weder ein Framework noch eine Programmiersprache, sondern eine Laufzeit-Open-Source Entwicklungsplattform zur serverseitigen Ausführung von JavaScript-Code. Mittels Node.js und eingebettete Module, wie beispielsweise: «Express» und «Mongoose», wird die Website lokal gehostet («Express») und die Verbindung der Datenbank («Mongoose») erfolgen.

5.2 Website

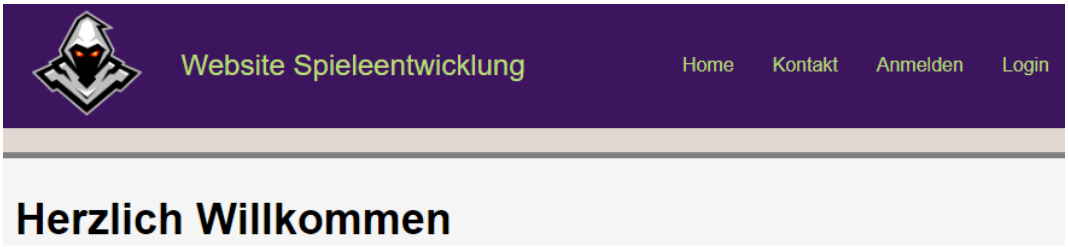


Abbildung 4 Website Navigationsmenü

Die Website wurde mittels HTML und CSS programmiert. HTML für die Elemente der Website und CSS für das Styling (Farben, Positionen der Elemente etc.). Ich entschied mich, ein einfaches Styling für die Website zu programmieren. Die Website besitzt einen «Responsive» Design, das heisst: Elemente werden verkleinert/ vergrössert, wenn sich das Browserfenster dementsprechend verkleinert/vergrössert. Alle Elemente, die auf der Website dargestellt werden, wurden mit HTML programmiert, um diese Darstellung zu ermöglichen.

Abgesehen von der Webseiten der drei Spiele, besitzt die Website auch drei Formulare: ein Kontaktformular, ein Anmeldungsformular und ein Log-in Formular. Die Formulare sind mit der Datenbank MongoDB verbunden, und ermöglichen die dementsprechenden Funktionen auszuführen, etwa die Kontaktaufnahme, die Anmeldung und das Log-in für die Benutzer.

Abbildung 5 Kontaktformular

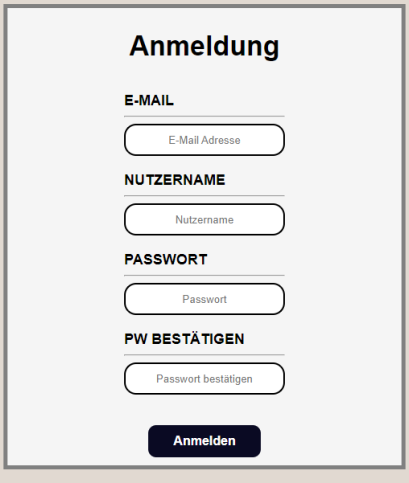


Abbildung 6 Anmeldungsformular

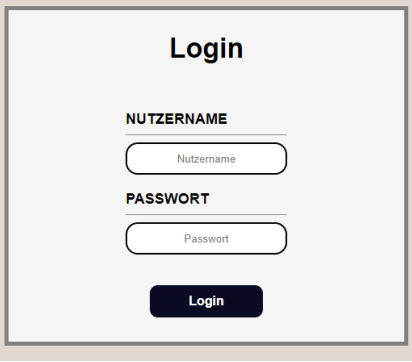


Abbildung 7 Log-in Formular

Weitere Elemente, wie das Impressum und den Datenschutz wurden in der Fusszeile, definiert («Footer»).

Abbildung 8 Website Fusszeile

Die Webseiten der drei Spiele sehen dementsprechend so aus:

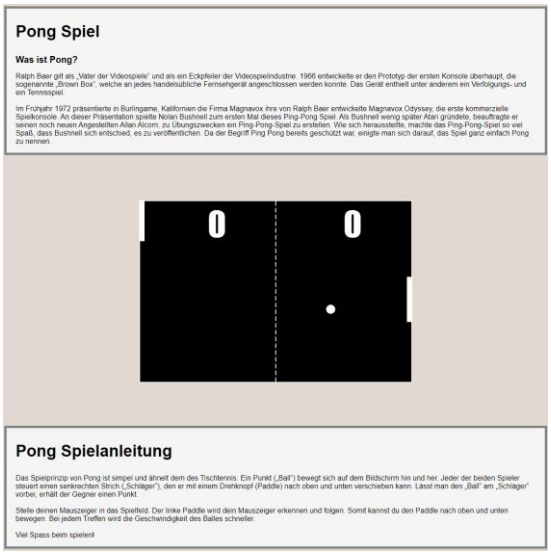


Abbildung 10 Website Pong

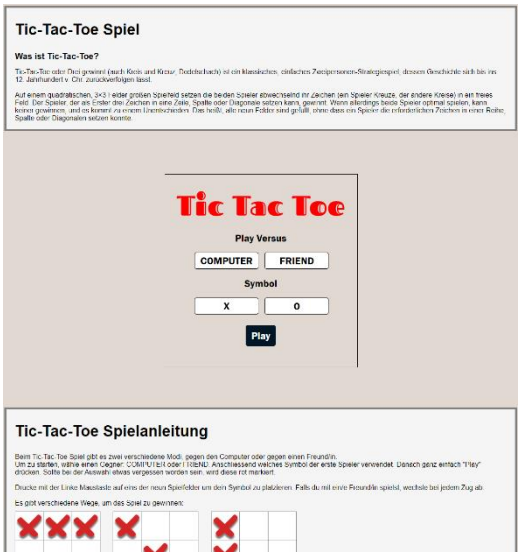


Abbildung 9 Website Tic-Tac-Toe

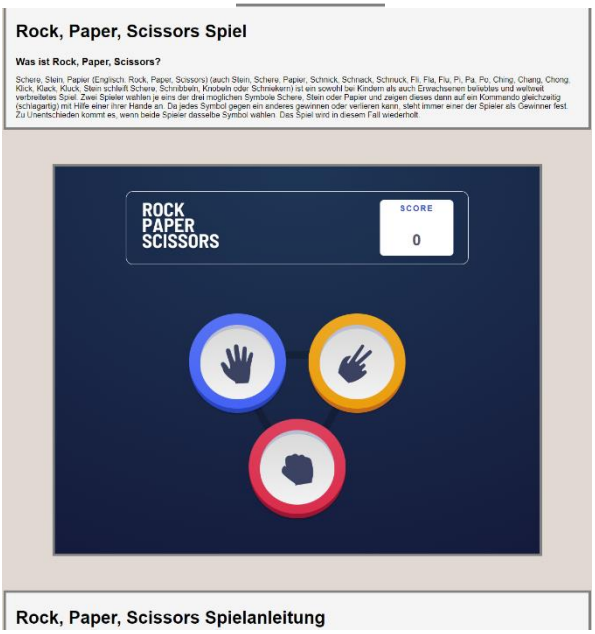


Abbildung 11 Website Rock, Paper, Scissors

Jedes Spiel hat eine Beschreibung, was für ein Spiel es ist (oberhalb) und wie man dieses Spiel spielt (Spielanleitung, unterhalb).

5.3 Spiele

Für die Website wurden drei Spiele programmiert. Ich entschied mich für die klassischen Spiele aus meiner Kindheit: Pong, Rock Paper Scissors, und Tic-Tac-Toe.

5.3.1 Pong

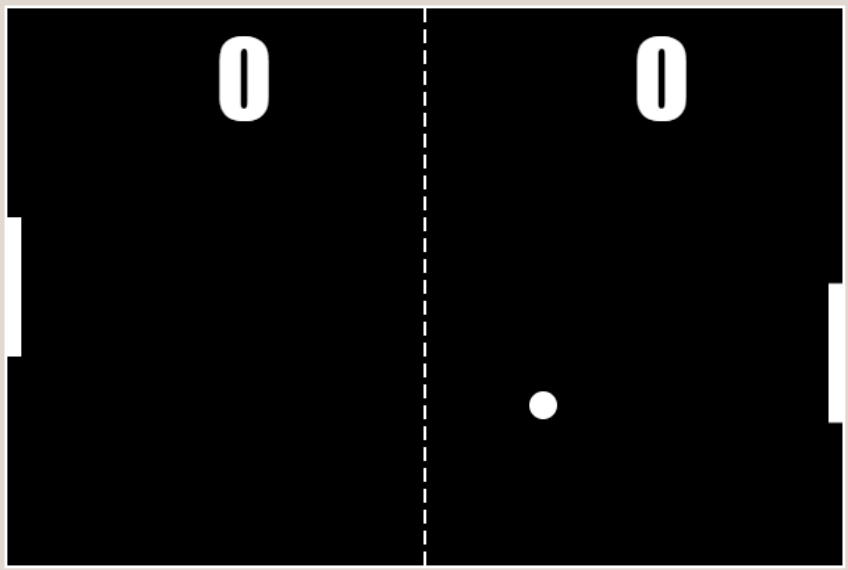


Abbildung 12 Spiele Pong

Ralph Baer gilt als „Vater der Videospiele“ und als ein Eckpfeiler der Videospielindustrie. 1966 entwickelte er den Prototyp der ersten Konsole überhaupt, die sogenannte „Brown Box“, welche an jedes handelsübliche Fernsehgerät angeschlossen werden konnte. Das Gerät enthielt unter anderem ein Verfolgungs- und ein Tennisspiel.

Im Frühjahr 1972 präsentierte in Burlingame, Kalifornien die Firma Magnavox ihre von Ralph Baer entwickelte Magnavox Odyssey, die erste kommerzielle Spielkonsole. An dieser Präsentation spielte Nolan Bushnell zum ersten Mal dieses Ping-Pong Spiel. Als Bushnell wenig später Atari gründete, beauftragte er seinen noch neuen Angestellten Allan Alcorn, zu Übungszwecken ein Ping-Pong-Spiel zu erstellen. Wie sich herausstellte, machte das Ping-Pong-Spiel so viel Spaß, dass Bushnell sich entschied, es zu veröffentlichen. Da der Begriff Ping Pong bereits geschützt war, einigte man sich darauf, das Spiel ganz einfach Pong zu nennen.

Das Spielprinzip von Pong ist simpel und ähnelt dem des Tischtennis: Ein Punk («Ball») bewegt sich auf dem Bildschirm hin und her. Jeder der beiden Spieler steuert einen senkrechten Strich («Schläger»), den er mit einem Drehknopf (Paddle) nach oben und unten verschieben kann. Lässt man den «Ball» am «Schläger» vorbei, erhält der Gegner einen Punk.

Pong Spielanleitung

Pong hat eine kurze und schnelle Anleitung: Um Pong zu spielen, muss man einfach den Mauszeiger auf das Spielfeld setzen. Der linke Paddle wird den Mauszeiger erkennen und folgen. Somit kann man den Paddle nach oben und unten bewegen. Bei jedem Treffen wird die Geschwindigkeit des Balles schneller.

5.3.2 Rock, Paper, Scissors

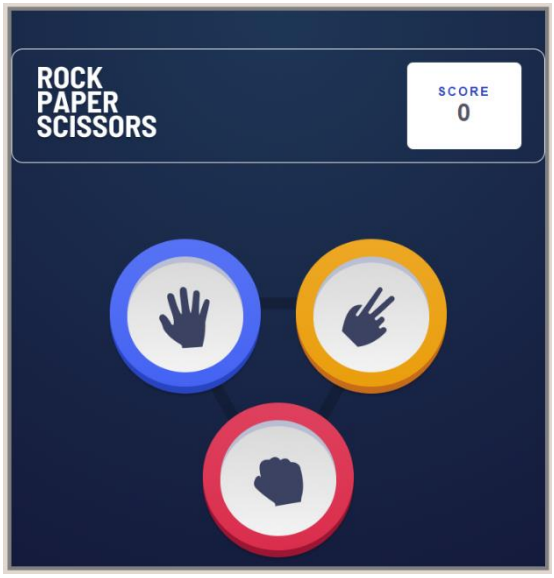


Abbildung 13 Spiele Rock, Paper, Scissors

Schere, Stein, Papier (Englisch: Rock, Paper, Scissors) (auch Stein, Schere, Papier; Schnick, Schnack, Schnuck; Fli, Fla, Flu; Pi, Pa, Po; Ching, Chang, Chong; Klick, Klack, Kluck; Steinschleift Schere; Schnibbeln, Knobeln oder Schniekern) ist ein sowohl bei Kindern als auch Erwachsenen beliebtes und weltweitverbreitetes Spiel. Zwei Spieler wählen je eins der drei möglichen Symbole Schere, Stein oder Papier und zeigen dieses dann auf ein Kommando gleichzeitig (schlagartig) mit Hilfe einer ihrer Hände an. Da jedes Symbol gegen ein anderes gewinnen oder verlieren kann, steht immer einer der Spieler als Gewinner fest. Zu Unentschieden kommt es, wenn beide Spieler dasselbe Symbol wählen. Das Spiel wird in diesem Fall wiederholt.

Rock, Paper, Scissors Spielanleitung

Das Spiel wird ausschließlich mit den Händen gespielt. Den Handhaltungen werden Symbole zugeordnet, die einander „schlagen“ können. Die drei Hauptfiguren sind Schere, Stein und Papier. Das Papier wird durch eine flache Hand mit ungespreizten Fingern dargestellt, das Symbol der Schere ist der gespreizte Zeige- und Mittelfinger, und der Stein wird durch eine Faust symbolisiert.

Um zu starten, muss man mit der Linke Maustaste auf eins der Symbole drücken, um die Auswahl zu markieren. Hier wird nur ein Computer als Gegner dargestellt. Wenn man gewinnt, geht die Punktzahl («Score)» hoch.

Nachdem man sein Zug gespielt hat, muss man «Play Again» drücken, um nochmals zu spielen.

5.3.3 Tic-Tac-Toe

Abbildung 14 Spiele Tic-Tac-Toe

Tic-Tac-Toe oder Drei gewinnt (auch Kreis und Kreuz, Dodelschach)ist ein klassisches, einfaches Zweipersonen-Strategiespiel, dessen Geschichte sich bis ins 12. Jahrhundert v. Chr. Zurück verfolgen lässt.

Auf einem quadratischen, 3×3 Felder großen Spielfeld setzen die beiden Spieler abwechselnd ihr Zeichen (ein Spieler Kreuze, der andere Kreise) in ein freies Feld. Der Spieler, der als Erster drei Zeichen in eine Zeile, Spalte oder Diagonale setzen kann, gewinnt. Wenn allerdings beide Spieler optimal spielen, kann keiner gewinnen, und es kommt zu einem Unentschieden. Das heisst, alle neun Felder sind gefüllt, ohne dass ein Spieler die erforderlichen Zeichen in einer Reihe, Spalte oder Diagonalen setzen konnte.

Tic-Tac-Toe Spielanleitung

Beim Tic-Tac-Toe Spiel gibt es zwei verschiedene Modi, gegen den Computer oder gegen einen Freund/in.

Um zu starten, muss man einen Gegner: COMPUTER oder FRIEND wählen. Anschliessend welches Symbol der erste Spieler verwendet. Danach ganz einfach "Play" drücken. Sollte bei der Auswahl etwas vergessen worden sein, wird diese rot markiert.

Um das gewählte Symbol zu platzieren, muss man mit der Linke Maustaste auf eins der neun Spielfelder drücken. Falls ein/e Freund/in dabei ist, muss man selbständig bei jedem Zug abwechseln.

Es gibt verschiedene Wege, um das Spiel zu gewinnen:

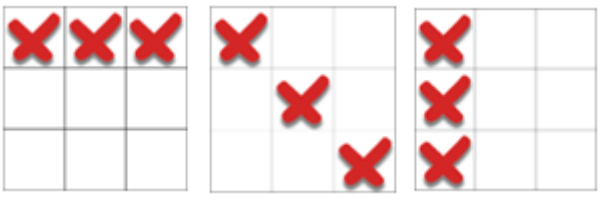


Abbildung 15 Spiele TTT-Gewinnen

Sobald die gleichen Symbolen eine Linie in einer Reihe, diagonale oder waagerechte Richtung bilden, gewinnt man das Spiel.

5.4 Backend

5.4.1 Node JS

Abbildung 16 Node JS

Für das Backend wird Node.js verwendet. Node.js ist eine plattformübergreifende Open-Source-JavaScript-Laufzeitumgebung, die JavaScript-Code ausserhalb eines Webbrowsers ausführen kann. Damit kann zum Beispiel ein Webserver betrieben werden. In meinem Fall verwende ich Node.js mit dem Modul «Express», um die Website lokal zu hosten. Express ist ein Framework, der eine Reihe von Werkzeugen für Webanwendungen bietet: HTPP-Anfragen und -Antworten, Routing und Middleware für die Entwicklung und den Einsatz von grossen, unternehmenstauglichen Anwendungen.

Express bietet auch eine Kommandozeilenschnittstelle (CLI) namens «Node Package Manager (NPM)», über die Entwickler/innen die entwickelten Pakete beziehen können. Ausserdem zwingt es die Entwickler dazu, dem DRY-Prinzip (Don’t Repeat Yourself) zu folgen.

Das DRY-Prinzip zielt darauf ab, die Wiederholung von Softwaremustern zu reduzieren, sie durch Abstraktionen zu ersetzen oder Datennormalisierungen zu verwenden, um Redundanzen zu vermeiden.

Mit dem NP-Manager werden Pakete wie beispielsweise: mongoose installiert. Mit mongoose wird die Verbindung der MongoDB-Datenbank hergestellt und ermöglicht das Speichern der Informationen mittels Modellen in der Datenbank.

5.4.2 Datenbank MongoDB

Abbildung 17 MongoDB

MongoDB ist ein dokumentenorientiertes NoSQL-Datenbankmanagementsystem, das in der Programmiersprache C++ geschrieben ist. Sie kann Sammlungen von JSON-ähnlichen Dokumenten verwalten. So können viele Anwendungen Daten auf natürlichere Weise modellieren, da die Daten zwar in komplexen Hierarchien verschachtelt werden können, dabei aber immer abfragbar und indizierbar bleiben.

Ein MongoDB-Prozess kann mehrere Datenbanken verwalten, und eine Datenbank kann mehrere Collections (Sammlungen) enthalten. Datenbank und Collection ergeben, durch einen Punkt getrennt, einen Namespace. Für eine Datenbank, die die Daten einer Firma verwalten soll, und eine Collection, die alle Mitarbeiter enthalten soll, könnte man beispielsweise den Namespace «firma.mitarbeiter» wählen. In unserem Fall beispielsweise «spieleentwicklung.mitarbeiter».

Eine Collection enthält Dokumente und ist mit einer Tabelle einer relationalen Datenbank vergleichbar. Ein wesentlicher Unterschied besteht darin, dass die Dokumente einer Collection völlig unterschiedlich aufgebaut sein können. Weder müssen sie einem Schema folgen, noch müssen die Werte desselben Schlüssels vom selben Datentyp sein.

6 Programmcode

Damit die Website ohne Störungen lokal gehostet werden kann, muss Node.js auf dem Rechner installiert worden sein und diese folgende Module danach mit dem Terminal im beispielsweise Visual Studio Code installiert werden:

- Express (npm install express)
- Bcryptjs (npm install bcryptjs)
- Cookie-parser (npm install cookie-parser)
- Express-session (npm install express-session)
- Mongoose (npm install mongoose)
- Nodemon (npm install nodemon)

Um alle Module gleichzeitig mit nur einem Befehl zu installieren:

- npm install express bcryptjs cookie-parser express-session mongoose nodemon

Diese Module werden im Abschnitt «Node.js Module» erklärt.

Im Programmcode wird ausführlich die Dokumentation der verschiedenen Dateien aufgelistet und erklärt.

6.1 Dateiverzeichnis

Im Dateiverzeichnis befinden sich verschiedene Dateien und Ordner.

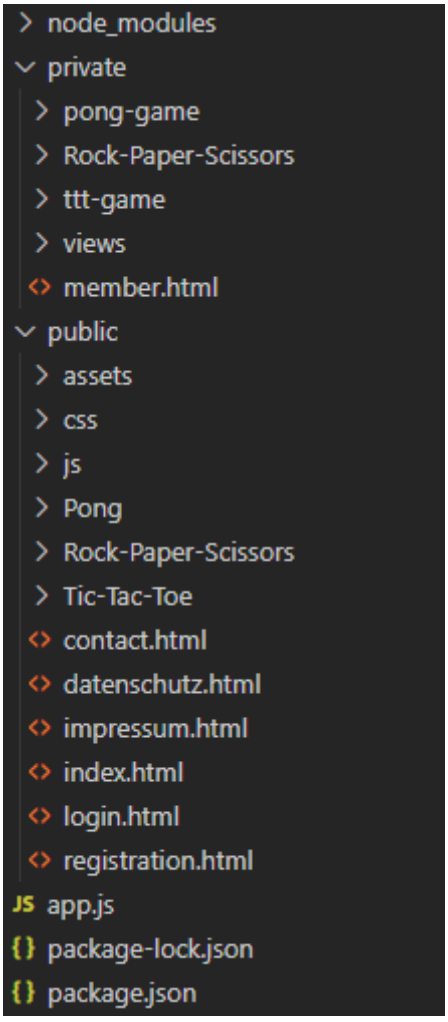


Abbildung 18 Dateiverzeichnis

«node_modules»: Der Ordner node_modules enthält alle installierten Abhängigkeiten für das Projekt. Alle Dateien der Module befinden sich in diesem Ordner.

«private»: Sobald sich einen Benutzer eingeloggt hat, wird der Benutzer in den privaten Ordner «private» umgeleitet. Hier befinden sich die gleichen Dateien wie beim «public» Ordner. Es wurden nur die dementsprechenden Dateipfade geändert und einige Elemente.

«public»: Bei nicht eingeloggten Benutzer ist der public Ordner zugänglich. Dieser beinhaltet alle HTML, CSS, JS-Dateien für die dementsprechenden Programmierungen.

«app.js»: In der app.js Datei befindet sich der Programmcode, der für das Aufstarten des lokalen Servers zuständig ist, sowie die Verbindung zu der Datenbank und die Verarbeitung der Formulare (Kontakt, Anmeldung, Log-in).

«package-lock.json»: In dieser .json Datei, werden alle Vorgängen automatisch generiert, bei denen NPM entweder den node_modules-Baum oder das Paket (packages.json) geändert wurde.

«package.json»: Diese Datei ist das Herzstück eines jeden Node-Projekts. Es zeichnet wichtige Metadaten zu einem Projekt auf, die vor der Veröffentlichung in NPM erforderlich sind, und definiert auch funktionale Attribute eines Projekts, das NPM verwendet, um Abhängigkeiten zu installieren, Skripte auszuführen und den Einstiegspunkt zu unserem Paket zu identifizieren.

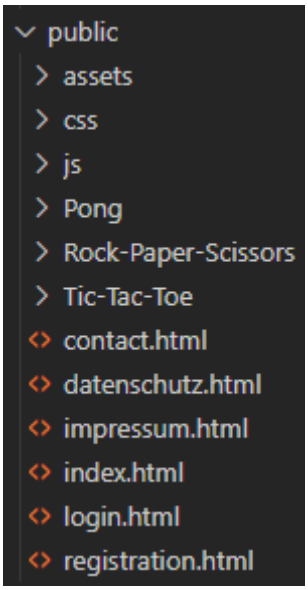
6.2 Website

Für die Programmierung der Website befinden sich viele verschiedene HTML-Dateien, die einigermassen gleiche Elemente besitzen, daher werden nur die wichtigste Elemente hier aufgelistet und erklärt.

Die Website beginnt bei der «index.html» Datei. Wird der Link «localhost:4000» aufgerufen, gelangen die Benutzer auf der Index-Datei (Titelseite, Fontpage).

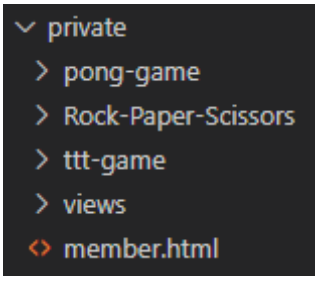
Bei den verschiedenen HTML-Dateien befinden sich diese wiederholte Kategorien:

- **Head** Metadaten, Verknüpfungen etc.
- **Header** Logo, Navigationsmenü etc.
- **Body** Enthält den gesamten Inhalt eines HTML-Dokuments, in unserem Fall auch den Header
- **Footer** Unteren Abschnitt der Website, beinhaltet Informationen über den Autor, Urheberrechte etc.



Auf der Website sind die Dateien im «public» Ordner für jeden zugänglich, der die Website mittels dem Link abruft.

Abbildung 19 Website Dv.



Die Dateien im «private» Ordner sind nur zugänglich für Benutzer, die sich eingeloggt haben.

Abbildung 20 Website Dv.

Index.html – Head

Wir beginnen mit der Erstellung des HTMLs. Mit Visual Studio Code, können wir einfach «!» eingeben und mit «Enter» die erste Auswahl wählen. Dies erstellt uns das HTML-Dokument mit einigen Elemente.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

Abbildung 21 Index HTML

Die drei Metadaten («meta») werden so gelassen, wie sie sind. Meta-Angaben, die von den meisten Suchmaschinen ausgewertet werden, sind: Angaben zum Autor, Kurzbeschreibung des Inhalts, Schlagwörter und nicht zu vergessen das Publikationsdatum. In unserem Fall brauchen wir das nicht, daher lassen wir es so wie es ist.

Im Head verknüpfen wir die entsprechende CSS- und JS-Dateien.

```
<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <link rel="stylesheet" href="css/styles.css" />
  <link rel="stylesheet" href="css/footer.css" />
  <!-- public folder is set in node.js, therefore not needed in the href -->
  <script defer type="text/javascript" src="js/javascript.js"></script>
  <!-- public folder is set in node.js, therefore not needed in the href -->

  <title>Website Spieleentwicklung</title>
</head>
```

Abbildung 22 Index Head

Das «defer»-Attribut im «script» legt fest, dass externe Skripte nach dem Laden der Seite ausgeführt werden. Wird das «defer» durch «async» ausgewechselt, wird das Skript asynchron mit der Webseite ausgeführt, währen diese weiter geladen und geparst wird. Wir benutzen «defer», weil wir in der JS-Datei auf Elemente der Website zugreifen die geladen werden müssen bevor wir diese auswählen.

Styling der Elemente im Body befinden sich in der «styles.css» Datei, die Fusszeile wurde in einer separaten Datei namens «footer.css» programmiert.

