



M431 - IPERKA-Methode

Dokumentinformationen:

Thema: *Website mit Minispielen*

Projekt-Anfang: *12.05.2025*

Abgabedatum: *04.07.2025*

Lehrperson: *Frau Berrin Ileri*

Autoreninformationen:

Autor: *Anders Kalt, Len Grunder, Dominik Willisch*

E-Mail: *dominik.willisch@edu.tbz.ch*

Klasse: *Pe24a*

Inhaltsverzeichnis

1	Projektauftrag.....	3
2	Informieren	4
2.1	Entwicklungsumgebung	4
2.2	Spiele	5
2.3	Tools.....	5
2.4	Design einer Website.....	5
3	Planen	6
3.1	Tool.....	6
3.2	Tasks	6
3.3	Gantt Diagramm.....	7
3.4	Ressourcen Aufteilung	7
3.5	Ablauf	7
4	Entscheiden	8
4.1	Mutliplayer vs Singleplayer.....	8
4.2	HTML, CSS & Java Script vs React	9
5	Realisieren	10
5.1	GitHub-Repository und Projektstruktur	10
5.2	Design mit Figma	10
5.3	Umsetzung der Hauptwebsite.....	11
5.4	Entwicklung der Spiele.....	12
5.5	Optimierungen und Tests	12
5.6	Tools und Arbeitsweise	12
6	Kontrollieren.....	13
7	Auswerten.....	14
7.1	Kommunikation und Zusammenarbeit.....	14
7.2	Arbeitsergebnis und Würdigung.....	15
8	Sonstiges	16
8.1	Anhänge.....	16
8.2	Quellen.....	16

1 Projektauftrag

Bezeichnung	<i>Play.TBZ</i>
Auftragsgeber	<i>Ileri Berrin</i>
Projektbeginn und -ende	<i>12.05.2025 – 04.07.2025</i>
Beschreibung	<i>Eine Webseite, auf welcher diverse Minispiele gespielt werden können.</i>
Unternehmensbedarf	<i>Wir haben uns vorgenommen, dass wir eine spassige Webseite für uns und andere Mitschüler erstellen. Die Webseite sollte eine kurze Spiel-Möglichkeit zwischen den Pausen bereitstellen.</i>
Ziele	<i>Erfolgreich eine Webseite gestalten</i>
Ergebnisse	<i>Website mit 3 – 5 Minispielen</i>
Budget	<i>0.-</i>
Projektteam	<i>Andres Kalt / Len Grunder / Dominik Willisch</i>
Beschränkungen	<i>Keine</i>
Ressourcenzuweisung	<i>Andres: Head of Design Len: Head of Coding / GitHub Dominik: Head of Coding / Dokumentation</i>
Terminvorgaben	<i>Abgabe: 04.07.2025</i>

2 Informieren

2.1 Entwicklungsumgebung

Wir haben uns intensiv mit verschiedenen Entwicklungsumgebungen auseinandergesetzt und uns im Internet über die verfügbaren Möglichkeiten informiert. Auch haben wir das Internet nach Frameworks durchsucht, welche für unser Projekt nützlich sein könnten.

2.1.1 Entwicklungsumgebung




Wir haben uns intensiv mit verschiedenen Entwicklungsumgebungen beschäftigt und im Internet recherchiert, welche Werkzeuge für die Erstellung einer Website am besten geeignet sind. Visual Studio Code kennen wir bereits aus dem Unterricht, weshalb wir diese Umgebung sicher in Betracht ziehen. Diese Umgebung bietet viele hilfreiche Erweiterungen für Webentwicklung, wie z. B. Live-Server und Git-Integration. Auch IntelliCode bietet ähnliche Features. Danach gibt es noch weitere IDE's wie „Rider DIE“, welche aber kostenpflichtig sind.

2.1.2 Hosting Dienst

Als Hosting-Dienste sind uns GitHub, GitLab und Bitbucket begegnet. Mit GitHub und GitLab haben wir bereits gearbeitet und sind damit vertraut. Bitbucket hingegen ist für uns neu, damit hatten wir bisher noch keine praktischen Erfahrungen.