Index.html – Header

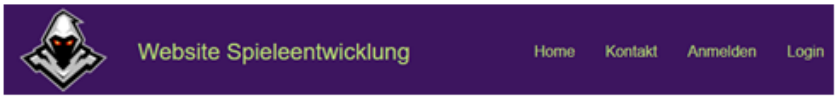
Danach beginnen wir mit dem Header der Website. Der Header befindet sich bei uns im Body. Der Body enthält den gesamten Inhalt eines HTML-Dokuments, in unserem Fall auch den Header.

```html
<body>
  <header>
    <nav class="navbar">
      <div class="logo">
        <a href="index.html">
          <img
            src="assets/imgs/gaming_logo.png"
            alt="Gaming logo von einem Gespenst mit roten Augen."
          />
        </a>
      </div>
      <div class="brand-title">Website Spieleentwicklung</div>
      <a href="#" class="toggle-button">
        <span class="bar"></span>
        <span class="bar"></span>
        <span class="bar"></span>
      </a>
      <div class="navbar-links">
        <ul>
          <li><a href="index.html">Home</a></li>
          <li><a href="contact.html">Kontakt</a></li>
          <li><a href="registration.html">Anmelden</a></li>
          <li><a href="login.html">Login</a></li>
        </ul>
      </div>
    </nav>
  </header>
```

Abbildung 23 Index Header

Mit dem «<header>»-Tag definieren wir der Beginn des Headers und mit «</header>» das Ende. Im Header programmieren wir den Navigationsmenü, der Navigationsmenü besitzt den Tag «<nav>». Im Navigationsmenü befindet sich, der Logo der Website, Websitetitel und die verschiedene Navigationslinks. Werden die Navigationslinks mit der Maustaste gedrückt, werden wir auf die verschiedene HTML-Dokumente umgeleitet.

Der Navigationsmenü sieht dementsprechend, mit dem CSS-Styling, so aus:




```css
/* NAVBAR */
.navbar {
    display: flex;
    align-items: center;
    background-color: #3d155f;
    color: #badd76;
}
.brand-title {
    font-size: 1.5rem;
    margin: .5rem;
}
.logo img {
    height: 100px;
    padding-left: 15px;
}

.navbar-links {
    margin-left: auto;
}
```

```css
.navbar-links ul {
    margin: 5px;
    padding: 0;
    display: flex;
}
.navbar-links li {
    list-style: none;
}
.navbar-links li a {
    text-decoration: none;
    color: #badd76;
    padding: 1rem;
    display: block;
    position: relative;
}
```

```css
.navbar-links li a:after {
    content: '';
    background-color: #ff3d00;
    width: 0;
    height: 2px;
    position: absolute;
    bottom: 0;
    left: 0;
    transition: width 0.5s;
}
.navbar-links li a:hover::after {
    width: 100%;
}
```

Abbildung 24 Index Header Nav

Als Nächstes, kommen die Inhalte, die in der Mitte der Website dargestellt werden sollen. Für das deklarieren wir in der «styles.css» Datei verschiedene Styling Elemente, wie den Hintergrund und die Schriftart. Die Schriftart wird für das gesamte HTML-Dokument verwendet, sowie die weiteren Elemente in (* { }) und (html { }).

```css
* {
    box-sizing: border-box;
    font-family: "poppins", sans-serif;
}

html {
    height: 100%;
    scroll-behavior: smooth;
}
body {
    margin: 0;
    padding: 0;
    background-color: #E1D9D1;
    min-height: 100%;
    display: flex;
    flex-direction: column;
}
```

Abbildung 25 Index HTML CSS

Für die Inhalte des Bodys programmieren wir verschiedene Abschnitte («Sections»). Wir haben den oberen Abschnitt, den mittleren Abschnitt und den Abschnitt der Fusszeile.

Der Erste Abschnitt (oberen Abschnitt) enthält eine Willkommens-Nachricht und listet die Funktionen der Webseite.

```html
<main class="main">
  <div class="outside-div">
    <section class="section-top">
      <div class="top">
        <h1>Herzlich Willkommen</h1>
        <p>
          Diese Website wurde als Diplomarbeit vom Diplomand
          <strong>Paulo Ribeiro</strong> erstellt.
          <br />
          <br />
          Die Website enthält über drei funktionstüchtige Spiele die mit der
          Programmiersprache JavaScript programmiert wurden und einen
          Backend mit der Datenbank MongoDB.
          <br />
          <br />
          Alle Funktionen der Website:
          <br />
          - Anmeldung
          <br />
          - Login
          <br />
          - Kontaktformular
          <br />
          - Spiele spielen
          <br />
          <br />
          Die Anmeldung und das Login finden Sie in dem Navigationsmenü
          oben. Um die Spiele zu spiele, drücken Sie auf eins der unteren
          dargestellten Bildern.
        </p>
      </div>
    </section>
    <section>
```

Abbildung 26 Index Abschnitt oben

Zweiter Abschnitt enthält die Darstellung der drei Spiele. Wenn man auf diese Bilder drückt, wird man zu der bestimmten HTML-Datei der Spiele umgeleitet.

```
    <section>
      <div class="top-2 wrapper">
        <a href="Pong/pong.html">
          <img
            src="assets/imgs/pong/pong-img.png"
            alt="Anklickbarer Bild für das Pong Spiel"
          />
          <div class="overlay">Pong Spielen</div>
        </a>
      </div>
    </section>
    <section>
      <div class="middle wrapper">
        <a href="Tic-Tac-Toe/tic-tac-toe.html">
          <img
            src="assets/imgs/ttt/ttt-img.png"
            alt="Anklickbarer Bild für das Tic-Tac-Toe Spiel"
          />
          <div class="overlay">Tic-Tac-Toe Spielen</div>
        </a>
      </div>
    </section>
    <section class="section-bottom">
      <div class="bottom wrapper">
        <a href="Rock-Paper-Scissors/rps.html">
          <img
            src="assets/imgs/rps/rps-img.png"
            alt="Anklickbarer Bild für das Rock, Paper, Scissors Spiel"
          />
          <div class="overlay">Rock, Paper, Scissors Spielen</div>
        </a>
      </div>
    </section>
  </div>
</main>
```

Abbildung 27 Index Abschnitt Mitte

Nach dem Styling dieses Abschnitts mit einige Effekte sieht der Style so aus:

Abbildung 28 Index Spiele Abschnitt

Die Abschnitte wurden mit diesen Zeilen Code gestylt in der «styles.css» Datei:

```css
/* CONTENT */
.section-top {
    padding-bottom: 50px;
}
.section-bottom {
    padding-bottom: 50px;
}
.main {
    display: table;
    margin-top: 20px;
    margin-left: auto;
    margin-right: auto;
}
.outside-div {
    width: 100%;
    margin-inline: auto;
}

.top {
    flex-direction: column;
    max-width: 1200px;
    background-color: #F5F5F5;
    border: 5px solid grey;
}
.top h1 {
    font-size: 36px;
    margin-left: 20px;
}
.top p {
    margin-left: 20px;
    margin-right: 20px;
}
```

Abbildung 29 Index Style.css

```css
.wrapper {
    flex-direction: column;
    position: relative;
    max-width: 1200px;
    background-color: #F5F5F5;
    border: 5px solid grey;
}
.wrapper img {
    width: 100%;
    display: block;
    transition: .5s ease;
    backface-visibility: hidden;
}
.overlay {
    max-width: 100%;
    transition: .5s ease;
    opacity: 0;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    -ms-transform: translate(-50%, -50%);
    padding: 16px 32px;
    background-color: #696969;
    color: white;
    font-size: 64px;
    text-align: center;
}
.wrapper:hover img {
    opacity: 0.3;
}
.wrapper:hover .overlay {
    opacity: 1;
}
```

Abbildung 30 Index Style.css

Der Letzte Abschnitt ist die Fusszeile. Die Fusszeile beinhaltet die Informationen des beispielsweise, Autors, Kontaktinformationen, sowie die Umleitung zum Impressum oder Datenschutz.

```html
<section class="section-footer">
  <footer class="footer">
    <div class="footer-main">
      <ul class="footer-info">
        <span class="footer-contact-me-text">Kontaktiere mich</span>
        <li>
          <a href="mailto:n.rib.paulo@gmail.com">n.rib.paulo@gmail.com</a>
        </li>
        <br />
        <li>
          <span class="footer-info-span-bottom"
            ><a href="contact.html">Kontaktformular</a></span
          >
        </li>
      </ul>
      <div class="footer-copyright">
        <span>&copy; Website Spieleentwicklung, Paulo Ribeiro 2022</span>
      </div>
      <div class="footer-privacy">
        <span><a href="datenschutz.html">Datenschutz</a></span>
      </div>
      <div class="footer-imprint">
        <span><a href="impressum.html">Impressum</a></span>
      </div>
    </div>
  </footer>
</section>
```

Abbildung 31 Index Abschnitt Fusszeile

Responsive-Design

Für einen «Responsive»-Design definieren wir weitere Styling-Elemente, die so bald, der Browserfenster vergrössert oder verkleinert wird, diese Styling-Optionen aktivieren:

```css
/* MEDIA CSS */
@media (max-width: 900px) {
    .brand-title {
        display: none;
    }
}

@media (max-width: 900px) {
    .overlay {
        font-size: 30px !important;
    }
}
```

Abbildung 32 Responsive Design CSS

```css
@media (max-width: 600px) {
    .toggle-button {
        display: flex;
        top: 38px;
    }

    .navbar-links {
        display: none;
        width: 100%;
    }

    .navbar {
        flex-direction: column;
        align-items: flex-start;
    }
```

Abbildung 33 Responsive Design CSS

```css
    .navbar-links ul {
        width: 100%;
        flex-direction: column;
    }

    .navbar-links li {
        text-align: center;
    }

    .navbar-links li a {
        padding: .5rem 1rem;
    }

    .navbar-links.active {
        display: flex;
    }

    .overlay {
        font-size: 20px !important;
    }
}
```

Abbildung 34 Responsive Design CSS

Beispielsweise wird der Titel mit der definierten Klasse («class»): «.brand-title» verschwinden, sobald der Browserfenster unterhalb der maximalen Breite von 900 Pixel verkleinert wurde.

Abbildung 35 Navigationsmenü

Wir der Browserfenster weiter vekleinert, so werden auch die Elemente in dem Navigationsmenü verschwinden und einen kleinen «Hamburger» dargestellt.

Abbildung 36 Navigationsmenü verkleinert

Um dies zu erreichen, brauchen wir weitere Elemente in der «styles.css» Datei:

Mit der «@media (max-width: 600px)» werden die Navigationslinks (Home, Kontakt, Anmeldung, Login) ausgeblendet (display: none) und den «.toggle-button» dargestellt.

```css
.toggle-button {
    position: absolute;
    top: .75rem;
    right: 1rem;
    display: none;
    flex-direction: column;
    justify-content: space-between;
    width: 30px;
    height: 21px;
}
.toggle-button .bar {
    height: 3px;
    width: 100%;
    background-color: #badd76;
    border-radius: 10px;
}
```

Abbildung 37 Toggle-Button CSS

Sobald der «Hamburger» dargestellt wird, möchten wir ihn aufklappen. Wir erreichen das, indem wir eine JavaScript Funktion anwenden:

```
// Header - Responsive Navbar (Button)
const toggleButton = document.getElementsByClassName('toggle-button')[0];
const navbarLinks = document.getElementsByClassName('navbar-links')[0];

toggleButton.addEventListener('click', () => {
    navbarLinks.classList.toggle('active');
});
```

Abbildung 38 Toggle-Button JS

In diesen Zeilen Code, wird das «toggle-button» und «navbar-links» Elemente mit der «document.getElementsByClassName» Funktion vom HTML abgeholt und als Konstanten definiert (const). Dadurch können wir diese zwei Elemente in der Funktion «toggleButton» verwenden. Hier wird ein «EventListener» eingebaut, sobald der «Hamburger» gedrückt wurde, wird dieser aufgeklappt und zeigt die Navigationslinks (Home, Kontakt, Anmeldung, Login).

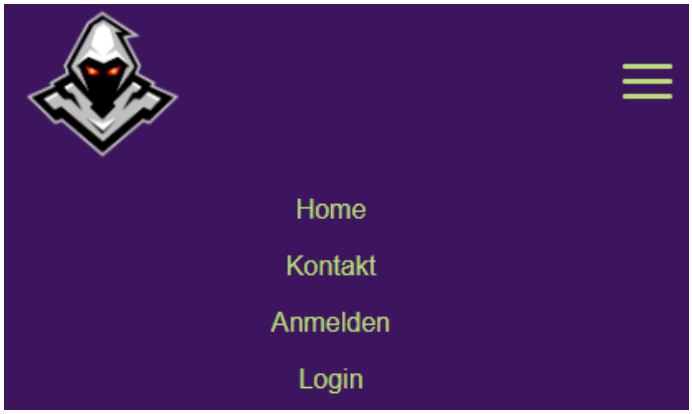


Abbildung 39 Hamburger ausgeklappt

Wenn einen Benutzer sich eingeloggt hat, werden nur die Navigationslinks: Home, Kontakt und Logout angezeigt, dies betrifft das gesamte Dokument.

Die Fusszeile besitzt auch einen «responsive»-Design in der separaten CSS-Datei.

```
@media (max-width: 1200px) {
    .footer-imprint {
        display: block;
        position: relative;
        margin-right: 0px;
    }
    .footer {
        height: 600px;
    }
    .footer-info span {
        margin-left: -1.5em;
    }
    .footer-contact-me-text {
        font-size: 28px;
    }
}
```

Abbildung 41 Responsive Design Fusszeile

```
@media (max-width: 1000px) {
    .footer-copyright{
        display: block;
    }
    .footer-imprint {
        display: block;
        position: relative;
        margin-right: 45px;

    }
    .footer-privacy {
        display: block;
        position: relative;
    }

}
```

Abbildung 40 Responsive Design Fusszeile

6.2.1 Weitere wichtige Website-Merkmale

Formulare

Die Formulare besitzen bei allen Eingabefelder (inputs) einen Namen (name). Diese Namen ermöglichen es beim Backend, die eingegebene Werte in der Eingabefelder abzurufen und diese in der MongoDB zu speichern oder abzurufen.

Beispiel Log-in Formular:

```
<main class="main">
  <div class="outside-div">
    <section>
      <div class="top">
        <div class="outter-form">
        <form class="reg-form" method="post" action="/member.html">
          <h1>Login</h1>
          <div class="form-group">
          <div class="form-group">
            <div class="form-group">
              <label><strong>NUTZERNAME</strong></label>
              <hr>
              <Input type="text" placeholder="Nutzername" class="form-input" name="username" required></Input>
              </div>
              <div class="form-group">
              <label><strong>PASSWORT</strong></label>
              <hr>
              <Input type="password" placeholder="Passwort" class="form-input" name="password" required></Input>
              </div>
          <br>
          <div class="reg-btn">
          <button type="submit" class="button"><strong>Login</strong></button>
        </form>
      </div>
    </section>
</main>
```

Abbildung 42 Log-in Formular HTML

Popup

Für das Kontaktformular und die Anmeldung wurde eine Bestätigung programmiert. Sobald man beispielsweise im Kontaktformular auf «Senden» drückt, erscheint der «Popup».

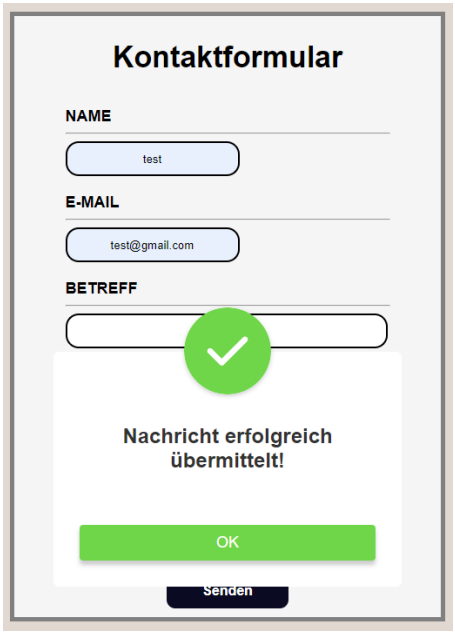


Abbildung 43 Kontaktformular Popup

Der Popup wurde folgendermassen programmiert:

Als Erstes wurde in der jeweiligen HTML-Dateien (contact.html und registration.html) diese Zeilen Code am Formular Zuletzt programmiert.

```html
<div class="popup" id="popup">
  <img src="./assets/imgs/tick.png" alt="Erfolgreich">
  <h2>Nachricht erfolgreich übermittelt!</h2>
  <a href="./index.html"><button type="button">OK</button></a>
</div>
```

Abbildung 44 Popup HTML

Da wir diesen Popup erst bei Knopfdruck sichtbar machen möchten, müssen wir dies mit CSS und JavaScript bestimmen.

```css
.popup {
    width: 400px;
    background: #fff;
    border-radius: 6px;
    position: absolute;
    top: 0%;
    left: 50%;
    transform: translate(-50%, -50%) scale(0.1);
    text-align: center;
    padding: 0 30px 30px;
    color: #333;
    visibility: hidden;
    transition: transform 0.4s, top 0.4s;
}
.open-popup{
    visibility: visible;
    top: 50%;
    transform: translate(-50%, -50%) scale(1);
}
```

Abbildung 46 Popup CSS

```css
.popup img {
    width: 100px;
    margin-top: -15%;
    border-radius: 50%;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
}
.popup h2{
    margin: 30px 0 10px;
}
.popup button {
    width: 100%;
    margin-top: 50px;
    padding: 10px 0;
    background: #6fd649;
    color: #fff;
    border: 0;
    outline: none;
    font-size: 18px;
    border-radius: 4px;
    cursor: pointer;
    box-shadow: 0 5px 5px rgba(0, 0, 0, 0.2);
}
```

Abbildung 45 Popup CSS

Beim CSS des Popups wird die Sichtbarkeit (visibility) auf versteckt (hidden) eingestellt und sobald der Knopf gedrückt wurde, ruft es die «openPopup()» Funktion in der «javascript.js» Datei.

```html
<div class="reg-btn">
<button type="submit" class="button" onclick="openPopup()"><strong>Senden</strong></button>
```

Abbildung 47 onclick openPopup()

```javascript
// POPUP - Kontaktformular, Anmeldung etc.
let popup = document.getElementById("popup");

function openPopup() {
        popup.classList.add("open-popup");
}

function openPopupReg() {
    popup.classList.add("open-popup");
}
```

Abbildung 48 openPopup() JS Funktion

Die «openPopup()» Funktion stellt die in der CSS-Datei programmierten Element namens «.open-popup» als aktives Element an und somit öffnet sich der Popup.

6.3 Node.js Modules

6.3.1 Express

Express ist ein Node-js-Framework für Webanwendungen, das umfassende Funktionen zum Erstellen von Web- und Mobilanwendungen bietet. Es wird verwendet, um einseitige, mehrseitige und hybride Webanwendungen zu erstellen.

Es ist eine Schicht, die auf dem Node.js aufgebaut ist und bei der Verwaltung von Servern und Routen hilft.

Warum Express?

- Express wurde entwickelt, um APIs und Webanwendungen einfach zu erstellen.
- Es spart eine Menge Programmierzeit, fast um die Hälfte.
- Web- und mobile Anwendungen sind effizient.
- Einen weiteren Grund für die Verwendung von Express ist, dass es in JavaScript geschrieben ist, da JavaScript eine einfache Sprache ist, auch wenn man keine Vorkenntnisse in einer Programmiersprache besitzt.
- Express ermöglicht vielen neuen Entwicklern den Einstieg in die Webentwicklung.

Der Grund für die Erstellung eines Express-Frameworks für Node JS ist:

- zeiteffizient
- schnell
- wirtschaftlich
- leicht zu lernen
- Asynchron

Da wir Express verwenden, können wir unseren lokalen Server mit dieser Programmierung starten:

```
const express = require('express');
const app = express();
const port = 4000;

app.get('/', function(req, res) {
    res.sendFile(__dirname + '/index.html')
});

app.listen(port, () => {
    console.log("Running")
})
```

Abbildung 49 Node JS Express

Mit der Eingabe von «npm start» im Terminal, starten wir den Express Server auf dem Port 4000. Im Browser können wir nun «localhost:4000» eingeben und uns wird die «index.html» Datei dargestellt.

6.3.2 Node JS Middleware

Die Middleware in Node JS ist eine Funktion, die den gesamten Zugriff hat, um ein Objekt anzufordern, auf ein Objekt zu antworten und zur nächsten Middleware-Funktion im Anfrage-Antwort-Zyklus der Anwendung zu wechseln. Diese Funktion kann zum Ändern der req- und res-Objekte für Aufgaben wie das Hinzufügen von Antwortheadern, das Analysieren von anfordernden Texten usw. verwendet werden.

Middleware ist erforderlich, um Entwicklern dabei zu helfen, Anwendungen effektiver und effizienter zu erstellen. Die Middleware fungiert als Verbindung zwischen Daten, Anwendungen und den Benutzern. Als Unternehmen mit einer Multi-Cloud-Umgebung wird Middleware die Entwicklung und den Betrieb der Anwendung in grossem Massstab kostengünstiger machen.

6.3.3 Bcryptjs

Für die Anmeldung der Kunden verwenden wir Bcryptjs, um die Passwörter zu verschlüsseln (Hashen). Dieses Modul ermöglicht das Speichern von Passwörtern als gehashte Passwörter anstelle von Klartext.

Die Verwendung von Bcryptjs erfolgt bei der Anmeldung und Log-in der Kunden.

6.3.4 Cookie-parser

Cookie-parser ist eine Middleware von Node JS, die zum Abrufen von Cookie-Daten verwendet wird. Um Cookie-Daten in Express JS abzurufen, wird die Eigenschaft «req.cookies» verwendet. «req.cookies» ist ein Objekt, das Cookies enthält, die nach dem Parsen per Anfrage in JSON gesendet werden.

Cookie-parser wird beispielsweise bei der Website verwendet, nachdem sich ein Kunde eingeloggt hat, um auf dem Mitgliedsbereich («Member Area») zu gelangen.

6.3.5 Express-session

Um eine Sitzung im Express-Framework zu erstellen, wird das Modul: express-session verwendet. Das Express-Session-Modul stellt eine Methode und Eingeschalten bereit, die Werte aus der Sitzung festlegen und abrufen können.

Express-Sitzungen werden in einer Node-js-Webanwendung verwendet, um den Status eines Benutzers beizubehalten.

Das am häufigsten verwendete Szenario der Sitzung ist das Authentifizierungssystem. Wenn sich Benutzer bei einem beliebigen System anmelden, können sie ihre Aktivitäten anhand ihrer ID sehen. Daher müssen wir die ID des Benutzers in einer Sitzung speichern und diese ID dann abfragen, um andere Informationen über diesen Benutzer anzuzeigen.

6.3.6 Mongoose

Mongoose ist eine Node JS-basierte ODM-Bibliothek(Object Data Modeling) für die Datenbank MongoDB. Das Problem, das Mongoose lösen möchte, besteht darin, Entwicklern zu ermöglichen, ein bestimmtes Schema auf der Anwendungsebene durchzusetzen. Neben der Erzwingung eines Schemas bietet Mongoose auch eine Vielzahl von Hooks, Modellvalidierung und andere Funktionen, die darauf abzielen, die Arbeit mit MongoDB zu vereinfachen.

MongoDB-Schemavalidierung ermöglicht es, ein Schema einfach gegen die MongoDB-Datenbank durchzusetzen und gleichzeitig ein hohes Mass an Flexibilität beizubehalten, sodass man das Beste aus beiden Welten erhalte. In der Vergangenheit bestand die einzige Möglichkeit, ein Schema für eine MongoDB-Sammlung durchzusetzen, darin, dies auf Anwendungsebene mit einem ODM wie Mongoose zu tun, aber das stellte Entwickler vor erhebliche Herausforderungen.

Mongoose hat die Aufgaben: Die Website mit der Datenbank MongoDB verbinden und Schemas festzulegen und diese an Funktionen weiterleiten.

6.3.7 Nodemon

Nodemon ist ein Tool, das bei der Entwicklung von Node JS-basierten Anwendungen hilft, indem es die Node-Anwendung automatisch neustartet, wenn Dateien im Verzeichnis verändert wurden und diese Veränderungen erkannt werden.

Nodemon erfordert keine zusätzlichen Änderungen am Code oder Entwicklungsmethode.

6.4 Node.js Code und Erklärung

Die Datei zum Starten vom Express Server und deren Funktionen heisst: app.js.

Wie bei der Erklärung des Expresses Modul starten wir zuerst den lokalen Server:

```
const express = require('express');
const app = express();
const port = 4000;

app.get('/', function(req, res) {
    res.sendFile(__dirname + '/index.html')
});

app.listen(port, () => {
    console.log("Running")
})
```

Abbildung 50 Node JS Express

Wir holen das Express-Modul mit der «require» Funktion und definieren diese als eine Konstante Variabel namens «express». Unsere Datei «app.js» daher wird auch diese als eine Konstante Variabel definiert namens «app» und als «express()» deklariert.

«App.get» holt die «index.html» Datei und sendet diese am Browser. Sodass wenn wir im Browser «localhost:4000» eingeben, die «index.html» Datei eingeblendet wird.

In der «app.js» Datei deklarieren wir zusätzlich alle Module und binden diese ein:

```
const express = require('express');
const port = 4000;
const app = express();
const bcrypt = require('bcryptjs');
const cookieParser = require('cookie-parser');
const sessions = require('express-session');

app.use(express.json());
app.use(express.urlencoded({ extended: true}));

app.use(express.static(__dirname));

app.use(cookieParser());

var session;
```

Abbildung 51 Node JS Konstanten

Da wir zwei Ordner besitzen: Private und Public, geben wir den Public frei. Mit dieser Zeile Code wird express den Pfad erkennen und jeweilige Dateien einblenden.

```
app.use(express.static(__dirname + '/public'));
```

Abbildung 52 Node JS Public Ordner

Für die Verbindung der Datenbank deklarieren wir das Modul: Mongoose nach der Serverprogrammierung.

```
// For the Connection, Registration, Login
const mongoose = require("mongoose");

mongoose.connect("", { useNewUrlParser: true }, { useUnifiedTopology: true});
```

Abbildung 53 Node JS Mongoose

In den Anführungszeichen wird der String zur Verbindung der MongoDB eingegeben (mongodb+srv://Paulo:1paulo1@spieleentwicklung.x3hjdb0.mongodb.net/spieleentwicklung?retryWrites=true&w=majority). Dieser String wird auf der MongoDB automatisch erstellt, nachdem eine Datenbank erstellt wurde.

Schemas

Wir verwenden zwei erstellten Schemas, eine namens «contactSchema» und die andere namens «userSchema». Um diese Schemas zu verwenden, deklarieren wir eine Konstante Variabel und definieren, wo in der Datenbank diese Daten gespeichert werden.

```
// Contact
// Setting the contactSchema
const contactSchema = {
  name: String,
  email: String,
  subject: String,
  message: String
}

// Contact Model accessing contactSchema
const Contact = mongoose.model("Contacts", contactSchema);
```

Abbildung 55 Node JS Contacts Schema

```
// Setting the userSchema
const userSchema = {
  email: String,
  username: String,
  password: String
}
// User Model accessing userSchema
const User = mongoose.model("Users", userSchema);
```

Abbildung 54 Node JS User Schema

Beim «contactSchema» werden die Daten in der «Collection» (Sammlung) «Contacts» gespeichert und das «userSchema» in der Sammlung «Users» gespeichert. Dies wird in dem Abschnitt «Datenbank MongoDB» erklärt, wie dies in der Datenbank funktioniert.

Formulare

Da wir insgesamt drei Formulare in der jeweiligen HTML-Dateien programmierten: Kontaktformular, Anmeldeformular und Log-in Formular, werden diese auch über die Datei «app.js» kontrolliert.

Beim Kontaktformular werden die jeweiligen Eingabefelder abgerufen, in einem neuen «Contact» gespeichert und an die MongoDB Datenbank versendet (post).

```
app.post("/contact", async function(req, res) {

  async function sendContact(){
    let newContact = new Contact({
      name: req.body.name,
      email: req.body.email,
      subject: req.body.subject,
      message: req.body.message
    });
    await newContact.save();
  }
  return sendContact();
});
```

Abbildung 56 Node JS Post Contact

Als beim Kontaktformular werden im Anmeldeformular noch weitere Funktionen eingebaut.

```
// Registration
app.post("/registration", async function(req, res) {
  try {
    const { email, username, password, passwordCheck } = req.body;
    // VALIDATING
    // STATUS CODE 300: BAD REQUEST
    // STATUS CODE 500: INTERNAL SERVER ERROR

    if (!email || !username || !password || !passwordCheck) {
      return res.status(400).json({ msg: "Es wurden nicht alle Felder ausgefüllt" });
    }
    // CHECKING THE PASSWORD ENTERED VS THE PASSWORD CHECKER
    if (password !== passwordCheck) {
      return res
        .status(400)
        .json({ msg: "Passwörter stimmen nicht überein. Bitte versuche es erneut" });
    }
    // CHECKING DATABASE AND EMAIL CHECK TO ENSURE NO DUPLICATE EMAILS UPON REGISTERING
    const existingEmail = await User.findOne({ email: email });
    if (existingEmail) {
      return res
        .status(400)
        .json({ msg: "Ein Konto mit dieser E-Mail existiert bereits" });
    }
    // CHECKING DATABASE AND USERNAME CHECK TO ENSURE NO DUPLICATE USERNAMES UPON REGISTERING
    const existingUsername = await User.findOne({ username: username });
    if (existingUsername) {
      return res
        .status(400)
        .json({ msg: "Benutzername ist vergeben" });
    }
    // USING BCRYPT TO HASH PASSWORDS FOR SECURITY
    const salt = await bcrypt.genSalt();
    const passwordHash = await bcrypt.hash(password, salt);

    // CREATING NEW USER, PASSWORD WILL BE THE HASHED PASSWORD AND NOT ENTERED PASSWORD
    const newUser = new User({
      email: email,
      username: username,
      password: passwordHash
    });
    await newUser.save();
    return newUser;

    // CATCHING ERRORS
  } catch (error) {
    res.status(500).json({ err: error.message });
  }
});
```

Abbildung 57 Node JS Registration

Die Funktion deklariert alle Eingaben: email, username, password, passwordCheck als eine Konstante, und fordert die Eingabefelder.

Zusätzlich werden drei «if» Anweisungen programmiert. Die Erste dient der Kontrolle, ob alle Eingabefelder ausgefüllt wurden. Sollte das nicht der Fall sein, wird eine Nachricht eingeblendet, dass nicht alle Eingabefelder ausgefüllt wurden.

Als Zweites wird überprüft, ob die eingegebene E-Mail-Adresse schon existiert. Dadurch wird vermieden, mehrere Kontos mit der gleichen E-Mail zu eröffnen.

Wie bei der E-Mail, wird als letztes überprüft, ob der Nutzername auch bereits existiert. Für zukünftige Funktionen möchten wir nicht, dass zwei Benutzer den gleichen Nutzernamen besitzen, beispielsweise für das Log-in mit Nutzername.

Nun, wird beim Anmeldungsformular das Passwort verschlüsselt. Wir holen das eingegebene Passwort und «hashen» (verschlüsseln) dieses mit dem generierten Schlüssel («bcrypt.genSalt()»).

```
// USING BCRYPT TO HASH PASSWORDS FOR SECURITY
const salt = await bcrypt.genSalt();
const passwordHash = await bcrypt.hash(password, salt);
```

Abbildung 58 Node JS Bcrypt

Sobald die «if» Anweisungen durch sind und das Passwort verschlüsselt wurde, wird der neue Benutzer in der Datenbank gespeichert. Anstatt das eingegebene Passwort zu speichern, wird das verschlüsselte Passwort gespeichert.

```
// CREATING NEW USER, PASSWORD WILL BE THE HASHED PASSWORD AND NOT ENTERED PASSWORD
const newUser = new User({
  email: email,
  username: username,
  password: passwordHash
});
await newUser.save();
return newUser;
```

Abbildung 59 Node JS newUser

Beim Log-in Formular geschieht fast das gleiche wie beim Anmeldungsformular.

```
app.post('/member.html', async (req,res) => {
  try {
    const { username, password } = req.body;
    // VALIDATE
    if (!username || !password) {
    return res.status(400).json({ msg: "Es wurden nicht alle Felder ausgefüllt" });
    }
    // CHECKING USERNAME THAT WAS ENTERED AND COMPARING EMAIL IN DATABASE
    const user = await User.findOne({ username: username });
    if (!user) {
    return res
      .status(400)
      .json({ msg: "Ungültige Anmeldeinformationen" });
    }
    // CHECKING PASSWORD ENTERED AND COMPARING IT WITH HASHED PASSWORD IN DATABASE
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) {
    return res.status(400).json({ msg: "Ungültige Anmeldeinformationen" });
    }

    if(username && password){
      session = req.session;
      session.userid = req.body.username;
      console.log(req.session)
      res.sendFile('private/member.html', {root:__dirname});
    } else {
      return res.status(400).json({ msg: "Benutzername oder Passwort ist nicht korrekt."});
    }

  } catch (error) {
    res.status(500).json({ err: error.message });
  }
})
```

Abbildung 60 Node JS Post Member

Es überprüft, ob alle Eingabefelder eingegeben wurden und sucht in der Datenbank nach dem Benutzer. Als Suchfilter wird der Nutzername verwendet.

Sobald es einen Benutzer gefunden hat, wird das Passwort überprüft. Mit der «bcrypt.compare()» Funktion, wird das eingegebene Passwort mit dem Benutzerpasswort verglichen. Nehmen wir an, wir haben uns mit dem Passwort «test» angemeldet und beim Log-in «test» eingeben. Auf der Datenbank wurde das verschlüsselte Passwort gespeichert und hier wird es verglichen, ob es die gleiche Verschlüsselung ist.

```
// CHECKING PASSWORD ENTERED AND COMPARING IT WITH HASHED PASSWORD IN DATABASE
const isMatch = await bcrypt.compare(password, user.password);
if (!isMatch) {
return res.status(400).json({ msg: "Ungültige Anmeldeinformationen" });
}
```

Abbildung 61 Node JS Bcrypt Compare

Stimmen der Nutzername und das Passwort, wird eine «Session» (Sitzung) geöffnet.

```
if(username && password){
  session = req.session;
  session.userid = req.body.username;
  console.log(req.session)
  res.sendFile('private/member.html', {root:__dirname});
} else {
  return res.status(400).json({ msg: "Benutzername oder Passwort ist nicht korrekt."});
}
```

Abbildung 62 Node JS Sitzung

Um die eingeloggten Benutzer in den privaten Ordner weiterzuleiten, wo sich der Mitgliedsbereich befindet, müssen wir mittels der Cookies die Sitzung starten. Für das definieren wir vor der Log-in Funktion eine Konstante und die «app.use(sessions()» Funktion.

```
const oneDay = 1000 * 60 * 60 * 24;
app.use(sessions({
  secret: "thisismysecretkeyfhasdhk32",
  saveUninitialized: true,
  cookie: { maxAge: oneDay },
  resave: false
}));
```

Abbildung 63 Node JS Sitzung-Cookie

«Secret» - Ein zufälliger eindeutiger Zeichenfolgeschlüssel, der zur Authentifizierung einer Sitzung verwendet wird. Es wird in einer Umgebungsvariablen gesichert und kann nicht öffentlich zugänglich gemacht werden. Der Schlüssel ist normalerweise lang und wird in einer Produktionsumgebung zufällig generiert.

«resave» - Nimmt einen booleschen Wert an. Dadurch kann die Sitzung wieder im Sitzungsspeicher gespeichert werden, selbst wenn die Sitzung während der Anfrage nie geändert wurde. Dies kann zu einer Renn-Situation führen, falls ein Client zwei parallele Anfragen an den Server stellt. Somit kann eine an der Sitzung der ersten Anfrage vorgenommene Modifikation überschrieben werden, wenn die zweite Anfrage endet. Der Standardwert ist wahr. Dies kann sich jedoch irgendwann ändern. «False» ist eine bessere Alternative.

«saveUninitialized» - Dadurch kann jede nicht initialisierte Sitzung an den Store gesendet werden, wenn eine Sitzung erstellt, aber nicht geändert wird, wird sie als nicht initialisiert bezeichnet.

«cookie: { maxAge: oneDay }» - Dadurch wird die Ablaufzeit des Cookies festgelegt. Der Browser löscht das Cookie nach Ablauf der eingestellten Dauer. Das Cookie wird in Zukunft keiner der Anfragen hinzugefügt. In diesem Fall haben wir das «maxAge» auf einen einzelnen Tag gesetzt, wie bei der Konstante «oneDay» berechnet.

6.5 Datenbank MongoDB

Die Datenbank «Spieleentwicklung» erstellte ich auf der MongoDB Cloud.
(MongoDB Cloud: https://cloud.mongodb.com)

Abbildung 64 MongoDB Datenbank

Verbindung aufbauen

Durch Betätigen des «Connect» Knopfes, wird den String durch MongoDB erstellt für die Verbindung mittels Node.js.

```
mongodb+srv://Paulo:<password>@spieleentwicklung.x3hjdb0.mongodb.net/?
retryWrites=true&w=majority
```

Abbildung 65 MongoDB Verbindungsstring

Wie beim Node.js Code erklärt, muss dieser String in der «mongoose.connect» Funktion eingefügt werden. Das <password> muss man durch das erstellte Passwort für die Datenbank ersetzen.

```
// For the Connection, Registration, Login
const mongoose = require("mongoose");

mongoose.connect("mongodb+srv://Paulo:1paulo1@spieleentw
```

Abbildung 66 MongoDB Node JS Verbindung

Somit ist die Verbindung von Node.js zur Datenbank MongoDB hergestellt worden.

Collections

Mittels Node.js werden zwei Sammlungen (Collections) erstellt, die Kontakte (contacts) und Benutzer (users).

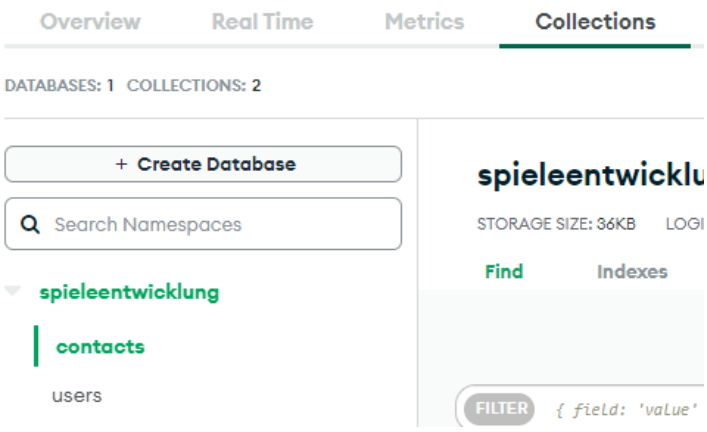


Abbildung 67 MongoDB Sammlungen

In diese zwei Sammlungen werden die Daten aus den Formularen gespeichert. Bei Kontaktformular-Anfragen werden diese in der Sammlung «contacts» gespeichert.

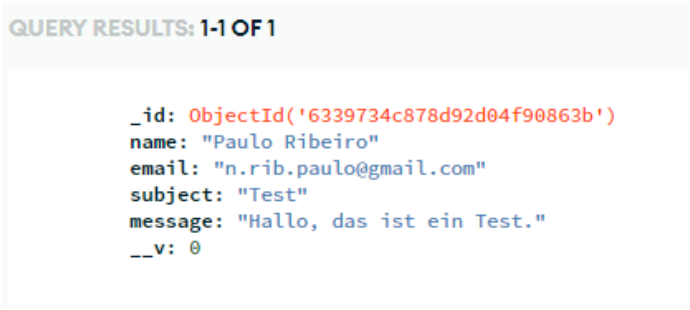


Abbildung 68 MongoDB Contacts

Für die Anmeldung werden die Daten in der Sammlung «users» gespeichert und dann beim Log-in abgerufen.

Abbildung 69 MongoDB Users

6.5.1 Datenbank MongoDB Prüfung

Als Prüfung erstellte ich eine Anmeldung mit den jeweiligen Daten:

- **email:** test@gmail.com
- **username:** test
- **password:** test

Wie man sieht, wurde nicht das Passwort «test» gespeichert, sondern das gehashte Passwort, wie beim Node.js Code erklärt.

Abbildung 70 MongoDB Users Prüfung

Bei der zweiten Prüfung wurde das gleiche Passwort eingegeben: «test». Bcrypt hashte für uns ein neues Passwort, wenn wir beide vergleichen, sind diese nicht gleich. Wir stellen nun fest das, dass hashing der Passwörter einwandfrei funktioniert.

Abbildung 71 MongoDB Users Prüfung

6.6 Spiele

Alle programmierten Spiele wurden mithilfe von Google und YouTube programmiert. Es werden nur die wichtigen Punkte hervorgehoben und erklärt. Beim Code selbst wurde so oft wie möglich kommentiert (// Beispiel), somit war es für mich einfacher, alles zu nachvollziehen und zu dokumentieren.

Für jedes Spiel wurde zusätzlich bei jeder HTML-Datei eine Erklärung geschrieben, die die Geschichte von den jeweiligen Spielen erzählt und unterhalb des Spiels befindet sich eine Spielanleitung.

Es wird empfohlen, die Dokumentation der Programmcodes mit dem geöffneten, kompletten Code durchzulesen.

6.6.1 Pong

Die Programmierung des Pong Spiels erfolgte mit der Hilfe und Quelle von «Code Explained» auf YouTube. Da wurden Funktionen beschrieben und erklärt, was diese tun.

Die Dateistruktur des Pong Spiels sieht folgendermassen aus:

Es befinden sich drei Dateien, die für die Programmierung bestimmt sind: pong-game.js, pong-style.css, pong.html und den Ordner mit den Tönen.

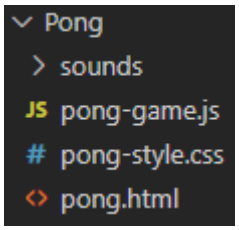


Abbildung 72 Pong Dateiverzeichnis

Die verwendeten Programmiersprachen: pong.html in HTML (Hypertext Markup Language), pong-style.css in CSS (Cascading Style Sheets) und pong-game.js (JavaScript).

HTML – pong.html

In der HTML-Datei befinden sich fast die gleichen Elemente wie beim index.html der Website. Es wurde den «Canvas» (Leinwand, Grafikelemente) Element hinzugefügt. Navigationsmenü und die Fusszeile blieben gleich.

```
<canvas id="pong" width="600" height="400"></canvas>
```

Abbildung 73 Pong Canvas

Das Canvas Element wurde definiert mit der jeweiligen ID, Breite und Höhe. Die ID wird verwendet, um bei der pong-game.js Datei das Canvas zu selektieren.

CSS – pong-style.css

Die pong-style.css Datei wurde in der pong.html Datei eingebettet und dient als Stylesheet für das Pong Spiel.

Hier wurde nur der Rand (Border), die Abstände (margin), die Schriftgrösse(font-size) und das Display definiert.

```
#pong {
    border: 2px solid #fff;
    display: flex;
    margin: auto;
    margin-top: 50px;
    margin-bottom: 75px;
  }
  .top h2 {
    margin-left: 20px;
    font-size: 20px;
  }
```

Abbildung 74 Pong Style CSS

JavaScript – pong-game.js

Als erstes, selektieren wir das Canvas. Hierzu habe ich zwei «const» (constant), auf Deutsch Konstanten, die nicht verändert werden, deklariert. Für das Zeichnen des Spielfelds und die weiteren Elemente, müssen wir die Konstante «ctx» als «canvas.getContext('2d')» definieren.

```
// SELECT CANVAS
const canvas = document.getElementById("pong");
const ctx = canvas.getContext('2d');
```

Abbildung 75 Pong Canvas auswählen

Für die Sounds laden wir vier Audio-Dateien. Für je einen Sound wurde je eine «let» Variabel definiert. Mit «let» definierte Variablen können nicht neu deklariert werden. Diese Sounds werden verwendet, wenn der Ball (hit), die Wand (wall) getroffen wird und beim Erhalten von Punkte, Spieler und Computer.

```
// LOAD SOUNDS
let hit = new Audio();
let wall = new Audio();
let userScore = new Audio();
let comScore = new Audio();

hit.src = "sounds/hit.mp3";
wall.src = "sounds/wall.mp3";
comScore.src = "sounds/comScore.mp3";
userScore.src = "sounds/userScore.mp3";
```

Abbildung 76 Pong Töne auswählen

Pong Komponenten

Als Nächstes, müssen wir die Pong Komponenten definieren. Wir haben einen «Table» (Canvas gefüllt mit Schwarz), zwei «Paddles» (Schläger) eins für den Spieler (links) und eins für den Computer (rechts), das Netz in der Mitte, der Ball und die Punktzahl des Spielers (links) und Computers (rechts).

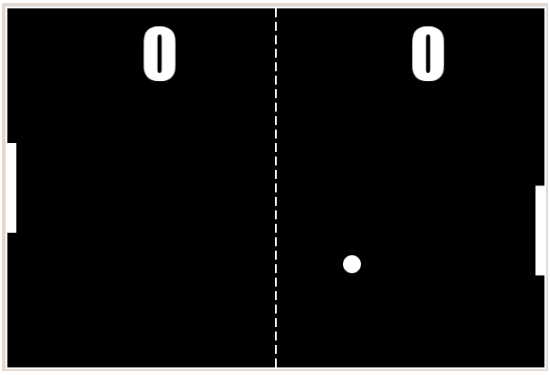


Abbildung 77 Pong Komponenten

Um die verschiedenen Komponenten zu erstellen, beziehungsweise auf das Canvas Zeichnen, müssen wir den Ball (ball), beide Schläger (User Paddle, Com Paddle), das Netz (net) und den Punktestand programmieren.

Wir beginnen beim User und Com Paddle (Schläger). Zuerst definieren wir die zwei Objekte User und Com:

```
// CREATE THE USER PADDLE
const user = {
    x : 0, // left side of canvas
    y : (canvas.height - 100)/2, // -100 the height of paddle
    width : 10,
    height : 100,
    score : 0,
    color : "WHITE"
}
```

Abbildung 78 Pong User Paddle

```
// CREATE THE COM PADDLE
const com = {
    x : canvas.width - 10, // - width of paddle
    y : (canvas.height - 100)/2, // -100 the height of paddle
    width : 10,
    height : 100,
    score : 0,
    color : "WHITE"
}
```

Abbildung 79 Pong Com Paddle

Wir definieren für den User und Com Schläger: eine Breite von 10 Pixel, Höhe von 100 Pixel, Farbe Weiss und den Score (Punkte) stellen wir auf 0. Die Position des Schlägers definieren wir, indem wir eine X- und Y-Position geben. Beim User hat die Position X eine Null, somit ist der Schläger ganz links beim Canvas. Für die Position des Schlägers vom Computer stellen wir bei der X-Position die «canvas.width» (Breite der Canvas) – 10 Pixeln für die Breite des Schlägers, somit ist der Com-Schläger ganz rechts. Für die Y-Position wird bei beiden das Gleiche programmiert: «canvas.height/2 – 100/2». Dies steht für die Hälfte der Canvas Höhe und die Höhe des Schlägers (100 Pixeln) geteilt durch zwei, um von der Hälfte aus auf beiden Seiten 50 Pixeln zu zeichnen.

Um es sich besser vorzustellen, kann man diese Abbildung betrachten:

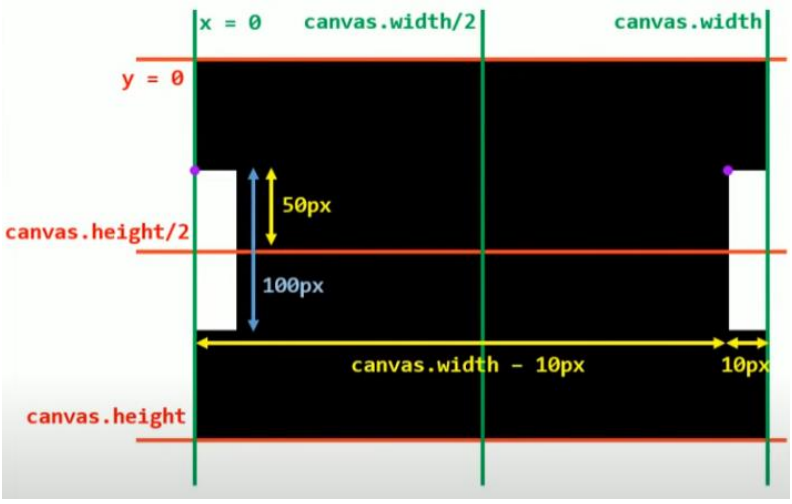


Abbildung 80 Pong Komponentenberechnung

Nachdem wir die Schläger Objekte definiert haben, können wir diese auch Zeichnen (drawRect, Rechteck zeichnen).

```
// Draw the user's paddle
drawRect(user.x, user.y, user.width, user.height, user.color);

// Draw the COM's paddle
drawRect(com.x, com.y, com.width, com.height, com.color);
```

Abbildung 81 Pong draw Funktion

Danach wird das Netz (net) erstellt. Auch hier wurde ein Objekt zuerst definiert:

```
// CREATE THE NET
const net = {
    x : (canvas.width - 2)/2,
    y : 0,
    height : 10,
    width : 2,
    color : "WHITE"
}
```

Abbildung 82 Pong Netz

Fast genau wie bei den Schläger wird auch hier die X- und Y-Position festgestellt, die Höhe, Breite und Farbe. Hier wird das Netz in der Mitte gezeichnet, daher: «(canvas.width – 2) / 2».

Auch hier, um es sich besser vorzustellen, kann man diese Abbildung betrachten:

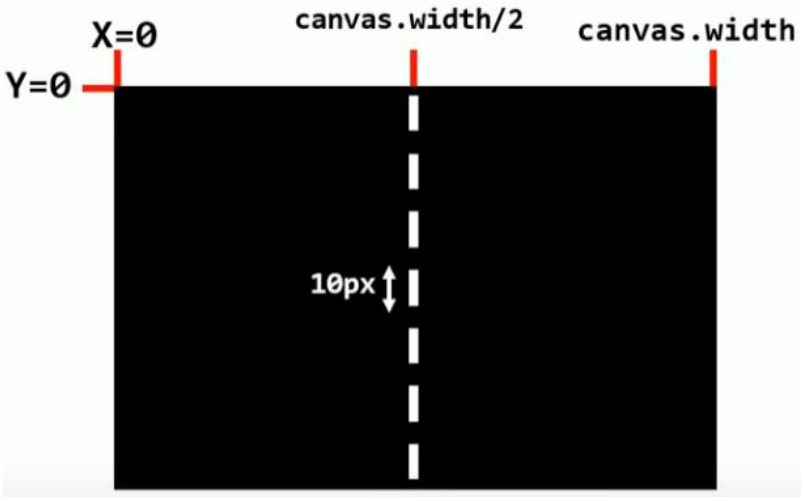


Abbildung 83 Pong Netzberechnung

Um das Netz gestrichelt zu zeichnen, wurde die Funktion «drawNet()» programmiert. Diese Funktion besitzt das gleiche «drawRect» wie bei den Schläger, aber auch eine For-Schleife um die Linie gestrichelt zu zeichnen. Somit zeichnete die Funktion die 10 Pixel hohen Striche und einen Abstand nach jedem Strich.

```
// DRAW NET
function drawNet(){
    for(let i = 0; i <= canvas.height; i+=15){
        drawRect(net.x, net.y + i, net.width, net.height, net.color);
    }
}
```

Abbildung 84 Pong drawNet Funktion

Als Nächstes definieren wir ein Objekt für den Ball:

```
// CREATE THE BALL
const ball = {
    x : canvas.width/2,
    y : canvas.height/2,
    radius : 10,
    velocityX : 5,
    velocityY : 5,
    speed : 7,
    color : "WHITE"
}
```

Abbildung 85 Pong Ball erstellen

Für die erstellen des Balles braucht man nur die X-Y-Position, den Radius und die Farbe. VelocityX, VelocityY und speed (Geschwindigkeit) wird später programmiert. Dennoch habe ich es jetzt schon miteinbezogen im Bild.

Wird hier die Abbildung der Erstellung des Balles betrachtet, sieht man der Ball in der Mitte mit einem Radius von 10 Pixel. Hier wurde ganz einfach für die X- und Y-Position die Hälfte der Canvas genommen.

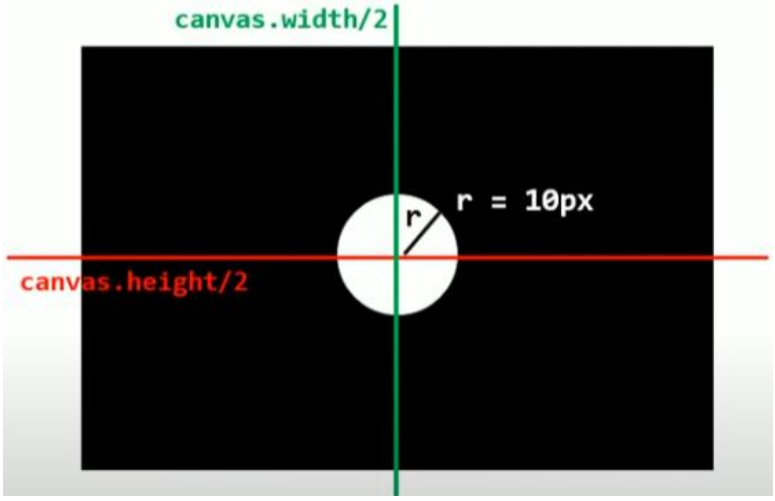


Abbildung 86 Pong Ballberechnung

Auch hier wird der Ball gezeichnet.

```
// Draw the ball
drawArc(ball.x, ball.y, ball.radius, ball.color);
```

Abbildung 87 Pong Ball draw Funktion

Als Letztes wird der Punktestand programmiert. Hier, um es einfacher zu machen, müssen wir auch diese Abbildung betrachten:

Abbildung 88 Pong Punktestandberechnung

Die grüne Striche markieren die Position X, das heisst: «W/4» ist die Canvasbreite geteilt durch vier, «2*W/4» ist die Breite mal zwei geteilt durch vier, «3*W/4» ist die Breite mal drei geteilt durch vier und «4*W/4» ist die Breite mal vier geteilt durch vier.

Die rote Striche markieren die Position Y, wie bei den Grünen Strichen heisst es hier nun: «H/5» ist die Canvas Höhe geteilt durch fünf, «2*H/5» ist die Höhe mal zwei geteilt durch fünf, «3*H/5» ist die Höhe mal drei geteilt durch fünf, «4*H/5» ist die Höhe mal vier geteilt durch fünf und «5*H/5» ist die Höhe mal fünf geteilt durch fünf.

Nun, um bei dieser Position wie bei der Abbildung den Punktestand zu definieren, brauchen wir diese Funktionen:

```
// Draw the user score to the left
drawText(user.score,canvas.width/4,canvas.height/5);
// Draw the COM score to the right
drawText(com.score,3*canvas.width/4,canvas.height/5);
```

Abbildung 89 Pong drawText Funktion

Da jetzt alle Elemente ausgerechnet und programmiert wurden, packen wir diese in einer Funktion ein, die Funktion «render()»:

```
// RENDER THE GAME
function render() {
    // Clear the canvas
    drawRect(0, 0, canvas.width, canvas.height, "#000");
    // Draw the user score to the left
    drawText(user.score,canvas.width/4,canvas.height/5);
    // Draw the COM score to the right
    drawText(com.score,3*canvas.width/4,canvas.height/5);
    // Draw the net
    drawNet();
    // Draw the user's paddle
    drawRect(user.x, user.y, user.width, user.height, user.color);
    // Draw the COM's paddle
    drawRect(com.x, com.y, com.width, com.height, com.color);
    // Draw the ball
    drawArc(ball.x, ball.y, ball.radius, ball.color);
}
```

Abbildung 90 Pong render Funktion

Danach werden alle Elemente wie unten abgebildet gezeichnet:

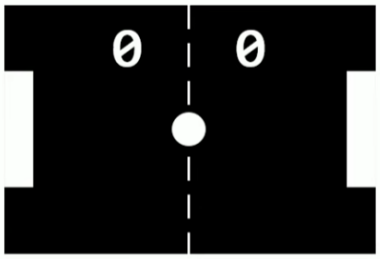


Abbildung 91 Pong Gezeichnete Elemente

Funktion «render()» und «game()»

Die Funktion «render()» wird in der Funktion «game()» eingebettet:

```
// GAME INIT
function game(){

    render();
}
```

Abbildung 92 Pong game Funktion

Zusätzlich definieren wir eine Konstante variabel namens «framePerSecond» mit dem Wert: 50.

```
let framePerSecond = 50;
// LOOP - CALL GAME(); 50 TIMES EVERY 1000MS = 1SEC
let loop = setInterval(game,1000/framePerSecond);
```

Abbildung 93 Pong framePerSecond Funktion

Und binden diese in eine weitere Konstante variabel ein namens «loop» mit der Funktion «setInterval(game,1000/framePerSecond)». Dies heisst: Die Funktion «game» wird 50-mal aufgerufen innerhalb 1000 Millisekunden (1 Sekunde), somit ergibt sich das 50 «Frames per second», auf Deutsch: 50 Bilder pro Sekunde. Das ermöglicht später, dass das Spiel flüssig läuft.

Funktion «update()»

Bei der Funktion «game()» wird zusätzlich eine «update()» Funktion hinzugefügt. Diese Funktion dient für die Bewegungen, Kollisionserkennung, Punktestand aktualisieren usw.

Nun programmieren wir die Bewegung des Balles:

```
// CREATE THE BALL
const ball = {
    x : canvas.width/2,
    y : canvas.height/2,
    radius : 10,
    velocityX : 5,
    velocityY : 5,
    speed : 7,
    color : "WHITE"
}
```

Abbildung 94 Pong Ballbewegung

Wie bei der Objektprogrammierung des Balles gezeigt, kommt nun die «VelocityX», «VelocityY» und «speed» des Balles ins Spiel. Bei Beginn des Spiels ist der Ball in der Mitte des Spielfelds. Daher brauchen wir nun das Attribut «speed» (Geschwindigkeit) des Balles und definieren es als sieben fest.

Hier betrachten wir wieder diese Abbildung, um es einfacher zu machen:

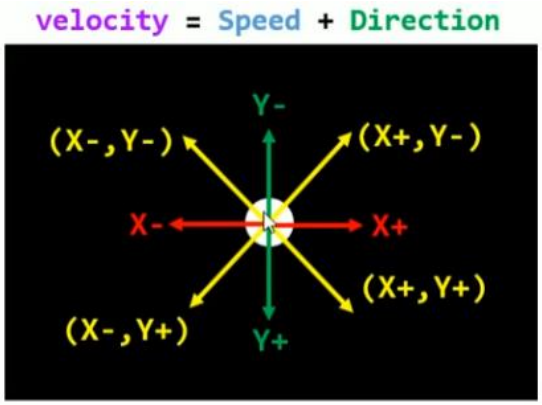


Abbildung 95 Pong Ballbewegungsberechnung

Nun zurück zu der Funktion «update()». Hier programmieren wir als Erstes die Geschwindigkeit für den Ball. Wie oben in der Abbildung dargestellt, wird die Geschwindigkeit beispielsweise: bei «VelocityX» inkrementiert und bei «VelocityY» auch inkrementiert, so bewegt sich der Ball diagonal nach rechts unten. Wird beispielsweise: Bei «VelocityX» inkrementiert und bei «VelocityY» dekrementiert, bewegt sich der Ball diagonal nach rechts oben. Mit dieser Funktion können wir die Richtung des Balles definieren.

```
// Ball Velocity
ball.x += ball.velocityX;
ball.y += ball.velocityY;
```

Abbildung 96 Pong Ballgeschwindigkeit

Danach wird betrachtet, wenn der Ball auf den Seiten des Spielfelds trifft:

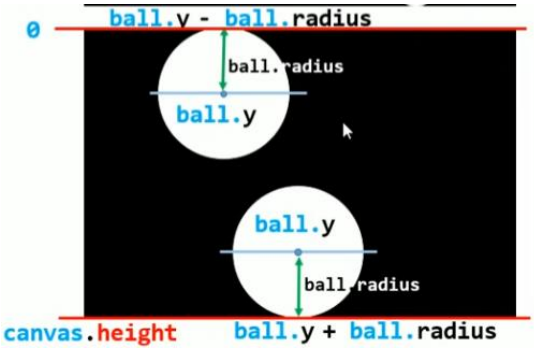


Abbildung 97 Pong Ballberührungsberechnung

Hier wird ein «if-Statement» verwendet. Falls der Ball die Seiten des Spielfelds berührt, wechselt es die Bewegungsrichtung.

```
if(ball.y - ball.radius < 0 || ball.y + ball.radius > canvas.height){
    ball.velocityY = -ball.velocityY;
    wall.play();
}
```

Abbildung 98 Pong Bewegungsrichtung

Kollisionserkennung

Um die Kollusionen zu erkennen, brauchen wir die Funktion «collision(b, p)» (b steht für Ball, p für Player).

Diese Abbildung hilft, die Kollusionen besser zu verstehen:

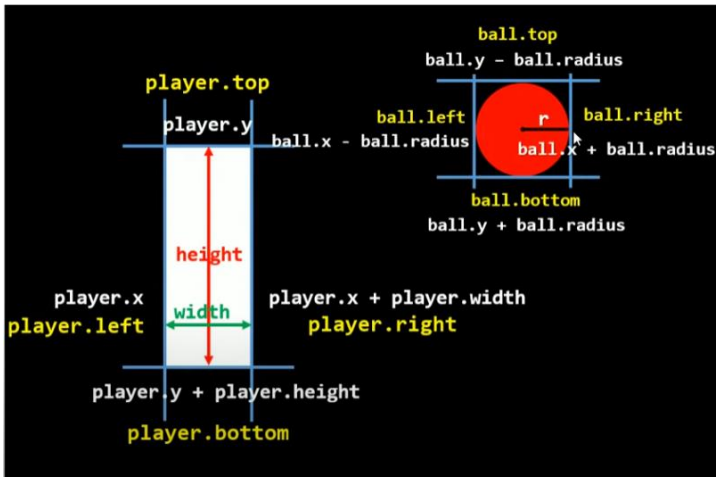


Abbildung 99 Pong Kollusionen

Mittels dieser Abbildung definieren wir alle Seiten des Schlägers und des Balles und schreiben die Funktion «collision(b, p)»:

```
// DETECTING COLLISION (b = BALL, p = PLAYER)
function collision(b,p){
    p.top = p.y;
    p.bottom = p.y + p.height;
    p.left = p.x;
    p.right = p.x + p.width;

    b.top = b.y - b.radius;
    b.bottom = b.y + b.radius;
    b.left = b.x - b.radius;
    b.right = b.x + b.radius;

    return p.left < b.right && p.top < b.bottom && p.right > b.left && p.bottom > b.top;
}
```

Abbildung 100 Pong collision Funktion

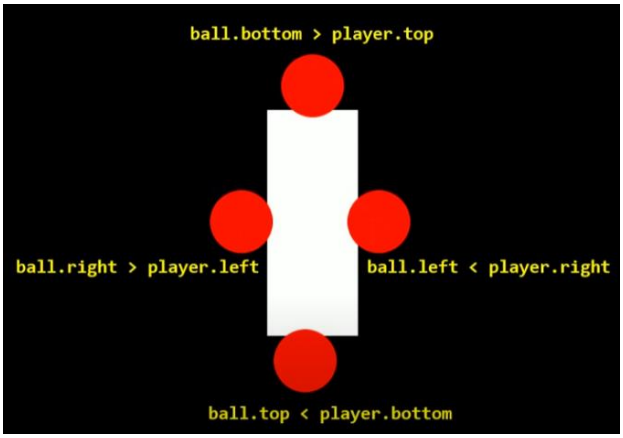


Abbildung 101 Pong collision Funktion

Wir geben das «return» heraus. Wenn die «return» Zeile alle eingegebenen Parameter als «True» definiert, dann besteht eine Kollusion und die Funktion «collision(b, p)» wird «wahr» ausgeben. Falls es diese als «False» definiert, besteht keine Kollusion und die Funktion «collision(b, p)» wird «falsch» ausgeben.

Jetzt müssen wir ein «if-Statement» programmieren, um herauszufinden, ob der User den Ball getroffen hat oder der Com (Computer).

Mit dieser Abbildung sehen wir, dass wenn der Ball sich auf der linken Seite aufhält, der User den Ball getroffen hat. Befindet sich der Ball auf der rechten Seite des Spielfelds, dann hat der Com den Ball getroffen.

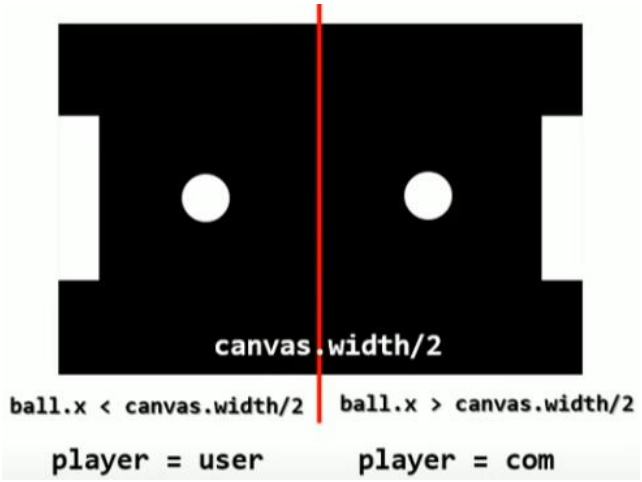


Abbildung 102 Pong Ballposition

Somit wurde diese «let» Variabel «player» definiert:

```
// Check if paddle hit the user or com paddle
let player = (ball.x + ball.radius < canvas.width/2) ? user : com;
```

Abbildung 103 Ballposition

Diese überprüft, ob der User den Ball getroffen hat oder der Com.

if(collision(ball,player))

Für dieses «if-Statement» muss man Mathematik anwenden. Wir müssen definieren, wo der Ball getroffen wurde, in der Mitte, oben oder unten. Wenn der Ball in der Mitte getroffen wurde, wird der Ball in gleicher Richtung bewegen, wenn aber der Ball oben oder unten getroffen wurde, wird der Ball in 45°-Richtung bewegen.

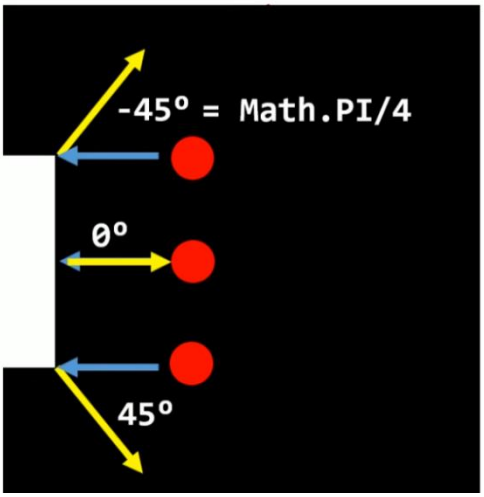


Abbildung 104 Pong Ballbewegungsrichtung

Selbstverständlich könnte man auch die 45° ändern und beispielsweise 30° oder eine beliebige Zahl nehmen. Da ich das originale Pong programmieren möchte, wählte ich 45°.

Um die Bewegungsrichtung des Balles zu verändern, werden diese mathematische Rechnungen durchgeführt:

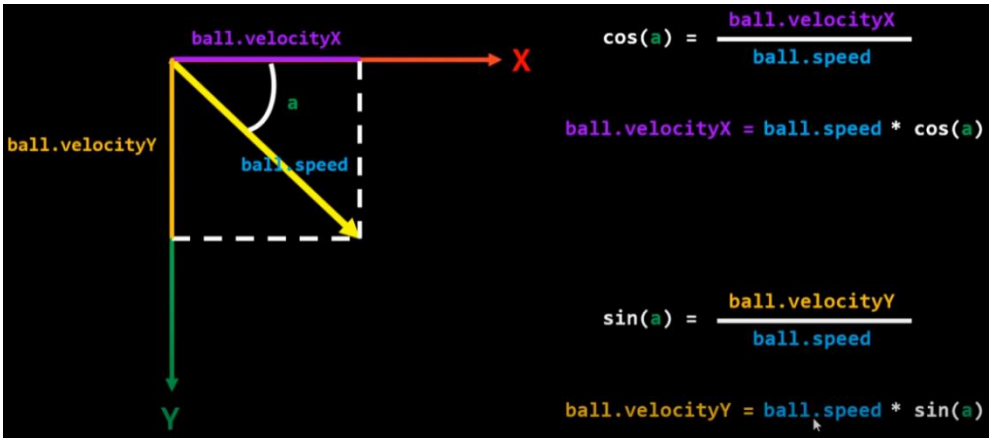


Abbildung 105 Pong Ballbewegungsrichtung Berechnung

Um die Berechnungen durchzuführen, müssen wir die Winkeln berechnen wie in der unteren Abbildung:

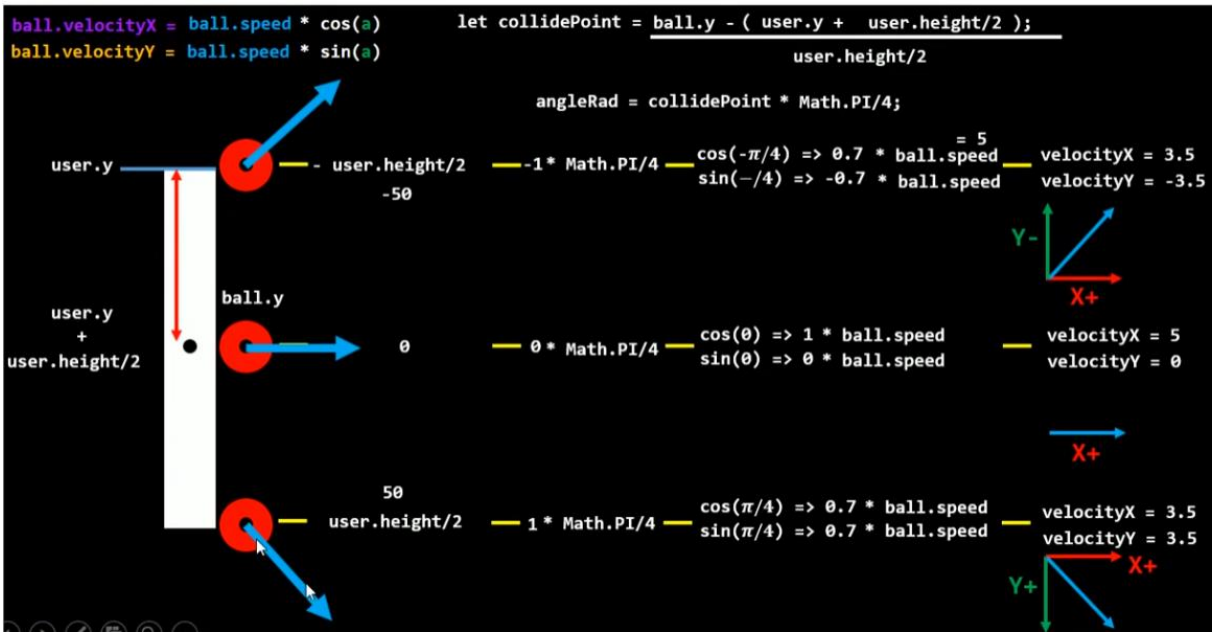


Abbildung 106 Pong Ballbewegungsrichtung Berechnung

Wir sehen das der Schläger, die obere Position «user.y» besitzt und die Mitte des Schlägers «user.y « user.height/2». Wenn jetzt der Ball genau in der Mitte treffen würde, stellen wir fest, dass die Y-Position dieselbe ist, somit ist die Position des Schlägers und des Balles null. Wenn wir der Ball bei dieser Stelle berührt wurde, wissen wir, dass es in der Mitte getroffen wurde.

Wir berechnen nun: «let collidePoint = ball.y – (user.y + user.height/2)» und es gibt null heraus. Wenn es also null ergibt, wissen wir, dass der Ball in der Mitte getroffen wurde.

Wenn der Ball am Schläger oben trifft, gibt es «- user.height/2» aus. Wird der Ball unten getroffen, gibt es «user.height/2». Das heisst, wir werden immer Nummern erhalten zwischen -50 und 50. Wir können diese Nummern normalisieren, das heisst, anstatt Nummern zwischen -50 und 50 zu bekommen, nehmen wir einfach Nummern zwischen -1 und 1. Um das zu berechnen, muss man einfach beim «let collidePoint» noch alles geteilt durch «user.height/2» berechnen, wie auf der Abbildung dargestellt und somit erhalten wir die -1 und 1.

Um die 45° Bewegung zu erhalten, wird bei den Nummern: « * Math.PI/4» berechnet und wir erhalten: «angleRad = collidePoint * Math.PI/4».

Bei der Abbildung sieht man Beispiele der Berechnungen, um zu sehen in welcher Richtung der Ball bewegt wird (blauer Pfeil ist die Ballbewegung).

Jetzt, wenn eine Kollision entsteht, berechnen wir den «collidePoint» (Wo wurde der Ball getroffen) und normalisieren diese Berechnung, um Nummern zwischen -1 und 1 zu erhalten.

```
if(collision(ball,player)){
    // Play Sound
    hit.play();
    // Where the ball hit the player
    let collidePoint = (ball.y - (player.y + player.height/2));
    // Normalization
    collidePoint = collidePoint / (player.height/2);
    // Calculate angle in Radian
    let angleRad = (Math.PI/4) * collidePoint;
    // X direction of the ball when it's hit
    let direction = (ball.x + ball.radius < canvas.width/2) ? 1 : -1;
    ball.velocityX = direction * ball.speed * Math.cos(angleRad);
    ball.velocityY = ball.speed * Math.sin(angleRad);
    // Ball hit's paddle, Ball speed increases
    ball.speed += 0.3;
}
```

Abbildung 107 Pong if collision

Danach wird der «angleRad» berechnet (den Grad, in unserem Fall: 45° nach oben/unten oder 0°). Die «ball.velocityX» und «ball.velocityY» werden dadurch immer wieder geändert. Zusätzlich, damit wir den Computer besiegen können und es für uns herausfordernd wird, wird die Geschwindigkeit des Balles bei jedem Treffen erhöht: «ball.speed += 0.3» (jedes Mal um 0.3 erhöht).

Damit der getroffene Ball die Richtung ändert, beispielsweise: Spieler trifft den Ball, müssen wir «let direction» definieren. Wenn der «ball.x + ball.radius» kleiner als die Canvasbreite geteilt durch zwei ist, was bedeutet, dass der User den Ball getroffen wird, erhalten wir einen Positiven wert: 1. Im Gegenteil, wenn der Computer den Ball getroffen hat, ist die Richtung negativ, daher -1.

Funktion «update()»

Nun aktualisieren wir den Punktestand. Wenn der Ball (rot angegeben) über den Canvas auf der Spielerseite herausgeht, bekommt der Computer ein Punkt. Sollte es auf das Gegenteil kommen und der Ball bei der rechten Canvas Seite herausgeht, bekommt der Spieler ein Punkt.

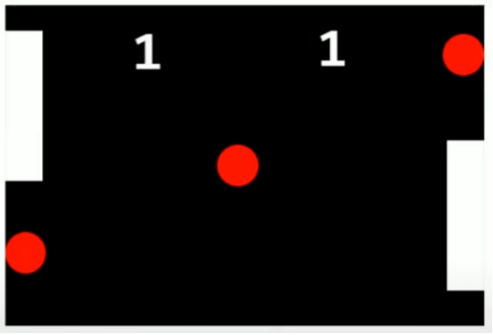


Abbildung 108 Pong Punktestand

Hier wird eine einfache «if» und «else if» Methode angewendet:

```
function update() {
    // Update the score
    if( ball.x - ball.radius < 0 ){
        com.score++;
        comScore.play();
        resetBall();
    }else if( ball.x + ball.radius > canvas.width){
        user.score++;
        userScore.play();
        resetBall();
    }
```

Abbildung 109 Pong update Funktion

Wenn der «ball.x – ball.radius» kleiner als null ist, bekommt der Computer einen Punkt: «com.score++», zusätzlich wird ein Ton gespielt: «comScore.play()» und der Ball muss zurückgesetzt werden. Für die Zurücksetzung des Balles wird die Funktion «resetBall()» erstellt:

```
// RESET BALL
function resetBall(){
    ball.x = canvas.width/2;
    ball.y = canvas.height/2;
    ball.velocityX = -ball.velocityX;
    ball.speed = 7;
}
```

Abbildung 110 Pong resetBall Funktion

Der Ball wird wieder in die Mitte des Spielfelds platziert, die Geschwindigkeit des Balles wird auch zurückgesetzt (Standard 7, da während des Spiels bei jedem Treffer die Geschwindigkeit um 0.3 erhöht wird) und die «velocityX» des Balles wird invertiert, sodass sich der Ball Richtung Computer bewegt.

Bei «else if»: Wenn der Ball auf der rechten Seite des Spielfelds über das Canvas herausgeht, bekommt der Spieler einen Punkt, der Ton wird gespielt und der Ball zurückgesetzt.

Die Funktion «update()» sieht vollständig dementsprechend aus:

```
// UPDATE: POS, MOV, SCORE, ETC.
function update() {
    // Update the score
    if( ball.x - ball.radius < 0 ){
        com.score++;
        comScore.play();
        resetBall();
    }else if( ball.x + ball.radius > canvas.width){
        user.score++;
        userScore.play();
        resetBall();
    }
    // Ball Velocity
    ball.x += ball.velocityX;
    ball.y += ball.velocityY;
    // AI that controls the com paddle
    com.y += ((ball.y - (com.y + com.height/2)))*0.1;

    if(ball.y - ball.radius < 0 || ball.y + ball.radius > canvas.height){
        ball.velocityY = -ball.velocityY;
        wall.play();
    }
    // Check if paddle hit the user or com paddle
    let player = (ball.x + ball.radius < canvas.width/2) ? user : com;

    if(collision(ball,player)){
        // Play Sound
        hit.play();
        // Where the ball hit the player
        let collidePoint = (ball.y - (player.y + player.height/2));
        // Normalization
        collidePoint = collidePoint / (player.height/2);
        // Calculate angle in Radian
        let angleRad = (Math.PI/4) * collidePoint;
        // X direction of the ball when it's hit
        let direction = (ball.x + ball.radius < canvas.width/2) ? 1 : -1;
        ball.velocityX = direction * ball.speed * Math.cos(angleRad);
        ball.velocityY = ball.speed * Math.sin(angleRad);
        // Ball hit's paddle, Ball speed increases
        ball.speed += 0.3;
    }
}
```

Abbildung 111 Pong komplette update Funktion

User- und Computerschläger bewegen

Um das Spiel zu spielen, müssen die Schläger bewegt werden. Eins vom Spieler selbst und der andere vom Computer.

Für den Spieler programmieren wir einen «EventListener», dieser aktiviert, sobald es den dementsprechenden Code erkennt, in unserem Fall: sobald die Maus auf dem Canvas befindet.

```
// CONTROL THE USER PADDLE
canvas.addEventListener("mousemove", getMousePos);

function getMousePos(evt){
    let rect = canvas.getBoundingClientRect();

    user.y = evt.clientY - rect.top - user.height/2;
}
```

Abbildung 112 Pong Schläger bewegen

Um die Mausposition zu erhalten, geben wir den Code: «user.y = evt.clientY» aus. Da diese Position nicht immer stimmen wird und wir ausserhalb des Spielfelds befinden, müssen wir die «let rect = canvas.getBoundingClientRect()» variabel deklarieren. Diese variabel gibt die Positionen: X,Y, oben, Unterseite, links und rechts aus sowie die Höhe und Breite des Canvas.

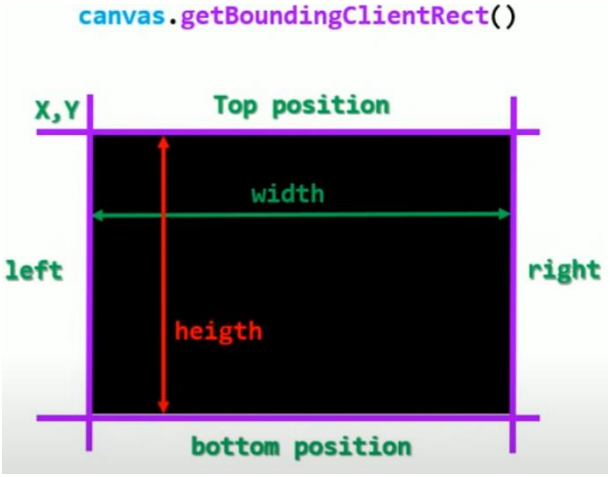


Abbildung 113 Pong Schläger bewegen Berechnung

Damit wir auch die richtige Position haben, müssen wir bei «user.y = evt.clientY» die «- rect.top – user.height/2» einfügen und schlussendlich sieht es dementsprechend aus:

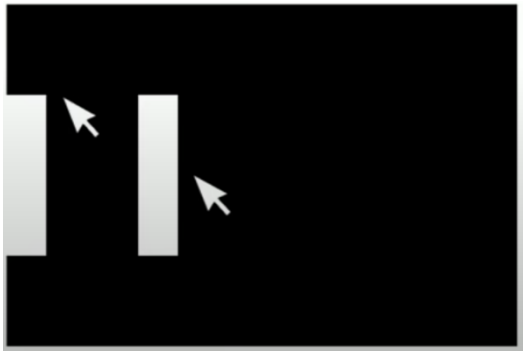


Abbildung 114 Pong Schläger bewegen

Als Letztes fehlt nun die künstliche Intelligenz des Computers:

```
// AI that controls the com paddle
com.y += ((ball.y - (com.y + com.height/2)))*0.1;
```

Abbildung 115 Pong Com KI

Diese Zeile Code wurde in der «update()» Funktion programmiert (siehe Abbildung Funktion «update()»).

Mit dieser Zeile wird die Höhe des Balles genau gleich sein wie die Höhe des Schlägers des Computers. Damit ist der Computer unschlagbar.

Abbildung 116 Pong Com KI

Da wir aber gegen den Computer gewinnen möchten, setzen wir einen «Computerlevel» = 0.1. Die 0.1 wird am Schluss multipliziert und sobald die Geschwindigkeit des Balles erhöht wird, werden wir den Computer besiegen können.

6.6.2 Rock, Paper, Scissors

Die Programmierung des Rock, Paper, Scissors Spiels erfolgte mit der Hilfe und Quelle von «Clever Programmer» auf YouTube. Da wurden Funktionen beschrieben und erklärt, was diese tun.

Die Dateistruktur des Rock, Paper, Scissors Spiels sieht folgendermassen aus:

Es befinden sich drei Dateien, die für die Programmierung bestimmt sind: rps-game.js, rps-style.css, rps.html und den Ordner mit den Bildern.

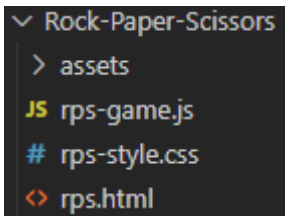


Abbildung 117 RPS Dv.

Die verwendeten Programmiersprachen: rps.html in HTML (Hypertext Markup Language), rps-style.css in CSS (Cascading Style Sheets) und rps-game.js (JavaScript).

HTML – tic-tac-toe.html

In der HTML-Datei befinden sich fast die gleichen Elemente wie beim index.html der Website. Es befinden sich mehrere «Divs» in der HTML-Datei. Navigationsmenü und die Fusszeile blieben gleich.

Es werden die verschiedene Bilder der Symbole geladen und sobald man auf diese drückt, wird eine Funktion ausgelöst.

```
<main class="main">
  <div class="outside-div">
    <div class="wrapper">
      <div class="scoreboard">
        <div class="title">
          <img src="../Rock-Paper-Scissors/assets/title.png" alt="" />
        </div>
        <div class="score">
          <p>SCORE</p>
          <h1>0</h1>
        </div>
      </div>
      <div class="hands">
        <div class="hand paper">
          <img
            src="../Rock-Paper-Scissors/assets/Paper.png"
            onclick="pickUserHand('paper')"
            alt=""
          />
        </div>
```

Abbildung 119 RPS HTML

```
        <div class="hand scissors">
          <img
            src="../Rock-Paper-Scissors/assets/Scissors.png"
            onclick="pickUserHand('scissors')"
            alt=""
          />
        </div>
        <div class="hand rock">
          <img
            src="../Rock-Paper-Scissors/assets/Rock.png"
            onclick="pickUserHand('rock')"
            alt=""
          />
        </div>
      </div>
      <div class="contest">
        <div class="userhand">
          <h1>YOU PICKED</h1>
          <div class="handImageContainer">
            <img
              id="userPickImage"
              src="../Rock-Paper-Scissors/assets/Paper.png"
              alt=""
            />
          </div>
```

Abbildung 118 RPS HTML

CSS – rps-style.css

Die ttt-style.css Datei wurde in der rps.html Datei eingebettet und dient als Stylesheet für das Rock, Paper, Scissors Spiel.

Das Styling des Rock, Paper, Scissors Spiels wurde durch verschiedene Elemente gestylt. Diese CSS-Datei hat die meisten Styles als bei den anderen zwei Spiele: Pong und Tic-Tac-Toe.

```css
.scoreboard {
    width: 700px;
    border: 1px solid white;
    border-radius: 15px;
    margin-top: 50px;
    height: 150px;
    display: flex;
    justify-content: space-between;
    align-items: center;
}
```

Abbildung 121 RPS Scoreboard CSS

```css
.score {
    width: 150px;
    height: 114px;
    background-color: white;
    border-radius: 8px;
    margin-right: 30px;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
```

Abbildung 120 RPS Score CSS

```css
.hands {
    background-image: url("assets/triangle.png");
    background-repeat: no-repeat;
    width: 476px;
    height: 430px;
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    margin-top: 100px;
    background-position: center;
}
```

Abbildung 123 RPS Hands CSS

```css
.hands .hand {
    cursor: pointer;
    transition: all 0.25s;
}
.hands .hand:hover {
    transform: translate3d(0px, -8px, 0px);
}
```

Abbildung 122 RPS Hands CSS

```css
.referee {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
}
.referee h1 {
    font-size: 45px;
}

.newGame {
    background-color: white;
    padding: 12px 24px;
    border-radius: 6px;
    cursor: pointer;
    color: hsl(229, 25%, 31%);
    transition: all 0.15s;
}
.newGame:hover {
    background-color: bisque;
    transform: translate3d(0px, -2px, 0px);
}
```

Abbildung 124 RPS Referee CSS

Verschiedene Menüs wurden mit CSS gestylt und einige «Hover» Effekte eingefügt.

JavaScript –rps-game.js

Das Rock, Paper, Scissors Spiel besitzt am wenigsten Funktionen von allen drei Spiele. Rock, Paper, Scissors wurde ganz einfach programmiert: Wir müssen die «handOptions» (Hand Optionen) definieren. Dies wird durch eine «const» Konstante durchgeführt.

```
const handOptions = {
    "rock": "/Rock-Paper-Scissors/assets/Rock.png",
    "paper": "/Rock-Paper-Scissors/assets/Paper.png",
    "scissors": "/Rock-Paper-Scissors/assets/Scissors.png"
}
```

Abbildung 125 RPS handOptions

Wir haben nun die drei Hand Optionen: Rock, Paper, Scissors und die jeweiligen Bilder der entsprechenden Hände.

Für das Spiel müssen wir die folgenden Punkte beachten:

- Der Spieler kann eine Hand auswählen, entweder Rock (Stein), Paper (Papier) oder Scissors (Schere).
- Der Computergegner wählt eins dieser Hände zufällig.
- Wir brauchen einen Schiedsrichter «Referee», der für uns kontrolliert, ob wir oder der Computer gewonnen hat.
- Um das Spiel neu zu starten, brauchen wir eine Funktion namens «restartGame».

Funktion «pickUserHand()»

Wir beginnen beim Spieler, der eine Hand auswählen will.

```
const pickUserHand = (hand) => {
    // Check in Console what I click
    console.log(hand);

    // Hide the current page
    let hands = document.querySelector(".hands");
    hands.style.display ="none"

    // Show the next page with the hand I picked
    let contest = document.querySelector(".contest");
    contest.style.display = "flex";

    // Set the user pick
        document.getElementById("userPickImage").src = handOptions[hand];

    let cpHand = pickComputerHand();

    referee(hand, cpHand);

}
```

Abbildung 126 RPS pickUserHand Funktion

Die Auswahl wird in der Konsole ausgegeben, um zu überprüfen, auf was wir gedrückt haben. Sobald der Spieler auf eine Hand drückt, werden alle Hände ausgeblendet. Das brauchen wir, um das nächste Menü einzublenden und unsere Hand mit der Hand des Gegners (Computer) zu vergleichen.

Unsere HTML-Datei besitzt viele «Divs» mit verschiedenen Klassen, das hilft uns die richtige Hand auszuwählen und diese speichern. «document.getElementById('userPickImage').src» holt die ausgewählte Hand und setzt diese als ausgewählt.

Danach wird die Hand des Computers ausgewählt und der «Referee» vergleicht beide gespielte Hände und entscheidet, welche von dieser gewonnen hat. Die «pickComputerHand()» und «referee» Funktionen werden wir als Nächstes anschauen.

Funktion «pickComputerHand()»

Eine neue Funktion namens «pickComputerHand» wird als nächstes programmiert. Hier werden wir die Hand des Computers festlegen. Dies passiert zufällig mithilfe der «Math.random()» Funktion.

```
const pickComputerHand = () => {
    // Computer hands options
    let hands = ["rock", "paper", "scissors"]

    // Math.floor = goes downwards to the full number, example: 5.55 goes to 5
    let cpHand = hands[Math.floor(Math.random() * 3)]

    // Set the computer pick (picking random)
    document.getElementById("computerPickImage").src = handOptions[cpHand];

    return cpHand;
}
```

Abbildung 127 RPS pickComputerHand Funktion

Der Computer kann zwischen den drei Händen: «rock», «paper» oder «scissors» auswählen. Das wurde mit der variabel «hands» definiert.

Die Funktion «Math.random()» gibt eine Fliesskomma-Pseudozufallszahl zurück, die grösser oder gleich 0 und kleiner als 1 ist, mit ungefähr gleichmässiger Verteilung über diesen Bereich.

Bei der Berechnung «Math.random()», setzten wir eine drei, da wir drei Elemente in unsere «hands» variabel haben. Dennoch kann man beispielsweise Werte bekommen die 2.5 entsprechen. Wir möchten Ganzzahlen erhalten, daher benutzen wir die «Math.random()» Funktion die «Math.floor» Funktion. Die Funktion «Math.floor()» rundet immer ab und gibt die grösste Ganzzahl zurück, die kleiner oder gleich einer bestimmten Zahl ist.

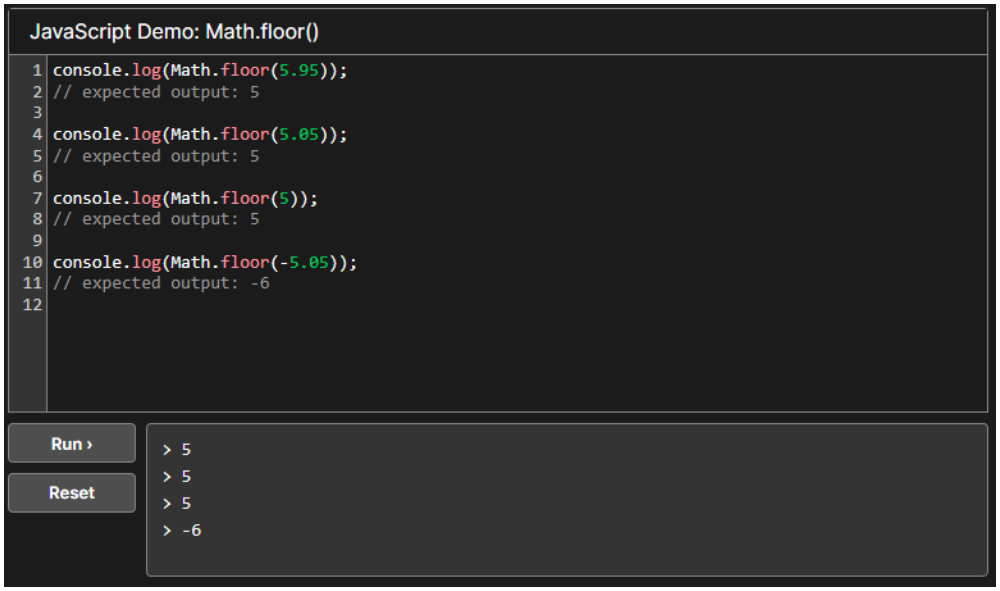


Abbildung 128 RPS Math.floor Demo

Zuerst wird die «Math.random() * 3» Funktion berechnet und anschliessend diese durch die «Math.floor()» Funktion abgerundet. Somit bekommen wir die gewünschte Ganzzahl und dementsprechend unsere Auswahl für den Computer.

Funktion «referee()»

Die Funktion «referee()» vergleicht mit verschiedenen «if» Statements, beide gespielte Hände (Spieler und Computer) und setzt eine Entscheidung: Spieler gewinnt, Spieler verliert oder es ist einen Gleichstand.

```
const referee = (userHand, cpHand) => {
    if(userHand == "paper" && cpHand == "scissors") {
        setDecision("YOU LOSE!")
    }
    if(userHand == "paper" && cpHand == "rock") {
        setDecision("YOU WIN!")
        setScore(SCORE + 1)
    }
    if(userHand == "scissors" && cpHand == "rock"){
        setDecision("YOU LOSE!")
    }
    if(userHand == "scissors" && cpHand == "paper"){
        setDecision("YOU WIN!")
        setScore(SCORE + 1)
    }
    if(userHand == "rock" && cpHand == "paper"){
        setDecision("YOU LOSE!")
    }
    if(userHand == "rock" && cpHand == "scissors"){
        setDecision("YOU WIN!")
        setScore(SCORE + 1)
    }
    if(userHand == "paper" && cpHand == "paper"){
        setDecision("IT'S A TIE!")
    }
    if(userHand == "rock" && cpHand == "rock"){
        setDecision("IT'S A TIE!")
    }
    if(userHand == "scissors" && cpHand == "scissors"){
        setDecision("IT'S A TIE!")
    }
}
```

Abbildung 129 RPS Referee

Bei jedem Entscheid wird eine Entscheidung festgelegt. Die «setDecision» Funktion, zeigt dem Spieler, ob er gewonnen, verloren hat oder ob es einen Gleichstand gibt.

```
const setDecision = (decision) => {
    document.querySelector(".decision h1").innerText = decision;
}
```

Abbildung 130 RPS setDecision Funktion

Gewinnt der Spieler, wird die Punktzahl des Spielers erhöht «setScore(SCORE + 1)». Verliert der Spieler, werden keine Massnahmen getroffen.

```
const setScore = (score) => {
    SCORE = score;
    document.querySelector(".score h1").innerText = score;
}
```

Abbildung 131 RPS setScore Funktion

Um die Punktzahl zu erhöhen, definieren wir eine Standardvariabel ausserhalb der Funktion namens: «let SCORE = 0». Und verweisen diese beim «setScore».

```
let SCORE = 0;
```

Abbildung 132 RPS Score

Funktion «restartGame»

Als Letztes, brauchen wir die Funktion «restartGame», um das Spiel neu zu starten nach jedem Spielzug.

```
const restartGame = () => {
    // Hide the current page -- Same as above but now reversing it
    let hands = document.querySelector(".hands");
    hands.style.display ="flex"

    // Show the next page with the hand I picked -- Same as above but now reversing it
    let contest = document.querySelector(".contest");
    contest.style.display = "none";
}
```

Abbildung 133 RPS restartGame Funktion

Hier werden die Elemente wie bei der «pickUserHand()» Funktion eingeblendet/ausgeblendet (Umkehrung). Da wir bei der CSS-Datei alle Elemente gestylt haben und verschiedene «Divs» erstellten, wird uns der «PLAY AGAIN» Knopf aufgezeigt und sobald wir diesen drücken, wird das Spiel neu gestartet.

6.6.3 Tic-Tac-Toe

Die Programmierung des Tic-Tac-Toe Spiels erfolgte mit der Hilfe und Quelle von «Code Explained» auf YouTube. Da wurden Funktionen beschrieben und erklärt, was diese tun.

Die Dateistruktur des Tic-Tac-Toe Spiels sieht folgendermassen aus:

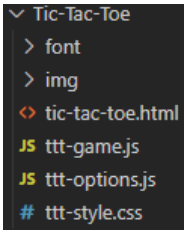


Abbildung 134 TTT Dv.

Es befinden sich vier Dateien und zwei Ordner, die für die Programmierung bestimmt sind: tic-tac-toe.html, ttt-game.js, ttt-options.js, ttt-style.css und die Ordner: img und font.

Die verwendeten Programmiersprachen: tic-tac-toe.html in HTML (Hypertext Markup Language), ttt-style.css in CSS (Cascading Style Sheets), ttt-game.js und ttt-options.js in JavaScript.

HTML – tic-tac-toe.html

In der HTML-Datei befinden sich fast die gleichen Elemente wie beim index.html der Website. Es wurde den «Canvas» (Leinwand, Grafikelemente) Element und einen «Container» hinzugefügt. Navigationsmenü und die Fusszeile blieben gleich.

```
<div class="container">
  <canvas id="cvs" width="450" height="450"></canvas>
  <div class="options">
    <h1>Tic Tac Toe</h1>
    <h2>Play Versus</h2>
    <div class="computer">COMPUTER</div>
    <div class="friend">FRIEND</div>
    <h2>Symbol</h2>
    <div class="x">X</div>
    <div class="o">O</div>
    <div class="play">Play</div>
  </div>
  <div class="gameover hide"></div>
</div>
```

Abbildung 135 TTT HTML

Das Canvas Element wurde definiert mit der jeweiligen ID, Breite und Höhe. Die ID wird verwendet, um bei der ttt-game.js Datei das Canvas zu selektieren.

Die andere Elementen werden diese drei Abbildungen darstellen (mit ttt-style.css und die beiden JavaScript Dateien):

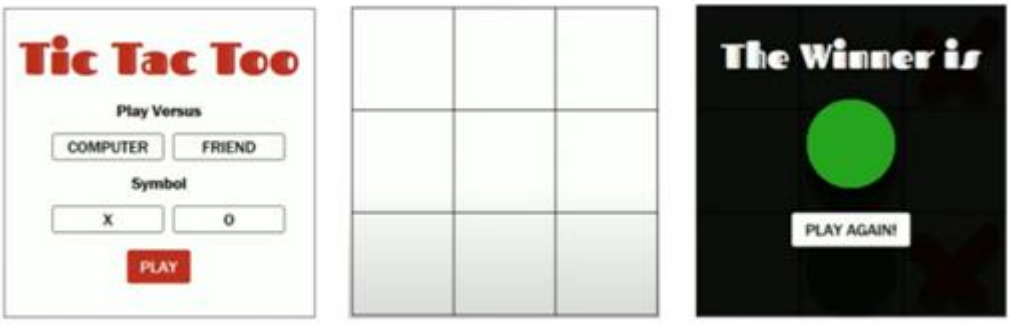


Abbildung 136 TTT Menüs

CSS – ttt-style.css

Die ttt-style.css Datei wurde in der tic-tac-toe.html Datei eingebettet und dient als Stylesheet für das Tic-Tac-Toe Spiel.

In der ttt-style.css Datei befinden sich verschiedene Elemente für das Styling. Es wurde eine Font implementiert und die verschiedene Menüs (Game Over, Start Menü etc.). Zusätzlich wurden auch «hover»(schweben)-Effekte programmiert.

JavaScript –ttt-options.js

Da wir bei der HTML-Datei die verschiedene «Divs» mit verschiedenen «classes» (Klassen) angegeben haben, müssen wir die auch selektieren. Dies erfolgt in der ttt-options.js Datei:

Abbildung 138 TTT Menü

```
// SELECT BUTTONS
const computerBtn = options.querySelector(".computer");
const friendBtn = options.querySelector(".friend");
const xBtn = options.querySelector(".x");
const oBtn = options.querySelector(".o");
const playBtn = options.querySelector(".play");
```

Abbildung 137 TTT Knöpfe

Um auf diese «Buttons» (Knöpfe) drücken zu können, müssen wir einen «EventListener» programmieren. Dieser reagiert, sobald man auf die bestimmte Felder, in unserem Fall die Buttons gedrückt hat.

```
// ADD AND EVENT LISTENER TO EVERY BUTTON
computerBtn.addEventListener("click", function() {
    OPPONENT = "computer";

    switchActive(friendBtn, computerBtn);
})
friendBtn.addEventListener("click", function() {
    OPPONENT = "friend";

    switchActive(computerBtn, friendBtn);
})
xBtn.addEventListener("click", function() {
    player.man = "X";
    player.computer = "O";
    player.friend = "O";

    switchActive(oBtn, xBtn);
})
oBtn.addEventListener("click", function() {
    player.man = "O";
    player.computer = "X";
    player.friend = "X";

    switchActive(xBtn, oBtn);
})
```

Abbildung 139 TTT EventListeners

Sobald diese Buttons gedrückt werden, müssen wir die Buttons speichern. Somit programmieren wir eine Konstante variabel «player» (Spieler) als neuen Objekt und eine «let» variabel «OPPONENT» (Gegner).

```
// VARIABLES TO STORE USER'S OPTIONS
let OPPONENT;
const player = new Object;
```

Abbildung 140 TTT Spieler

Dadurch können wir die verschiedene Werte den Spieler und Gegner zuweisen. Beispiel:

```
xBtn.addEventListener("click", function() {
    player.man = "X";
    player.computer = "O";
    player.friend = "O";

    switchActive(oBtn, xBtn);
})
```

Abbildung 141 TTT Spieler

Hier drückt der Spieler auf das Symbol «X» und wir verweisen dies dem «Player». Der Computergegner bekommt das Symbol «O» und der «player.friend» (Spieler Freund) bekommt auch das Symbol «O».

Die «switchActive(off, on)» Funktion ist mit der ttt-style.css verbunden. Diese wechselt den Status «active» auf die Knöpfe, die man gedrückt hat.

```
// SWITCH ACTIVE CLASS BETWEEN TWO ELEMENTS
function switchActive(off, on) {
    off.classList.remove("active");
    on.classList.add("active");
}
```

Abbildung 142 TTT switchActive Funktion

Abbildung 143 TTT Menü

```
.active{
    background-color: #011627 !important;
    color: #FFF !important;
}
```

Abbildung 144 TTT active CSS

Sollte der Spieler kein Gegner oder Symbol ausgewählt haben und einfach «Play» drückt, werden die Felder rot markiert, um den Spieler aufzubieten, ein Gegner und Symbol zu wählen.

Abbildung 146 TTT Markierung

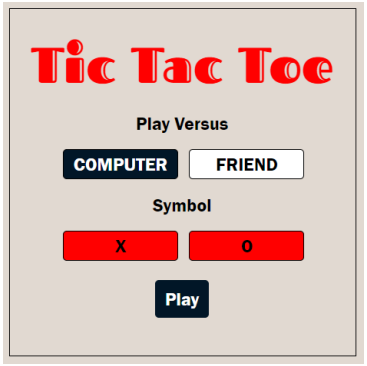


Abbildung 145 TTT Markierung

Um dies umzusetzen, brauchen wir einen weiteren «EventListener», das überprüft, ob der Spieler einen Gegner beziehungsweise ein Symbol ausgewählt hat.

```
playBtn.addEventListener("click", function() {
    // CHECK IF THE USER CHOSE AN OPPONENT
    if ( !OPPONENT){
        computerBtn.style.backgroundColor = "#f00";
        friendBtn.style.backgroundColor = "#f00";

        return
    }
    if ( !player.man){
        xBtn.style.backgroundColor = "#f00";
        oBtn.style.backgroundColor = "#f00";

        return
    }
    init(player, OPPONENT);
    options.classList.add("hide");
})
```

Abbildung 147 TTT EventListener

Sobald der Gegner und das Symbol ausgewählt wurden, kann man nun den «Play» Knopf drücken. Wird der «Play» Knopf gedrückt, verschwindet das Menü (Zeile Code: «options.classList.add('hide')» und das Spielfeld erscheint:

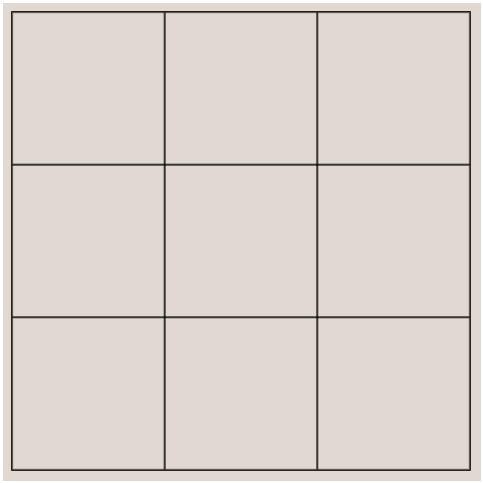


Abbildung 148 TTT Spielfeld

JavaScript – ttt-game.js

Wir selektieren das Canvas. Hierzu habe ich zwei «const» (constant) auf Deutsch Konstanten, die nicht verändert werden, deklariert. Für das Zeichnen des Spielfelds und die weiteren Elemente müssen wir die Konstante «ctx» als «canvas.getContext('2d')» definieren.

```
// SELECT CANVAS
const canvas = document.getElementById("cvs");
const ctx = canvas.getContext("2d");
```

Abbildung 149 TTT Canvas auswählen

Danach programmieren wir die Funktion «init(player, OPPONENT)». Diese Funktion beinhaltet den kompletten Programmiercode in der ttt-game.js Datei (auch die Selektion des Canvas).

Zuerst definieren wir das Spielfeld mit zwei Konstanten:

```
const COLUMN = 3;
const ROW = 3;
```

Abbildung 150 TTT Column, Row

Wir wollen drei Säule und drei Zeilen:

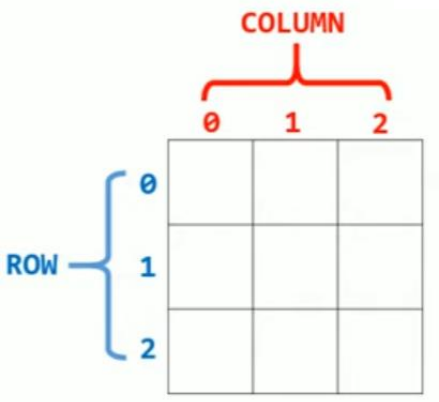


Abbildung 151 TTT Column, Row

Den Abstand der Vierecke müssen wir nun berechnen, da wir das Canvas mit einer Breite von 450 Pixel definiert haben, ergibt die Breite für je einen Viereck 150 Pixel. Dasselbe ergibt auch bei der Höhe, da die komplette Höhe auch mit 450 Pixel definiert wurde.

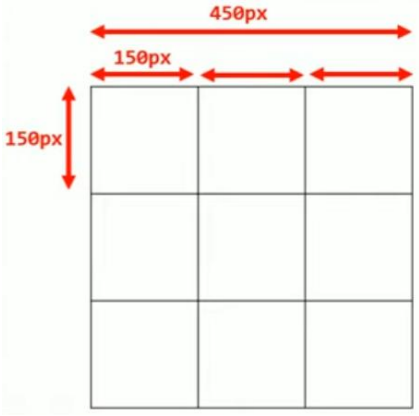


Abbildung 152 TTT Column, Row

Da wir die Säule und Zeilen definieren, haben wir einen «Array» (Datentyp: Feld) und wir programmieren das zusammen mit den Abständen zusammen:

```
// ADDING BOARD VARIABLES
let board = [];
const COLUMN = 3;
const ROW = 3;
const SPACE_SIZE = 150;
```

Abbildung 153 TTT Column, Row

Die Felder werden mittels einen «for-Loop» erstellt. Wir erstellen zusätzlich einen «board» (Tafel) die unser zweidimensionales Array darstellt. Die jeweiligen Felder werden mittels «strokeRect» auf das Spielfeld gezeichnet und besitzen den zugewiesenen Abstand (SPACE_SIZE).

```
for(let i = 0; i < ROW; i++){
    board[i] = [];
    for(let j = 0; j < COLUMN; j++){
        board[i][j] = id;
        id++;

        // draw the spaces
        ctx.strokeStyle = "#000";
        ctx.strokeRect(j * SPACE_SIZE, i * SPACE_SIZE, SPACE_SIZE, SPACE_SIZE);
    }
}
```

Abbildung 154 TTT for-loop

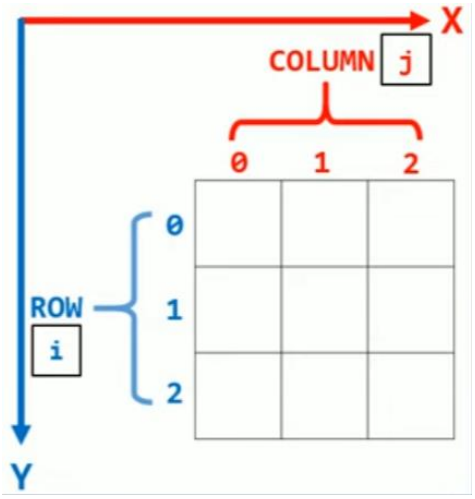


Abbildung 155 TTT Column, Row

Jedes Feld wird eine eigene ID besitzen. Dadurch können wir später auf diese bestimmte Felder zugreifen, beispielsweise: bei der Positionierung der Symbole. Die ID wird im «board» Array gespeichert.

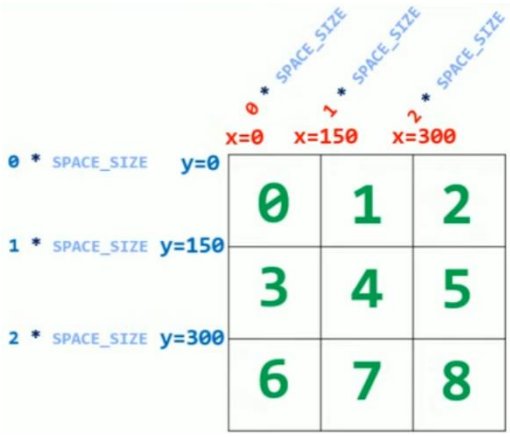


Abbildung 156 TTT Spielfeld ID

Wir packen den «for-Loop» in der «drawBoard()» Funktion und rufen diese am Ende nach der Funktion aus.

```
// DRAWING THE BOARD
function drawBoard(){
    // Every space has a unique ID
    // Therefore I know where the player's move goes on the gameData Array
    let id = 0
    for(let i = 0; i < ROW; i++){
        board[i] = [];
        for(let j = 0; j < COLUMN; j++){
            board[i][j] = id;
            id++;

            // draw the spaces
            ctx.strokeStyle = "#000";
            ctx.strokeRect(j * SPACE_SIZE, i * SPACE_SIZE, SPACE_SIZE, SPACE_SIZE);
        }
    }
}
drawBoard();
```

Abbildung 157 TTT drawBoard Funktion

gameData [ ]

Wir werden uns nicht mit unserem «board» Array befassen, weil es ein zweidimensionales Array ist und dies sehr schwierig sein wird, das umzusetzen. Stattdessen arbeiten wir mit ein eindimensionales Array namens «gameData[]».

Das erste Element dieses Array besitzt den ersten Index 0 und letzten Index 8. Das ist simpel die IDs unsere Felder.

Sobald der Spieler auf ein Feld drückt, müssen wir wissen, welches Feld gedrückt wurde, damit wir das bestimmte Feld mit dem Symbol aktualisieren können Um das zu erreichen, müssen wir den «[i]» und «[j]» ausfindig machen, die wir oben programmiert haben (for-loop), mit der «id» des Feldes.

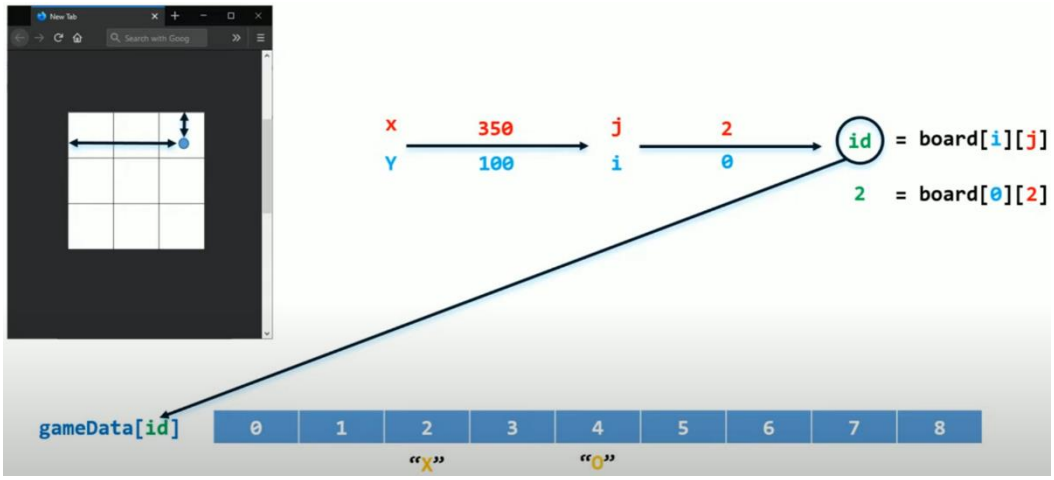


Abbildung 158 TTT gameData

Danach können wir die «X» und «Y» Position bestimmen, auf welchem Feld gedrückt wurde. Wie auf der Abbildung oben wird die «X» und «Y» Position auf «[j]» und «[i]» umgewandelt und diese beim «board» Array aufgerufen. Der «board» Array greift auf die «gameData[id]» und gibt uns diese ID aus.

Mit einem weiteren «EventListener» geben wir die «X» und «Y» Position, wo der Spieler auf dem Browser gedrückt hat. Die Position wird dadurch nicht stimmen, da sich der Canvas irgendwo auf dem Browser befinden kann. Damit wir auf dem Canvas bleiben, schreiben wir diese Zeilen Programmcode:

```
// X & Y position of mouse click relative to the canvas
let X = event.clientX - canvas.getBoundingClientRect().x;
let Y = event.clientY - canvas.getBoundingClientRect().y;
```

Abbildung 159 TTT Mausposition

Die «canvas.getBoundingClientRect().x» und «canvas.getBoundingClientRect().y» Funktionen holen die Daten aus dem Browser, wo sich das Canvas befindet. Somit können wir diese Werte subtrahieren und erhalten die genaue Position von «X» und «Y» im Canvas.

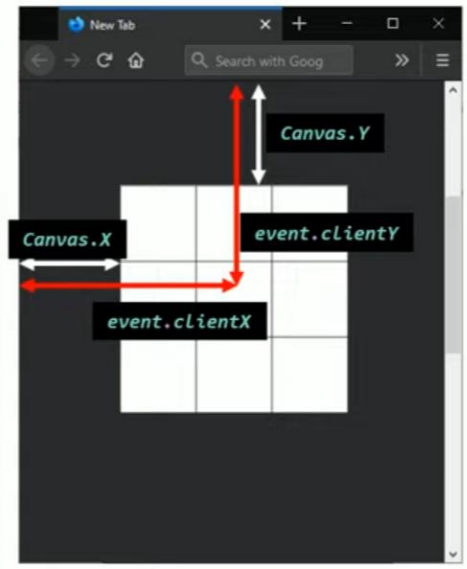


Abbildung 160 TTT Mausposition

Nun wissen wir die gedrückte Position durch den Spieler. Jetzt möchten wir aber das «[i]» und «[j]» dieses Feldes ausfindig machen. Anbei eines Beispiels wird dies erklärt.

Beispiel:

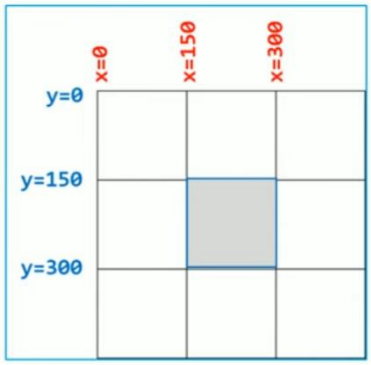


Abbildung 161 TTT Beispiel Feld

Drücken wir auf das Spielfeld in der Mitte.

Bestimmen wir «X» 200 und «Y» 195. Wir haben zuvor «X = j * SPACE_SIZE» und «Y = i * SPACE_SIZE» definiert. Mittels einfache mathematische Anwendung können wir herausfinden, was «j» und «i» ergeben: «j = X / SPACE_SIZE» und «i = Y / SPACE_SIZE». Also geben wir aus: «i» ist 195 geteilt durch 150, das ergibt: «i = 1.3».

Aber für unsere Anzahl der Säulen und Zeilen sind die Felder ganze Zahlen:

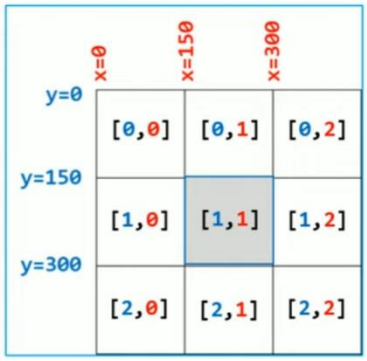


Abbildung 162 TTT Beispiel Feld

Da wir das Feld in der Mitte auswählen möchten, müssen wir das «i = 1.3» abrunden. Dies wird durch den Code: «i = Math.floor(1.3)» ermöglicht und wir erhalten «i = 1».

Dasselbe wird auch beim «j» verwendet. «j = 200 / 150», «j = 1.3333», «j = Math.floor(1.3333)» ergibt «j = 1».

Sobald dies gerechnet wurde, erhalten wir «i = 1» und «j = 1», das bedeutet, wir haben tatsächlich das Feld in der Mitte ([1,1]) gedrückt. Durch diese Anwendung finden wir auch heraus, welche ID das Feld besitzt, und können es mit dem Symbol aktualisieren.

```
let gameData =  new Array(9);
let currentPlayer = player.man;
canvas.addEventListener("click", function(event){

 let X = event.clientX - canvas.getBoundingClientRect().x;
 let Y = event.clientY - canvas.getBoundingClientRect().y;
 let i = Math.floor( Y/SPACE_SIZE );
 let j = Math.floor(X/SPACE_SIZE);
 let id = board[i][j];
 if( gameData[id] ) return;
 gameData[id] = currentPlayer;
```

Abbildung 163 TTT Beispiel gameData

Sodass der Spieler nicht auf ein schon ausgewähltes Feld spielt, programmierten wir die «if (gameData[id]) return;» Zeile. Wenn das Spielfeld nicht leer ist, wird das Returnieren.

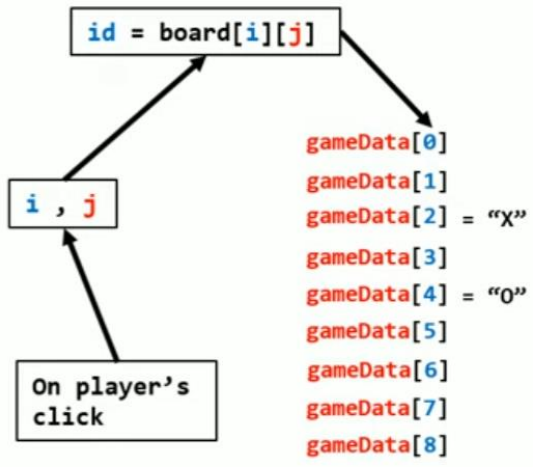


Abbildung 164 TTT Beispiel board

Funktion «drawOnBoard(player, i, j)»

Danach müssen wir nur noch eine Funktion namens «drawOnBoard()» erstellen, um das Spiel zu beenden. Die Funktion «drawOnBoard()» zeichnet ein Bild basieren auf dem Spieler. Wenn es «X» ist, zeichnen wir das X-Bild, wenn es «O» ist, zeichnen wir das O-Bild.

```
// DRAW ON BOARD
function drawOnBoard(player, i, j){
    let img = player == "X" ? xImage : oImage;


    // the x,y position of the image are the x,y of the clicked space
    ctx.drawImage(img, j * SPACE_SIZE, i * SPACE_SIZE);
}
```

Abbildung 165 TTT drawOnBoard Funktion

Dann, nachdem sich der Spieler bewegt, werden wir bestimmen, ob er der Gewinner ist, wenn er der Gewinner ist, zeigen wir den «showGameOver» Element setzen die «GAME_OVER» variabel auf «true» und das Spiel ist somit beendet.

```
// Check if play is winner
if(isWinner(gameData, player.computer)){
    showGameOver(player.computer);
    GAME_OVER = true;
    return;
}
```

Abbildung 166 TTT isWinner Funktion

Die «GAME_OVER» Variable hilft uns, die Gesamtfunktion zu verlassen, sodass niemand einen anderen Zug spielen kann, nachdem das Spiel vorbei ist.

```
// FOR GAME OVER CHECK
let GAME_OVER = false;
```

Abbildung 167 TTT GameOver

Die variabel «GAME_OVER» wird als Standard «false» deklariert, sodass wir spielen können, bis dieser Wert sich verändert.

Wenn es jetzt keinen Gewinner gibt, prüfen wir, ob es ein Unentschieden gibt. Wenn es ein Unentschieden gibt, zeigen wir wieder das «showGameOver» Element und setzen die «GAME_OVER» variabel auf «true».

```
// Check if play is tie
if(isTie(gameData)){
    showGameOver("tie");
    GAME_OVER = true;
    return;
}
```

Abbildung 168 TTT isTie Funktion

Wenn es keinen Gewinner und kein Unentschieden gibt, müssen wir dem anderen Spieler den Zug geben. Wir werden überprüfen, ob der aktuelle Spieler der «player.man» ist, wenn dies der Fall wird, geben wir den «player.friend» den nächsten Zug. Sollte der aktuelle Spieler den «player.friend» sein, geben wir den «player.man» den nächsten Zug.

```
} else {
    // GIVE TURN TO THE OTHER PLAYER
    currentPlayer = currentPlayer == player.man ? player.friend : player.man;
}
```

Abbildung 169 TTT Spielerwechsel

```
let gameData =  new Array(9);
let currentPlayer = player.man;
canvas.addEventListener("click", function(event){
 if(GAME_OVER) return;
 let X = event.clientX - canvas.getBoundingClientRect().x;
 let Y = event.clientY - canvas.getBoundingClientRect().y;
 let i = Math.floor( Y/SPACE_SIZE );
 let j = Math.floor(X/SPACE_SIZE);
 let id = board[i][j];
 if( gameData[id] ) return;
 gameData[id] = currentPlayer;
 drawOnBoard();
 if(isWinner()){
       showGameOver();
       GAME_OVER = true;
       return;
 }
 if(isTie()){
       showGameOver();
       GAME_OVER = true;
       return;
 }
 if( currentPlayer == player.man ){
     currentPlayer = player.friend;
 }else{
     currentPlayer = player.man;
```

Abbildung 170 TTT gameData

«X» und «O» zeichnen

Um das «X» oder «O» auf dem Spielfeld zu zeichnen, müssen wir zuerst die Bilder definieren:

```
// X & 0 images
const xImage = new Image();
xImage.src = "img/X.png";

const oImage = new Image();
oImage.src = "img/O.png";
```

Abbildung 171 TTT X, O

Nun werden auch die Bilder gezeichnet, wie gesehen bei der Funktion «drawOnBoard()»

```
// DRAW ON BOARD
function drawOnBoard(player, i, j){
    let img = player == "X" ? xImage : oImage;


    // the x,y position of the image are the x,y of the clicked space
    ctx.drawImage(img, j * SPACE_SIZE, i * SPACE_SIZE);
}
```

Abbildung 172 TTT drawOnBoard Funktion

Bei der «drawOnBoard()» Funktion stellen wir auch fest, welches Symbol der Spieler «player» besitzt. Wenn der Spieler «X» ausgewählt hat, geben wir ihm den «xImage», ansonsten den «oImage».

Gewinnkombinationen

Wir definieren nun die Gewinnkombinationen:

```
// Win combinations, every possible way to win the game
const COMBOS = [
    [0, 1, 2],
    [3, 4, 5],
    [6, 7, 8],
    [0, 3, 6],
    [1, 4, 7],
    [2, 5, 8],
    [0, 4, 8],
    [2, 4, 6]
];
```

Abbildung 174 TTT Gewinnkombinationen

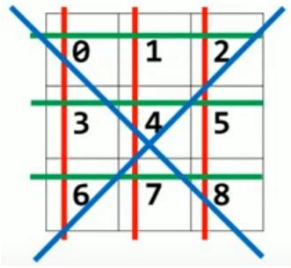


Abbildung 173 TTT Gewinnrichtung

Die Gewinnkombinationen stellen dar, alle Felder IDs, die zu einen Gewinn führen können.

Das alles befindet sich in einer Konstante namens «COMBOS», die ein Array ist, also haben wir hier wieder ein zweidimensionales Array.

Machen wir erneut ein weiteres Beispiel.

Beispiel:

Nehmen wir an, dass das «X» das zweite oder das Feld mit «ID = 2» ist, dann werden wir das «X» auf die «gameData[2]» setzen. Wenn «O» auf «ID = 4» gesetzt wurde, werden wir das «O» auf die «gameData[4]» setzen und so weiter, bis wir eine Reihe von «X» oder «O» haben.

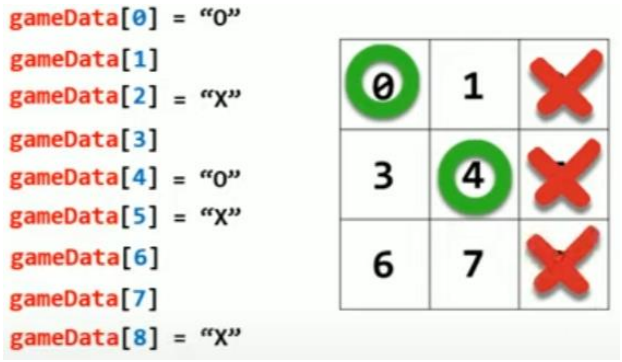


Abbildung 175 TTT gameData ID

Es ist also ganz einfach, jetzt zu überprüfen, ob das «X» der Gewinner ist. Alles, was wir tun müssen, ist, Zeile für Zeile im «COMBOS» Array zu überprüfen und zu prüfen, ob die IDs in den «COMBOS» mit denen in den «gameData[ ]» übereinstimmen, wo das «X» Symbol platziert wurde.

```
const COMBOS = [
    [0, 1, 2],
    [3, 4, 5],
    [6, 7, 8],
    [0, 3, 6],
    [1, 4, 7],
    [2, 5, 8],
    [0, 4, 8],
    [2, 4, 6]
];
```

Abbildung 177 TTT Combos

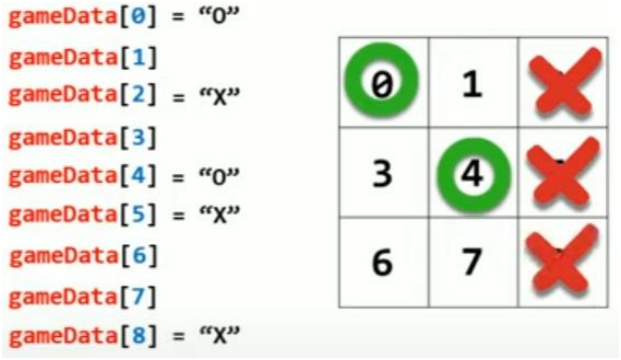


Abbildung 176 TTT gameData

Also gehen wir in die erste Reihe und wir werden die «gameData[ ]» mit Index 0 überprüfen, hat es «X» im Index 0 oder nicht. Wir sehen, dass dies falsch ist, also wird dies «false» zurückgeben. Wir überprüfen die 1, hat diese ein «X», nein auch «false», die 2, diese hat einen «X» also gibt dies «true» heraus. Wir bekommen zweimal «false» und einmal «true», das bedeutet die Gewinnkombination ist «false», «X» kann mit dieser Gewinnkombination gewinnen. Es überprüft dies so, bis es an die Gewinnkombination: «[2, 5, 8]» gelangt. Bei dieser Gewinnkombination wird es dreimal «true» ausgeben und die Gewinnkombination auf «true» setzen. Somit hat «X» gewonnen.

Also werden wir eine «for» Schleife verwenden, die von null bis «COMBOS.length» (COMBOS Länge) beginnt. Wir werden Zeile für Zeile überprüfen, dann werden wir in jeder Zeile die Elemente überprüfen, die die IDs aus unseren Feldern sind, also erstellen wir eine weitere «for» Schleife, diese wird nach den Elementen innerhalb der Zeilen suchen.

```
const COMBOS = [
        [0, 3, 6],
        [1, 4, 7],
        [2, 5, 8],
        [0, 1, 2],
        [3, 4, 5],
        [6, 7, 8],
        [0, 4, 8],
        [2, 4, 6]
];

function isWinner( player, gameData ){
   for(let i = 0; i < COMBOS.length; i++){
      let won = true;
      for(let j = 0; j < COMBOS[i].length; j++){
          let id =  COMBOS[i][j];
          won = gameData[id] == player && won;
      }
      if(won){
          return true;
      }
   }
   return false;
}
```

Abbildung 178 TTT isWinner Funktion

Funktion «isTie(gameData)»

Für die Funktion bei einem Gleichstand ist das Prinzip genau gleich wie beim Gewinner.

```
function isTie( gameData ){
    let isBoardFill = true;

    for(let i = 0; i < gameData.length; i++){
        isBoardFill = gameData[i] && isBoardFill;
    }

    if(isBoardFill){
            return true;
    }

    return false

}
```

Abbildung 180 TTT isTie Funktion

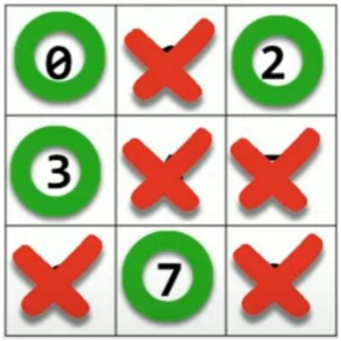


Abbildung 179 TTT Gleichstand

Funktion «showGameOver()»

Die Funktion «showGameOver()» überprüft, ob der Spieler gewonnen hat oder verloren und gibt die entsprechende Menüs hervor.

Abbildung 181 TTT GameOver Menüs

```
// SHOW GAME OVER
function showGameOver(player){
    let message = player == "tie" ? "No Winner" : "The Winner is";
    let imgSrc = `img/${player}.png`;

    gameOverElement.innerHTML = `
        <h1>${message}</h1>
        <img class="winner-img" src=${imgSrc} </img>
        <div class="play" onclick="location.reload()">Play Again!</div>`;

    gameOverElement.classList.remove("hide");
}
```

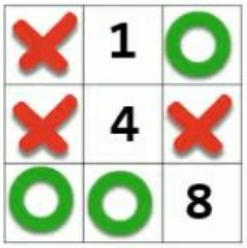
Abbildung 182 TTT showGameOver Funktion

MiniMax Algorithm – Computer Erklärung

Zu guter Letzt müssen wir den Computer programmieren. Der Computer wird mittels der MiniMax Algorithmus programmiert.

Nehmen wir an, jetzt ist der Computer an der Reihe und er muss herausfinden, welches der leeren Felder am besten für ihn ist zu spielen. Der Computer spielt mit «X» und der Spieler spielt mit «O»:

Abbildung 183 TTT MiniMax




```
EMPTY_SPACES = [1, 4, 8];
```

Abbildung 184 TTT Empty_Spaces

Das Erste, was der Computer tun wird, wird nach den leeren Stellen suchen, weil er keine Augen hat. Jetzt sind die IDs der leeren Felder: 1, 4 und 8. In Bezug auf den Code erstellt er also ein Array mit dem Namen: «EMPTY_SPACES = [1, 4, 8]», dass alle IDs aller leeren Felder enthalten.

Jetzt muss der Computer den besten Zug finden, der ihn zum Sieg oder zumindest zu einem Unentschieden bringt. Es ist ganz einfach, der Computer spielt alle leeren Felder nacheinander ab und sieht sich das Ergebnis jedes Zuges an, dann entscheidet er, welcher der Beste ist.

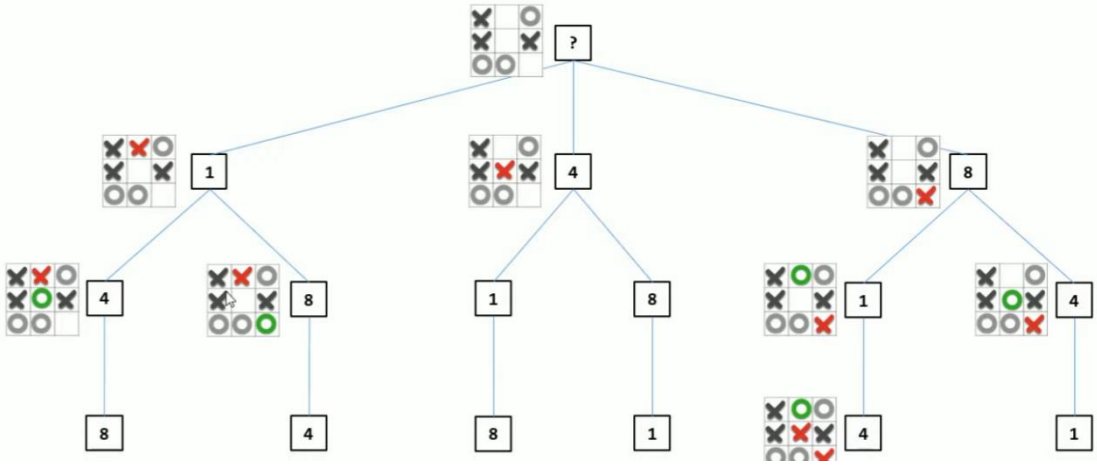


Abbildung 185 TTT MiniMax Alogirthm

Also geht er und spielt 1, der erste leere Feld mit der ID 1. Jetzt kann er sich noch nicht entscheiden, ob es ein guter oder schlechter Zug ist. Er muss noch für den Gegner spielen, in diesem Fall den Spieler. Bevor er das tut. Muss er sich die leeren Felder noch einmal ansehen. Die «EMPTY_SPACES» sind nun 4 und 8. Das bedeutet, dass der Spieler einen von ihnen wählen kann und der Computer das Ergebnis des Spielers überprüfen muss, 4 und 8. Hier merkt der Computer, dass beide Züge der Spieler am Gewinnen ist. Somit wird der Computer diese Felder auf die linke Seite der Abbildung nicht spielen.

Jetzt muss der Computer von vorne Anfangen und den nächsten Zug machen, die 4. Hier gewinnt er und muss nicht für den Spieler spielen.

Als Letztes überprüft er noch das Feld mit der ID 8. Hier merkt der Computer, wenn der Spieler 1 spielt, kann er immer noch gewinnen, wenn er 4 auswählt. Nimmt aber der Spieler 4, dann kann der Computer nicht mehr gewinnen.

Jedes Mal, wenn der Computer einen Zug macht, muss er sehen, ob er das Spiel gewinnt oder verliert. Gute Züge validiert er als «+10» und die schlechte Züge als «-10».

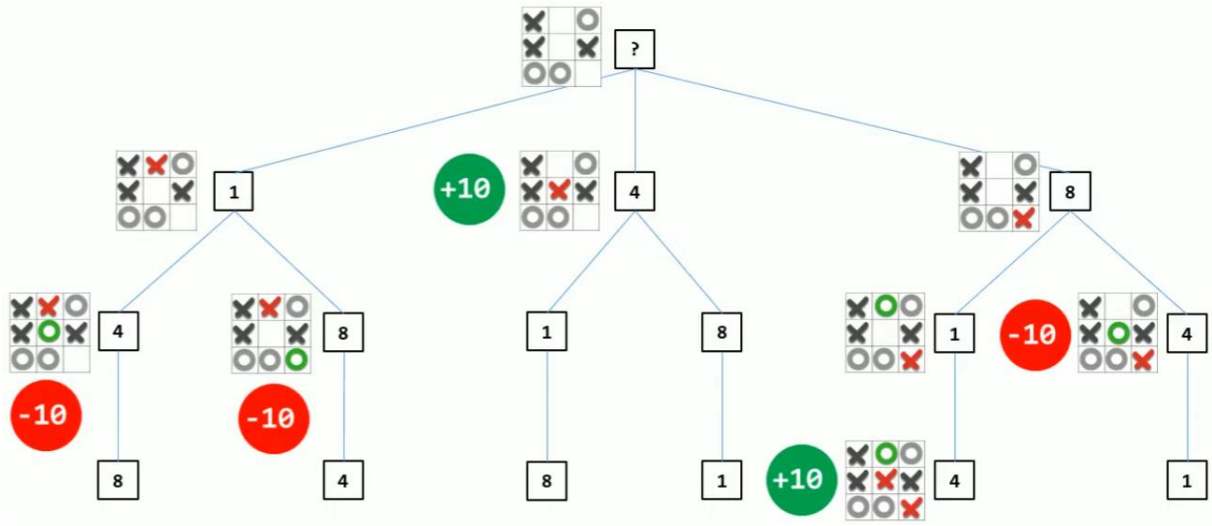


Abbildung 186 TTT MiniMax Algorithm

Der Computer wird validieren, welchen Weg für ihn am besten (maximize) und welchen am schlechtesten (minimize) ist. Sein Ziel ist es immer zu gewinnen.

Auf der rechten Seite der Abbildung sieht man die Addition und Subtraktion von Punkten. Wenn man die linke Seite der Abbildung betrachtet, wird man merken, dass der Balken sich zwischen -10 und +10 bewegen wird.

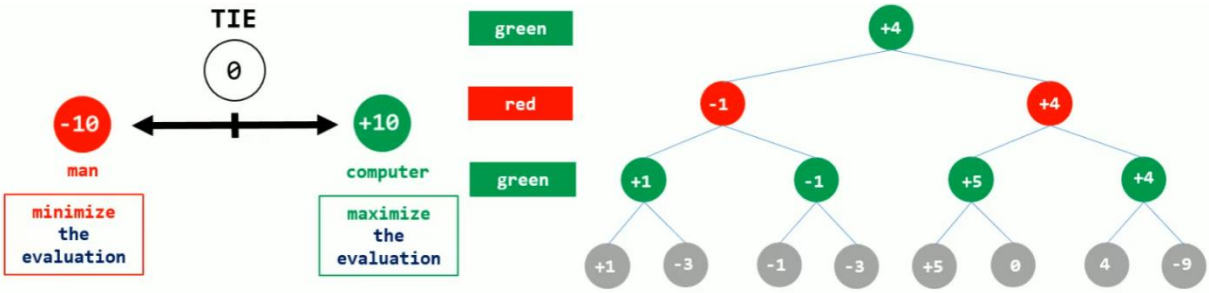


Abbildung 187 TTT MiniMax Algorithm

Vergleichen wir die verschiedenen Wege, wird festgestellt, dass die bestmögliche Option das Feld mit der ID 4 für den Computer ist.

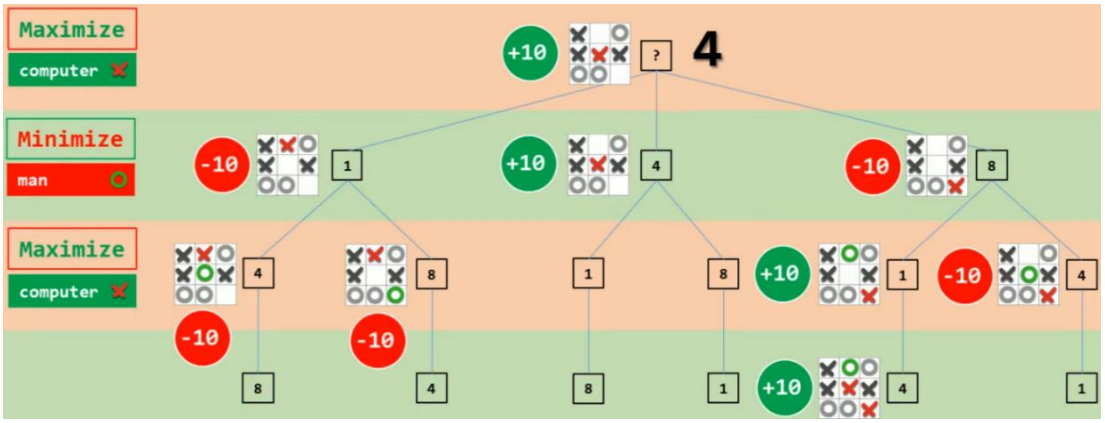


Abbildung 188 TTT MiniMax Algorithm

MiniMax Algorithm – Computer Programmcode

```
// MINIMAX
function minimax(gameData, PLAYER){
    // BASE
    if( isWinner(gameData, player.computer) ) return { evaluation : +10 };
    if( isWinner(gameData, player.man)      ) return { evaluation : -10 };
    if( isTie(gameData)                     ) return { evaluation : 0 };
    // LOOK FOR EMPTY SPACES
    let EMPTY_SPACES = getEmptySpaces(gameData);
    // SAVE ALL MOVES AND THEIR EVALUATIONS
    let moves = [];
    // LOOP OVER THE EMPTY SPACES TO EVALUATE THEM
    for( let i = 0; i < EMPTY_SPACES.length; i++){
        // GET THE ID OF THE EMPTY SPACE
        let id = EMPTY_SPACES[i];
        // BACK UP THE SPACE
        let backup = gameData[id];
        // MAKE THE MOVE FOR THE PLAYER
        gameData[id] = PLAYER;
        // SAVE THE MOVE'S ID AND EVALUATION
        let move = {};
        move.id = id;
        // THE MOVE EVALUATION
        if( PLAYER == player.computer){
            move.evaluation = minimax(gameData, player.man).evaluation;
        }else{
            move.evaluation = minimax(gameData, player.computer).evaluation;
        }
        // RESTORE SPACE
        gameData[id] = backup;
        //SAVE MOVE TO MOVES ARRAY
        moves.push(move);
    }
    // MINIMAX ALGORITHM
    let bestMove;
    if(PLAYER == player.computer){
        // MAXIMIZER
        let bestEvaluation = -Infinity;
        for(let i = 0; i < moves.length; i++){
            if( moves[i].evaluation > bestEvaluation ){
                bestEvaluation = moves[i].evaluation;
                bestMove = moves[i];
            }
        }
    }else{
        // MINIMIZER
        let bestEvaluation = +Infinity;
        for(let i = 0; i < moves.length; i++){
            if( moves[i].evaluation < bestEvaluation ){
                bestEvaluation = moves[i].evaluation;
                bestMove = moves[i];
            }
        }
    }
    return bestMove;
}
```

Abbildung 189 TTT Computer Programmcode

Wir werden die Funktion «minimax(gameData, PLAYER) erstellen, die Parameter sind «gameData» und der «PLAYER». Das Erste, was in unserer Minimax-Funktion ist, ist die Basis. Hier sagen wir, wenn der Gewinner der «player.man» ist, geben wir die Bewertung «-10» zurück. Wenn der Spieler der «player.computer» ist, dann ist die Bewertung +10. Bei Gleichstand ist die Wertung null.

```
// BASE
if( isWinner(gameData, player.computer) ) return { evaluation : +10 };
if( isWinner(gameData, player.man)      ) return { evaluation : -10 };
if( isTie(gameData)                     ) return { evaluation : 0 };
```

Abbildung 190 TTT Base

Dann suchen wir als Nächstes nach leeren Feldern in unserem «gameData» Array, indem wir die Funktion namens «getEmptySpaces» verwenden.

```
// LOOK FOR EMPTY SPACES
let EMPTY_SPACES = getEmptySpaces(gameData);
```

Abbildung 191 TTT Empty_Spaces

```
// GET EMPTY SPACES
function getEmptySpaces(gameData){
    let EMPTY = [];

    for( let id = 0; id < gameData.length; id++){
        if(!gameData[id]) EMPTY.push(id);
    }

    return EMPTY;
}
```

Abbildung 192 TTT getEmptySpaces Funktion

Wir erstellen nochmals ein neues Array namens «moves». Darin werde die gemachte Züge gespeichert, damit der Computer diese validieren kann.

```
// SAVE ALL MOVES AND THEIR EVALUATIONS
let moves = [];
```

Abbildung 193 TTT moves Array

Wir erstellen nun eine «for» Schleife, um alle leeren Felder zu durchlaufen. Wir müssen die ID abrufen, dann müssen wir die ID und die Auswertung in einer Variablen namens «move» speichern. «move.id = id», um die ID zu speichern. Wir speichern es erneut als «backup». Wir wollen nicht mit dem normalen Array herumspielen, also rufen wir am Ende des Prozesses den Raum des Feldes zurück, was bedeutet, dass wir ihn wieder so herstellen, wie er war. Nach dem «backup» wird es diesen Raum für den Spieler spielen und um den Zug auszuwerten, verwenden wir eine «if» Anweisung. Wenn der Spieler der «player.computer» ist, beginnen wir die Funktion «minimax» mit «player.man», andernfalls, wenn der Spieler der «player.man» war, nennen wir den «minimax» mit dem «player.computer.

```
// LOOP OVER THE EMPTY SPACES TO EVALUATE THEM
for( let i = 0; i < EMPTY_SPACES.length; i++){
    // GET THE ID OF THE EMPTY SPACE
    let id = EMPTY_SPACES[i];

    // BACK UP THE SPACE
    let backup = gameData[id];

    // MAKE THE MOVE FOR THE PLAYER
    gameData[id] = PLAYER;

    // SAVE THE MOVE'S ID AND EVALUATION
    let move = {};
    move.id = id;
    // THE MOVE EVALUATION
    if( PLAYER == player.computer){
        move.evaluation = minimax(gameData, player.man).evaluation;
    }else{
        move.evaluation = minimax(gameData, player.computer).evaluation;
    }

    // RESTORE SPACE
    gameData[id] = backup;

    //SAVE MOVE TO MOVES ARRAY
    moves.push(move);
}
```

Abbildung 194 TTT Empty_Spaces loop

Also, wenn wir das erste Mal die Funktion «minimax» mit dem Computer aufrufen, wird er gehen und nach einem Gewinner suchen. Wenn es keinen gibt, wird er nach leeren Felder suchen und dann die leeren Felder durchlaufen. Sobald er zur «if» Anweisung kommt, jetzt ist der Spieler der player.man, ruft der Computer jetzt die Minimax-Funktion mit player.man auf. Danach gehen wir zurück zum Überprüfen des Gewinners. Wenn es keinen Gewinner gibt, suchen wir nach leeren Feldern, führen die vollständige Schleife erneut aus und jetzt ist der Spieler: «player.man» dran, also wird der «minimax» player.man aufgerufen. Der Computer geht wieder hoch und wenn es einen Gewinner gibt, geht die Wertung zurück auf die andere Funktion.

Am Ende der Schleife stellen wir unser Feld wieder her und schieben unseren «move» in unser Array «moves», dass wir ausserhalb unserer «for» Schleife erstellt haben.

Um die besten Züge auszuwählen, benötigen wir unseren «Minimax-Algorithmus». Wir erstellen eine Variable namens «let bestMove» und eine «if» Anweisung. Also, wenn der Spieler «player.computer» ist, dann, wird es der «MAXIMIZER» (Maximierer) sein, andernfalls, wenn es «player.man» ist, wird es der «MINIMIZER» (Minimierer) sein. Wir müssen in unser «moves» Array zugreifen und nachschauen, welches die beste Bewertung hat (-Infinity steht für eine kleine Zahl, +Infinity für eine grosse Zahl).

```
// MINIMAX ALGORITHM
let bestMove;

if(PLAYER == player.computer){
    // MAXIMIZER
    let bestEvaluation = -Infinity;
    for(let i = 0; i < moves.length; i++){
        if( moves[i].evaluation > bestEvaluation ){
            bestEvaluation = moves[i].evaluation;
            bestMove = moves[i];
        }
    }
}else{
    // MINIMIZER
    let bestEvaluation = +Infinity;
    for(let i = 0; i < moves.length; i++){
        if( moves[i].evaluation < bestEvaluation ){
            bestEvaluation = moves[i].evaluation;
            bestMove = moves[i];
        }
    }
}

return bestMove;
```

Abbildung 195 TTT MiniMax Algorithm

Wir verwenden eine «for» Schleife und schauen nacheinander in alle «moves» des Arrays. Dann eine «if» Anweisung, wenn der erste Index 0 in Zügen eine grössere Bewertung hat als die beste Bewertung, dann möchte ich diese verwende und diese als beste Bewertung speichern («bestMove»). Dann beginnt die Schleife erneut und überprüft den Index 1, dann den Index 2 und so weiter, bis «bestMove» nicht grösser als die beste Bewertung «bestEvaluation» ist.

Am Ende wird den «bestMove» übergeben.

7 Abschlussphase

7.1 Zielbeurteilung

Die Ziele werden hier wieder aufgelistet und überprüft, ob diese erreicht wurden. Sollte ein Ziel nicht erreicht worden sein, wird erklärt, warum das Ziel nicht erreicht wurde. Es wurden die Ziele der Zielscheibe und des Pflichtenheftes für diese Zielbeurteilung verwendet.

Ziel	Ziel erreicht?	Beurteilung
Die Spieler/innen können auf die Website zugreifen, Inhalte der Website ansehen und die Spiele spielen.	☑	Die Programmierung der Website und Spiele wurde erfolgreich durchgeführt. Die notwendigen Funktionen sind vorhanden.
Website läuft über localhost (lokalem Host).	☑	Mit Node JS und das Modul «Express» konnte einen lokalen Server hergestellt und gestartet werden.
Spieler/innen können ein Konto eröffnen und sich einloggen.	☑	Mittels Node JS und das Modul «Mongoose» werden die Anmeldedaten der Spieler/innen auf der MongoDB Datenbank gespeichert und beim Log-in abgerufen. Die Spieler/innen gelangen nach einem erfolgreichen Log-in in den Mitgliedsbereich.
Drei programmierte Spiele, die mittels Vanilla JavaScript programmiert wurden: Rock-Paper-Scissors, Tic-Tac-Toe und Pong.	☑	Die Spiele funktionieren fehlerfrei. Jedes Spiel besitzt eine Spielanleitung und einen Computergegner.
Die Website besitzt über eine schnelle Performance.	☑	Die Website lädt innerhalb von maximal 5 Sekunden.
Inhaltssprachen: Deutsch für die Website, Englisch für die Spiele.	☑	Website ist komplett auf Deutsch geschrieben und die Spiele auf Englisch.

Ziel	Ziel erreicht?	Beurteilung
Die Spiele (sofern der Benutzer ein Konto besitzt) speichert den Highscore des Benutzers auf der MongoDB Datenbank.	☒	Aus zeitlichen und Erfahrungsgründe wurde dieses Ziel nicht erreicht. Die Spiele Pong und Rock, Paper, Scissors besitzen einen aktuellen Punktestand.
Kann-Ziel	**Ziel erreicht?**	**Beurteilung**
Die Website besitzt eine «Dunkel-Modus» Einstellung.	☒	Aus zeitlichen Gründen war es nicht möglich, diese Funktion zu implementieren.
Die Website besitzt ein Kontaktformular.	☑	Durch das Navigationsmenü oder der Fusszeile, gelangt man zu der Webseite, wo sich das Kontaktformular befindet. Die Kontaktanfragen werden in der MongoDB Datenbank gespeichert.
Die Spiele können umgeschaltet werden (Spielmodus), sodass man gegen einen weiteren Spieler spielen kann (lokal).	☒	Dieses Ziel wurde teilweise erreicht. Diese Funktion ist nur beim Spiel: Tic-Tac-Toe möglich.
Die Spiele besitzen einen Highscore (Bestenliste) der die Top 10 Benutzer anzeigt.	☒	Aus zeitlichen Gründen war es nicht möglich, diese Funktion zu implementieren.
Die Spiele werden ergänzt: Jump & Run Spiel, Memory Spiel, Spielautomat (Slot Machine), Embedded Games.	☒	Aus zeitlichen Gründen war es nicht möglich, diese Funktionen zu implementieren

Tabelle 2 Zielbeurteilung

7.2 Lessons Learned

Eins meiner wichtigsten Lessons Learned, die ich feststellen musste, war Geduld zu haben. Während der Programmierung stoss ich regelmässig auf kleinere und grössere Problemstellungen. Dabei nahmen einige Problemstellungen mehr Zeit in Anspruch, als die anderen, um diese zu lösen.

Bei dieser Diplomarbeit konnte ich Node JS und das dazugehörige Hauptmodul Express zum ersten Mal einsetzen. Während der Programmierung kamen weitere Module wie zum Beispiel: Bcryptjs und Mongoose zum Einsatz. Die Anwendung von Node JS und der Module, ermöglichte es mir ein grosser Einblick in die Programmierung von einem Backend zu werfen und eins für meine erste Website zu erstellen.

Anschliessend zum Backend, konnte ich die Datenbank MongoDB einsetzen und Erfahrungen aufnehmen. Zuvor kannte ich nur die MSSQL (Microsoft SQL Server) Datenbank, die wir im Fachunterricht Datenbankdesign SQL erlernten. Spannend an der MongoDB ist die Anwendung von JSON Einträge, die in einer NoSQL Datenbank gespeichert werden, was wiederum sehr verschieden ist als bei der MSSQL Datenbank.

Mathematik ist nicht eins meiner grössten Stärken, dennoch gelang es mir, mathematische Anwendungen beziehungsweise Berechnungen in dem JavaScript programmierten Spielen anzuwenden und implementieren. Dabei entstand mein erster Algorithmus namens «MiniMax Algorithm» und somit bekam ich einen Einblick in die Programmierung von künstlicher Intelligenz.

Während der Diplomarbeit merkte ich, dass aufschreiben und notieren von verschiedenen Punkten sehr wichtig ist. Notizen zu schreiben, verhindert wichtige Merkmale während der Praxisarbeit zu verlieren, da viele verschiedene Problemstellungen auftauchen.

Dementsprechend fiel mir auf, dass ich viel Programmcode doppelt schrieb. Bei der Programmierung von HTML und CSS schrieb ich gleiche Elemente mit verschiedenen Klassennamen und stylte diese gleich, aber doppelt. Richtung Ende der Diplomarbeit habe ich einige Änderungen an den Programmcode durchgeführt, Elemente, die gleich gestylt wurden, bekamen die gleichen Klassennamen, um in den CSS-Dateien dementsprechend weniger doppelter Code zu erhalten.

Durch Recherche stellte ich fest, dass sich das ganze Backend in meiner «app.js»-Datei befindet. Für das nächste Mal werde ich versuchen, mehrere Dateien zu verwenden, um eine bessere Übersicht zu schaffen. Zum Beispiel kann ich den Programmcode für das Aufstarten des lokalen Servers separat in einer «server.js»-Datei programmieren und die Schemas der MongoDB Datenbank in einer anderen Datei namens «schemas.js» definieren.

7.3 Reflexion

HTML, CSS und JavaScript lernte ich zum ersten Mal an der TEKO Olten beim Dozenten Simon Strebel. Das erlernte Grundwissen konnte ich in der Diplomarbeit anwenden und mein Wissen vertiefen. Die Aufgaben der Diplomarbeit waren sehr herausfordern und teilweise einfach bis mittelschwer zu verstehen. Dank dieser Diplomarbeit werde ich es in Zukunft in meinem Berufsumfeld leichter haben, verschiedene HTML, CSS und JavaScript Aufgaben zu implementieren.

Bei Node JS und dessen Module hatte ich keine Vorkenntnisse sowie bei der Datenbank MongoDB. Ich erhielt einen guten Einblick ins Node JS Backend Framework mit dem Hauptmodul Express und die verschiedenen Module, die meine Website zum Laufen brachten. Ich interessiere mich sehr für die «Cybersecurity» (Internet-Sicherheit), daher gefall mir die Implementierung und Anwendung von Bcryptjs auf meiner Website zum «hashing» (Verschlüsselung) der Passwörter sehr.

In Zukunft werde ich in meiner Freizeit und hoffentlich auch im Geschäftlichen an weiteren interessanten Projekten arbeiten. Meine zukünftigen Projekte in der Freizeit bestehen aus Sicherheitsanwendungen, 2D-Spiele sowie 3D-Spiele zu entwickeln und meine Fähigkeiten zu erweitern. Selbstverständlich möchte ich andere Programmiersprachen erlernen und einige habe ich schon im Visier: Python, Java, C/C++ und C# sowie die dazugehörenden Frameworks erlernen und anwenden, wie zum Beispiel für JavaScript: React und Angular JS.

Während der Diplomarbeit musste ich vieles recherchieren, umsetzen und testen. Im Ganzen bin ich sehr zufrieden mit dem Endresultat. Leider wurden nicht alle Ziele erreicht. Aus zeitlichen und Erfahrungsgründe konnte ich die Umsetzung der Programmierung für das Speichern der Highscores auf der Datenbank nicht implementieren. Abgesehen von diesem Ziel wurden alle andere Ziele, die im Pflichtenheft (Muss-Kriterien) und in der Zielscheibe definiert wurden, erreicht.

7.3.1 Problemstellungen

Während der Diplomarbeit traten einige Probleme auf. Die aufgetretene Probleme und Lösungswege werden hier aufgelistet und erklärt. Ich merkte, dass auch kleinere Probleme viel Zeit aufwenden können wie grössere Probleme.

HTML

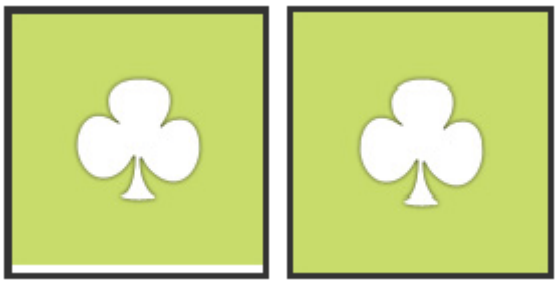
Problem	Stylesheets sowie Skripte zu verlinken, funktionierte nicht.
Lösungsweg	Da ich in der «app.js»-Datei den «public» Ordner deklariert habe, musste ich den Dateipfad beim Verlinken der Stylesheets und Skripte in den HTML-Dokumenten nicht mit /public/ definieren.

Tabelle 3 Problemstellungen HTML

CSS

Problem	Die Fusszeile erscheint in der Mitte und nicht im unteren Bereich der Website.
Lösungsweg	Eigenes Stylesheet für die Fusszeile erstellt (footer.css) mit verschiedenen Styling-Attribute. Mit den Attributen «Display», «Position» und «Margin» die Fusszeile im unteren Bereich platziert.

Problem	Wiederholende Programmierung in der Stylesheets-Dateien.
Lösungsweg	In den HTML-Dateien wurden gemeinsame Klassen («class») definiert, sodass im Stylesheet die Attribute sich nicht mit verschiedenen Klassennamen wiederholen.

Problem	Titelseite (Frontpage): Die Abschnitte (Sections) werden mit den Hintergrundbilder der Spiele nicht vollständig ausgefüllt.
Lösungsweg	Das Attribut «Display» musste umgeändert werden auf «display: block». Bild links: «display: flex», Bild rechts: «display: block».  *Abbildung 196 Problemstellungen CSS Display*

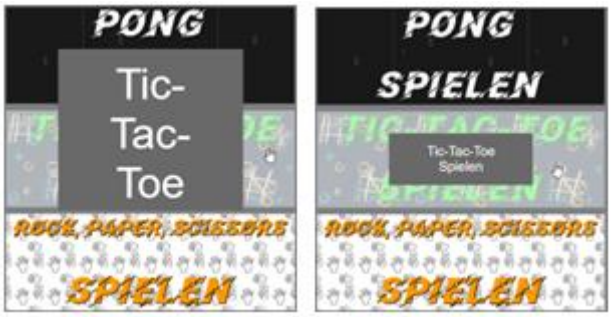
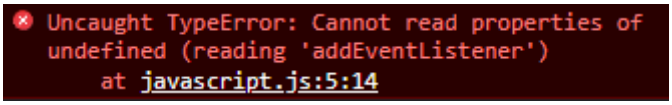
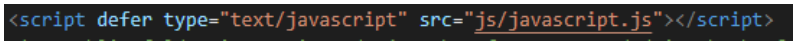
Problem	Titelseite (Frontpage): beim Hover-Effekt der Spielauswahl, Schriftgrösse zu gross.
Lösungsweg	Im Stylesheet die «@media» Regel angewendet, sobald das Browserfenster eine bestimmte Grösse erreicht, wurden die Attribute in dieser Regel geändert. Beim Verkleinern des Browserfensters, wurde die Schriftgrösse kleiner. Bild links: vorher, Bild rechts: nachher  *Abbildung 197 Problemstellungen Spielauswahl*

Tabelle 4 Problemstellungen CSS

JavaScript

Problem	Funktion «toggleButton.addEventListener» in der javascript.js-Datei meldet einen Fehler in der Konsole und ist nicht anwendbar.  *Abbildung 198 Problemstellungen JavaScript Error*
Lösungsweg	Die «toggleButton.addEventListener» Funktion holt Elemente von der Website, die zuerst geladen werden müssen. Um diesen Fehler zu beheben, muss beim Laden des Skriptes das Attribut «defer» verwendet werden. Mit diesem Attribut wird das Skript erst, nachdem die Website geladen wurde, nachgeladen.  *Abbildung 199 Problemstellung Skript "defer"-Attribut*

Problem	Popup für die Anmeldung und das Kontaktformular geht nicht auf.
Lösungsweg	In der javascript.js-Datei war die Funktion «getElementById» falsch geschrieben. Ich habe «getElementsById» geschrieben (Elements anstatt Element).

Tabelle 5 Problemstellungen JavaScript

JavaScript Spiele

Problem	Rock, Paper, Scissors: Punktestand bleibt bei eins fest und ändert sich nicht bei erneutem Gewinnen.
Lösungsweg	Variabel «SCORE» wurde als «SCORE = 0» deklariert, um diesen Wert zu ändern, musste ich, sobald der Spieler einen Punkt bekommt, den «SCORE» mittels «SCORE + 1» aktualisieren.

Problem	Pong: Sobald der Gegner (Computer) einen Punkt bekommt, wird der Ton nicht abgespielt.
Lösungsweg	Durch die Anwendung der Konsole im Browserfenster, merkte ich, dass mein «AdBlocker» diesen Ton blockierte. Der Grund für dies habe ich bis jetzt noch nicht herausgefunden. Ton funktioniert ohne «AdBlocker» einwandfrei.

Problem	Pong: Im Mitgliedsbereich werden die Töne des Pong Spiels nicht abgespielt.
Lösungsweg	Alle Dateien der Spiele wurden in den Mitgliedsbereich kopiert und die Dateipfade dementsprechend umgeändert. Die Spiele funktionieren somit im Mitgliedsbereich und öffentlichen Bereich fehlerfrei.

Tabelle 6 Problemstellungen JavaScript Spiele

Node JS

Problem	Keinen Zugriff von der «app.js» Datei zu dem «public» Ordner.
Lösungsweg	Der Dateipfad musste deklariert werden. Index.html ist die Titelseite (Frontpage), die zuerst geladen und dargestellt wird.

```
app.use(express.static(__dirname + '/public'));

app.get('/', function(req, res) {
    res.sendFile(__dirname + '/index.html')
});
```

Abbildung 200 Problemstellung Node JS Dateipfad

Problem	Nach Ausführung des Log-ins erfolgt die Umleitung in den Mitgliedsbereich nicht.
Lösungsweg	In der «app.js»-Datei musste ich die «Cookies» und «Sessions» (Sitzungen) programmieren und erstellen. Diese anschliessend ausgeben und bei der Log-in Funktion einbauen.

Tabelle 7 Problemstellungen Node JS

MongoDB

Problem	Daten in der MongoDB Datenbank speichern und diese ablesen (Insert, Read).
Lösungsweg	Daten in der MongoDB Datenbank speichern: Die jeweiligen Schemas deklarieren, um die Daten zu speichern (contactSchema, userSchema) und die «Post»-Methode verwenden, mit der «.save()» Funktion am Ende. Daten von der MongoDB Datenbank ablesen: Die beispielsweise Funktion «.findOne({})» anwenden mit dem zu findendem Parameter (Beispiel: username: username).

Problem	Beim Start der «app.js»-Datei wurde eine neue Datenbank erstellt.
Lösungsweg	Der Verbindungs-String stimmte nicht. Falscher String: mongodb+srv://Paulo:1paulo1@spieleentwicklung.x3hjdb0.mongodb.net/?retryWrites=true&w=majority Im String wurde nicht definiert auf welche Datenbank zugegriffen werden soll, dies passiert vor dem «?retryWrites». Richtiger String: mongodb+srv://Paulo:1paulo1@spieleentwicklung.x3hjdb0.mongodb.net/spieleentwicklung?retryWrites=true&w=majority

Tabelle 8 Problemstellungen MongoDB

Dokumentation

Problem	Dokumentation der Programmcodes dauerte länger als erwartet.
Lösungsweg	Da ich während der Programmierung mir Notizen aufschrieb, im Programmcode selbst, sowie ausserhalb des Programmcodes, ermöglichte es mir, die Dokumentation der Programmcodes noch im Zeitplan zu dokumentieren.

Tabelle 9 Problemstellungen Dokumentation

8 Literaturverzeichnis

Erklärung und Hilfe bei der Programmierung des Tic-Tac-Toe Spiels

Code Explained, YouTube, www.youtube.com/c/CodeExplained

Erklärung und Hilfe bei der Programmierung des Pong Spiels

Code Explained, YouTube, www.youtube.com/c/CodeExplained

Erklärung und Hilfe bei der Programmierung des Rock-Paper-Scissors Spiels

Clever Programmer, YouTube, www.youtube.com/c/CleverProgrammer

Erklärung und Hilfe bei der Programmierung des Node JS Backend

Node JS, www.nodejs.org/en/

Express JS, www.expressjs.com

RemyFamily, YouTube, Mongoose, Express, Bcrypt, www.youtube.com/watch?v=-MriYSYBmg0

MongoDB, www.mongodb.com

Marina Kim, YouTube, Mongoose, Express, www.youtube.com/channel/UCV-AVH8pbyU9rjt4EUr3fgg

Sämtliche Abbildungen und Hilfequellen

W3Schools www.w3schools.com

Stack Overflow www.stackoverflow.com

Coding Forums www.thecodingforums.com

Wikipedia https://de.wikipedia.org

9 Abbildungsverzeichnis

10 Tabellenverzeichnis

11 Anhang

11.1 Steckbrief Paulo Ribeiro

Person

Vorname, Name	Paulo Ribeiro
Geburtsdatum	18.05.1998
Zivilstand	Ledig
Wohnort	Rothrist

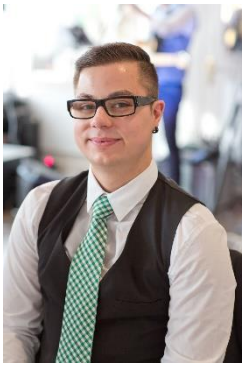


Abbildung 201 Paulo

Beruf

Erlernter Beruf	Produktionsmechaniker EFZ

TEKO

Weiterbildung	Dipl. Techniker HF Informatik Applikationsentwicklung

Aktuelle Arbeitstätigkeit

Univativ Schweiz AG	Temporärer Mitarbeiter als Junior Service Techniker beim Universitätsspital Basel seit Januar 2022.

11.2 Qualifikationsprofil

Paulo Ribeiro
Dipl. Techniker HF
Sennhofweg 7
4852 Rothrist

Qualifikationsprofil

Dipl. Techniker HF
Informatik,
Applikationsentwicklung

Menschen führen *Prozess 1*	Nach meinem Lehrabschluss als Produktionsmechaniker, ein Lehrling in der Mechanik auf die Individuelle praktische Arbeit (IPA) vorbereitet und betreut.
Entscheidungen fällen *Prozess 2*	Bei verschiedene, technische Supportfälle wurden sämtliche Entscheidungen getroffen, um diese zu lösen.
Projekte planen und leiten *Prozess 3*	Im Verein der Guggenmusik Sträggelebrätscher Strengelbach als Organisator tätig für ein Jahr. In dieser Zeit wurde von mir die komplette Tournee geplant und geleitet.
Sich sprachlich verständigen *Prozess 4*	Im Supportbereich schriftlich und sprachlich auf Deutsch, Englisch und Portugiesisch mit Kunden verständigt.
Wirkungsvoll präsentieren und kommunizieren *Prozess 5*	Kundenschulung vor Ort und remote an neuer Software. Notwendige Kenntnisse zu Aufbau, Funktion und Betrieb an den Kunden vermittelt.
Umfeld berücksichtigen *Prozess 8*	Für die Revidierung von verschiedenen HP-Druckermodelle in den jeweiligen Operationssälen, erstellte ich eine Revidierung-Planung, sodass sich niemanden in diese Räume befand und wir die Revidierungen durchführen konnten.
Probleme analysieren und lösen *Prozess 9*	Technische Probleme an verschiedene HP-Druckermodelle analysiert und mit dem notwendigen technischen Hintergrundwissen gelöst, sowie serverseitige Störungen an den Druckerserver behoben.

Datenschutz und Datensicherheit gewährleisten *Prozess 14*	Hashfunktionen bei programmierten Anwendungen verwendet, um die Passwörter der Kunden in der Datenbank sicher zu speichern.
Applikationen entwickeln, Programme erstellen und testen *Prozess 16*	Drei verschiedene Spiele (Rock, Paper, Scissors, Tic-Tac-Toe, Pong) mittels der Programmiersprachen: HTML, Vanilla JavaScript und CSS für die Diplomarbeit programmiert und getestet.
Spezifische Hardware programmieren *Prozess 20*	Selbstgemachte Arduino Alarmanlage mit verschiedenen Komponenten (Sensoren, Bildschirm, Tastatur etc.) zusammengebaut, programmiert und getestet.

11.3 Verdankungen

Für die Unterstützung, Betreuung und die Hilfsbereitschaft bedanke ich mich beim Dozenten Benjamin Bäni. Die Zusammenarbeit mit Benjamin Bäni wurde auf professionelle Ebene und in einem kollegialen Umfeld gehalten.

Während der Programmierung von HTML, CSS und JavaScript bekam ich zusätzlich während des Unterrichts Webengineering, geleitet vom Dozenten Simon Strebel, Unterstützung und Antworten auf aufgetretenen Fragen. Ich bedanke mich bei Simon Strebel für die Hilfsbereitschaft.

11.4 Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Diplomarbeit selbständig und nur unter Verwendung der angegebenen Literaturen und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Diplomarbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Rothrist, den 23. Oktober 2022

Paulo Ribeiro

11.5 Projektablaufplanung ausgefüllt

Projektablaufplanung

Projekt	Spieleentwicklung auf Website
Projektnummer	DA_P1
Projektleiter/in	Paulo Ribeiro
Datum	12.09.2022
Projektinformationen	Programmierung von Website (HTML, CSS, JS), drei Spiele (Vanilla JavaScript) & Backend (Node.js und Modul "Express")
Stand:	*23.10.2022*

Meilenstein

Pos.	Aufgabe/Tätigkeit	Beginn [Datum]	Ende nach [Tage]	Ende [Datum]	Status [0 bis 100]	Meilenstein [Datum]	Verantwortlich [Name]	Unterstützung [Name]
1	Projektziele definieren (Zielscheibe)	12.09.2022	2	13.09.2022	100%		Paulo Ribeiro	Benjamin Bäni
2	Projektstrukturplanung erstellen	12.09.2022	2	13.09.2022	100%		Paulo Ribeiro	Benjamin Bäni
3	Projektablaufplanung erstellen	12.09.2022	2	13.09.2022	100%		Paulo Ribeiro	Benjamin Bäni
4	Software installieren (VSCode, Node.js, etc.)	12.09.2022	2	13.09.2022	100%		Paulo Ribeiro	
5	Node.js und Modul: Express recherchieren	14.09.2022	4	17.09.2022	100%		Paulo Ribeiro	
6	Statusbericht 1	18.09.2022	1	18.09.2022	100%	18.09.2022	Paulo Ribeiro	Benjamin Bäni
7	Lokalhost Programmierung mit Node.js und Modul Express	18.09.2022	2	19.09.2022	100%	19.09.2022	Paulo Ribeiro	
8	Frontpage der Website programmieren	19.09.2022	2	20.09.2022	100%		Paulo Ribeiro	
9	Frontpage Styling	19.09.2022	2	20.09.2022	100%		Paulo Ribeiro	
10	Spiele: Pong, Tic-Tac-Toe und Rock, Paper, Scissors Webseiten programmieren und styling	21.09.2022	1	21.09.2022	100%		Paulo Ribeiro	
11	1. Vorzeigetermin	22.09.2022	1	22.09.2022	100%	22.09.2022	Paulo Ribeiro	Benjamin Bäni
12	Spiel Pong programmieren	22.09.2022	3	24.09.2022	100%	24.09.2022	Paulo Ribeiro	
13	Statusbericht 2	25.09.2022	1	25.09.2022	100%	25.09.2022	Paulo Ribeiro	Benjamin Bäni
14	Spiel Tic-Tac-Toe programmieren	25.09.2022	3	27.09.2022	100%	27.09.2022	Paulo Ribeiro	
15	Spiel Rock, Paper, Scissors programmieren	28.09.2022	3	30.09.2022	100%	30.09.2022	Paulo Ribeiro	
16	Dokumentation Titelblatt, Inhaltsverzeichnis	01.10.2022	1	01.10.2022	100%		Paulo Ribeiro	Benjamin Bäni
17	Impressum, Datenschutz Webpages programmieren	01.10.2022	1	01.10.2022	100%		Paulo Ribeiro	
18	Kontaktformular programmieren	01.10.2022	1	01.10.2022	100%		Paulo Ribeiro	
19	Statusbericht 3	02.10.2022	1	02.10.2022	100%	02.10.2022	Paulo Ribeiro	Benjamin Bäni
20	Dokumentation der Spiele	02.10.2022	2	03.10.2022	100%		Paulo Ribeiro	
21	MongoDB recherchieren	04.10.2022	2	05.10.2022	100%		Paulo Ribeiro	
22	2. Vorzeigetermin	06.10.2022	1	06.10.2022	100%	06.10.2022	Paulo Ribeiro	Benjamin Bäni
23	Programmierung, Verbindung der MongoDB Datenbank	06.10.2022	3	08.10.2022	100%	08.10.2022	Paulo Ribeiro	
24	Statusbericht 4	09.10.2022	1	09.10.2022	100%	09.10.2022	Paulo Ribeiro	Benjamin Bäni
25	Anmeldung, Login programmieren und mit MongoDB Datenbank verbinden	09.10.2022	2	10.10.2022	100%	10.10.2022	Paulo Ribeiro	Benjamin Bäni
26	Dokumentation der Initialisierung, Planung und Realisierung	10.10.2022	4	13.10.2022	100%		Paulo Ribeiro	
27	Programmcode: Kontrolle, Anpassungen Korrekturen	14.10.2022	2	15.10.2022	100%		Paulo Ribeiro	
28	Statusbericht 5	16.10.2022	1	16.10.2022	100%	16.10.2022	Paulo Ribeiro	Benjamin Bäni
29	Dokumentation: Lessons Learned, Reflexion, Problemstellungen, Management Summary	16.10.2022	3	18.10.2022	100%		Paulo Ribeiro	
30	Dokumentation: Kontrolle, Anpassungen Korrekturen, Fertigstellung	19.10.2022	3	21.10.2022	100%	21.10.2022	Paulo Ribeiro	
31	Qualifikationsprofil erstellen	22.10.2022	1	22.10.2022	100%	22.10.2022		
32	Statusbericht 6	23.10.2022	1	23.10.2022	100%	23.10.2022	Paulo Ribeiro	Benjamin Bäni
33	Produktübergabe	24.10.2022	1	24.10.2022	100%	23.10.2022	Paulo Ribeiro	Benjamin Bäni

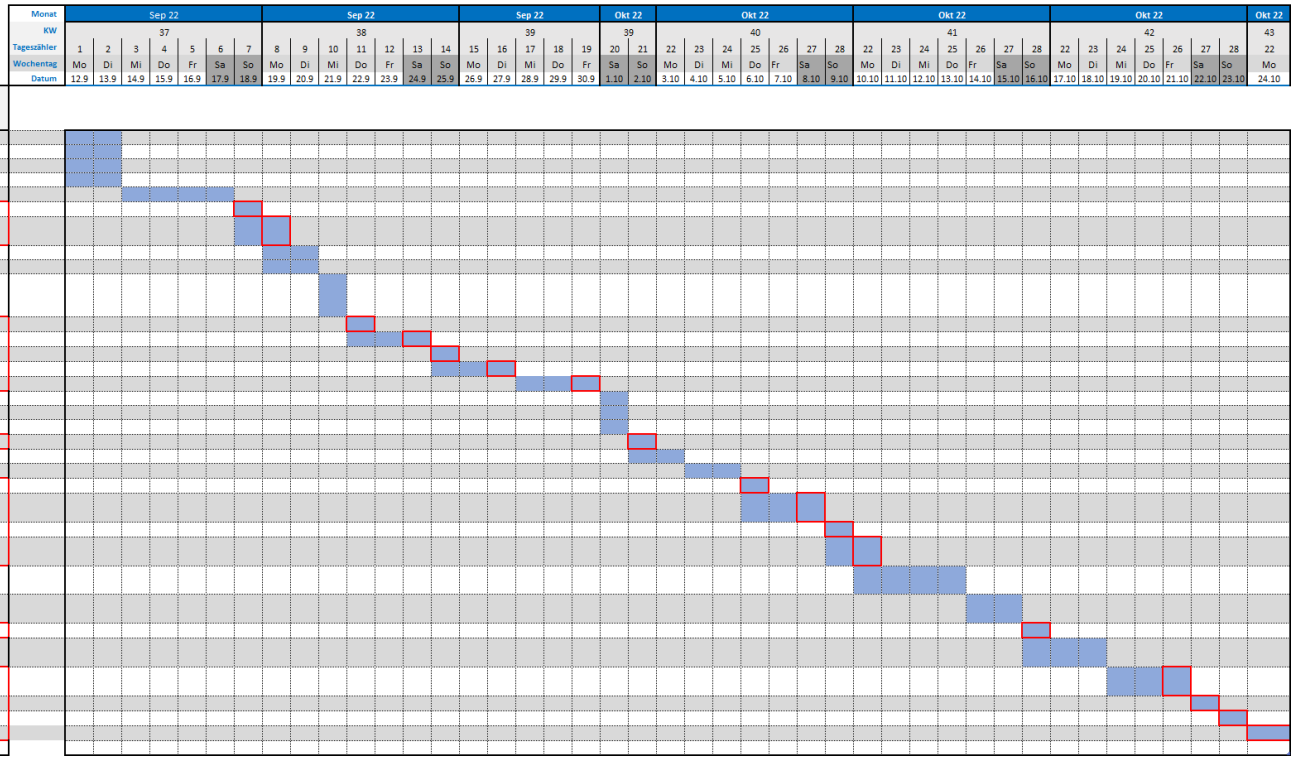


Abbildung 202 Projektablaufplanung ausgefüllt

11.6 Statusberichte

Projekt: Website Spieleentwicklung **Statusbericht: 1 / KW 37**

Projektleiter	Projektziele	Verteiler
Paulo Ribeiro	• Drei Vanilla JavaScript Spiele programmieren • HTML Website programmieren und mit CSS Stylen • Node.js mit Modul «Express» als Backend	• Benjamin Bäni

Gesamt-beurteilung	Projektverlauf	Projektklima	Termine	Risiken	Ressourcen
Tendenz	⇨	⇨	⇨	⇨	⇨

Aktueller Projektstand

- Zielscheibe wurde erstellt
- Zeitplan wurde erstellt
- Beginn der Diplomarbeit: Ressourcen recherchieren
- Mit der Programmierung der Website und Spiele angefangen

Was läuft gut?

- Die Programmierung läuft gut. Hilfequellen wie Google, YouTube etc. verwendet.

Was läuft nicht gut?

- -

Geplante nächste Schritte / getroffene Massnahmen

- Website und Spiele weiter programmieren
- Dokumentation teilweise beginnen

Projekt-Statusbericht; Stefan Thöni, Josef Räber

Abbildung 203 Statusbericht 1

Projekt: Website Spieleentwicklung

Statusbericht: 2 / KW 38

Projektleiter	Projektziele	Verteiler
Paulo Ribeiro	• Drei Vanilla JavaScript Spiele programmieren • HTML Website programmieren und mit CSS Stylen • Node.js mit Modul «Express» als Backend	• Benjamin Bäni

Gesamt-beurteilung	Projektverlauf	Projektklima	Termine	Risiken	Ressourcen
	■ □ □	■ □ □	■ □ □	■ □ □	■ □ □
Tendenz	⇗	⇨	⇗	⇨	⇨

Aktueller Projektstand

- Die drei Vanilla JavaScript Spiele: Tic-Tac-Toe, Pong und Rock, Paper, Scissors wurden programmiert und funktionieren fehlerfrei.
- Es wurde eine Verbindung zu der Datenbank MongoDB hergestellt mittels Node.js und das Modul «Mongoose».
- Dokumentation: Titelblatt und Inhaltsverzeichnis wurde erstellt.

Was läuft gut?

- Momentan im Ablaufplan gut unterwegs. Verschiedene Ziele/Meilensteine wurden schnell erreicht.

Was läuft nicht gut?

- -

Geplante nächste Schritte / getroffene Massnahmen

- An der Website weiter programmieren. Es fehlen an Informationen und Styling.
- MongoDB: Konto eröffnen, Daten in der MongoDB speichern und diese für den Login abrufen.
- Dokumentation der Spiele (Programmcode, Umsetzung, etc.)

Projekt-Statusbericht; Stefan Thöni, Josef Räber

Abbildung 204 Statusbericht 2

Projekt: Website Spieleentwicklung

Statusbericht: 3 / KW 39

Projektleiter	Projektziele	Verteiler
Paulo Ribeiro	• Drei Vanilla JavaScript Spiele programmieren • HTML Website programmieren und mit CSS Stylen • Node.js mit Modul «Express» als Backend	• Benjamin Bäni

Gesamt-beurteilung	Projektverlauf	Projektklima	Termine	Risiken	Ressourcen
	■ □ □	■ □ □	■ □ □	■ □ □	■ □ □
Tendenz	⇨	⇨	⇨	⇨	⇨

Aktueller Projektstand

- MongoDB Daten werden gespeichert und man kann diese abrufen.
- Impressum, Datenschutz Webpages wurden programmiert.

Was läuft gut?

- Anmeldungsdaten werden in der MongoDB gespeichert.
- Die Daten auf der MongoDB können nun von Node.js abgerufen und verwendet werden.

Was läuft nicht gut?

- Schwierigkeiten mit der Programmierung des Log-ins (Sessions, etc.)

Geplante nächste Schritte / getroffene Massnahmen

- An der Log-in-Funktion weiter arbeiten und diese umsetzen.
- Kontaktformular programmieren, sodass die Informationen in der Datenbank (MongoDB) gespeichert werden.
- Dokumentation der Spiele (Programmcode, Umsetzung etc.) fertigstellen.

Projekt-Statusbericht; Stefan Thöni, Josef Räber

Abbildung 205 Statusbericht 3

Projekt: Website Spieleentwicklung

Statusbericht: 4 / KW 40

Projektleiter	Projektziele	Verteiler
Paulo Ribeiro	• Drei Vanilla JavaScript Spiele programmieren • HTML Website programmieren und mit CSS Stylen • Node.js mit Modul «Express» als Backend	• Benjamin Bäni

Gesamt-beurteilung	Projektverlauf	Projektklima	Termine	Risiken	Ressourcen
Tendenz	⇨	⇨	⇨	⇨	⇨

Aktueller Projektstand

- Kontaktformular wurde programmiert und die Informationen werden in der MongoDB Datenbank gespeichert.
- Die Dokumentation der Spiele (Programmcode, Umsetzung etc.) wurde fertig gestellt.
- Einige Funktionen in der app.js-Datei wurden verändert (Anmeldung, Log-In, Kontaktformular etc.)

Was läuft gut?

- Trotz mehrere Probleme immer noch im Zeitplan.

Was läuft nicht gut?

- Die Log-In Funktion funktioniert noch nicht wie gewollt (Weiterleitung in einem privaten Ordner funktioniert nicht).

Geplante nächste Schritte / getroffene Massnahmen

- Log-In Funktion umsetzen mit dem privaten Ordner (falls das nicht gelingt: Eine HTML-Datei erstellen und auf diese verweisen («Coming Soon» / «Demnächst»)
- Dokumentation der Initialisierung, Planung und Realisierung.
- Programmcode: Kontrolle, Anpassungen, Korrekturen.

Projekt-Statusbericht; Stefan Thöni, Josef Räber

Abbildung 206 Statusbericht 4

Projekt: Website Spieleentwicklung

Statusbericht: 5 / KW 41

Projektleiter	Projektziele	Verteiler
Paulo Ribeiro	• Drei Vanilla JavaScript Spiele programmieren • HTML Website programmieren und mit CSS Stylen • Node.js mit Modul «Express» als Backend	• Benjamin Bäni

Gesamt-beurteilung	Projektverlauf	Projektklima	Termine	Risiken	Ressourcen
Tendenz	⇨	⇨	⇨	⇨	⇨

Aktueller Projektstand

- Log-in Funktion wurde umgesetzt. Umleitung in privaten Ordner erfolgreich nach Log-in.
- Dokumentation der Initialisierung, Planung und Realisierung wurde erledigt.
- Programmcode: Kontrolle, Anpassungen, Korrekturen erledigt.
- Qualifikationsprofil wurde erstellt.

Was läuft gut?

- Log-in Funktion wurde umgesetzt, sodass der Benutzer in einen Mitgliedsbereich umgeleitet wird nach erfolgreichem Log-in.

Was läuft nicht gut?

- -

Geplante nächste Schritte / getroffene Massnahmen

- Dokumentation: Lessons Learned, Reflexion, Projektbericht (Stand des Projekts), Problemstellungen
- Korrektur, Anpassung und Fertigstellung der Dokumentation
- Produktübergabe

Projekt-Statusbericht; Stefan Thöni, Josef Räber

Abbildung 207 Statusbericht 5

Projekt: Website Spieleentwicklung **Statusbericht: 6 / KW 42**

Projektleiter	Projektziele	Verteiler
Paulo Ribeiro	• Drei Vanilla JavaScript Spiele programmieren • HTML Website programmieren und mit CSS Stylen • Node.js mit Modul «Express» als Backend	• Benjamin Bäni

Gesamt-beurteilung	Projektverlauf	Projektklima	Termine	Risiken	Ressourcen
	■ □ □	■ □ □	■ □ □	■ □ □	■ □ □
Tendenz	⇨	⇨	⇨	⇨	⇨

Aktueller Projektstand

- Die Dokumentation ist vollständig.
- Der Programmcode ist vollständig.
- Produktübergabe am 23. Oktober 2022

Was läuft gut?

- Abgabe der Diplomarbeit.

Was läuft nicht gut?

- -

Geplante nächste Schritte / getroffene Massnahmen

- Onlinepublikation bis zur Diplomarbeitspräsentation
- Diplomarbeitspräsentation am 12. November 2022

Projekt-Statusbericht; Stefan Thöni, Josef Räber

Abbildung 208 Statusbericht 6

11.7 Website Layout – Prototyp

Abbildung 209 Prototyp Layout Frontpage

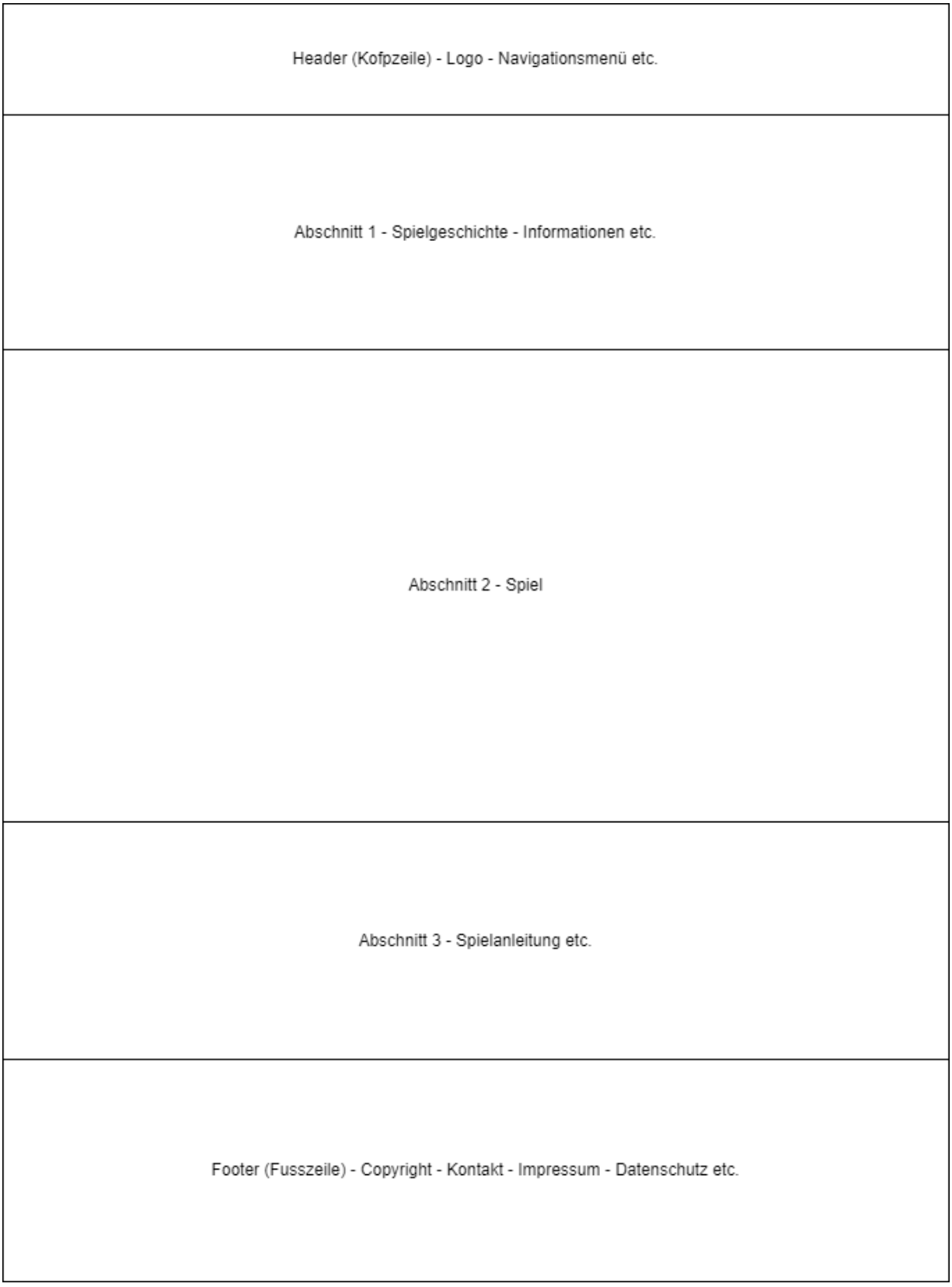


Abbildung 210 Prototyp Layout Spielseite

Abbildung 211 Prototyp Layout Formulare