2.1.3 Programmiersprachen

Für die Gestaltung einer Website sind verschiedene Programmiersprachen erforderlich:

HTML		HTML ist die Struktur einer Webseite – wie das Grundgerüst. Es legt fest, welche Inhalte (Text, Bilder, Links etc.) angezeigt werden.
CSS		CSS ist für das Aussehen und Design zuständig. Es bestimmt Farben, Schriftarten, Layouts und Abstände.
Java Script		JavaScript macht Webseiten interaktiv und dynamisch. Es kann auf Nutzeraktionen reagieren und Inhalte live verändern.

2.2 Spiele

Wir haben uns angeschaut, was ein gutes Spielerlebnis auf einer Website ausmacht. Dabei war uns besonders wichtig, dass die Spiele flüssig laufen, intuitiv zu bedienen sind und den Spielenden langfristig Spass machen. In unserer Recherche haben wir verschiedene Spielarten analysiert von Reaktions- und Denkspielen bis hin zu kleinen Geschicklichkeitsspielen.

Ausserdem haben wir überlegt, welche Arten von Spielen uns selbst am meisten Freude bereiten. Basierend darauf haben wir eine Ideenliste erstellt, die inzwischen deutlich mehr als fünf verschiedene Spielkonzepte umfasst. Die Liste enthält sowohl einfache Spielideen mit wenig technischem Aufwand als auch komplexere Varianten mit mehr Interaktivität.

Da wir uns aktuell noch in der Informations- und Planungsphase befinden, haben wir bewusst noch keine Auswahl getroffen. Ziel war es zunächst, möglichst viele unterschiedliche Ideen zu sammeln, um später aus einem breiten Spektrum die geeignetsten Spiele für unsere Website auszuwählen.

2.3 Tools

Um eine Website zu gestalten, sind viele Vorbereitungsschritte notwendig. Deshalb haben wir im Internet recherchiert, welche Tools es gibt, um den gesamten Gestaltungsprozess zu vereinfachen. Dabei sind wir unter anderem auf folgende Werkzeuge gestossen:



Figma, ein Design Tool für Mockups von z.B. Websites



Monday, ein Planungs Tool von Aufgaben und Terminen



Canva, ein Design Tool von Präsentationen und Dokumenten

2.4 Design einer Website

Wir haben uns auch verschiedene Websites angesehen und dabei besonders den Aufbau und die Benutzerführung analysiert. Dadurch konnten wir uns ein gutes Bild davon machen, wie unsere eigene Website möglichst ansprechend gestaltet werden könnte.

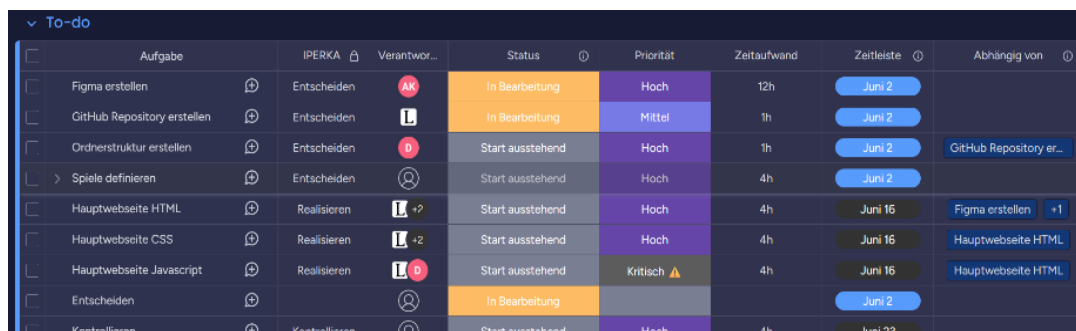
3 Planen

3.1 Tool

Für die Planung nutzen wir das Online-Tool Monday, das uns von unserer Lehrperson empfohlen wurde. Auch Andres hat bereits mehrfach beruflich mit diesem Tool gearbeitet und kennt sich daher gut damit aus. Er hat uns schnell die Grundlagen erklärt, wodurch wir viel Zeit gespart haben und früher mit der eigentlichen Planung starten konnten.

3.2 Tasks

Wir haben eine Liste aller notwendigen Aufgaben (Tasks) erstellt, um unser Produkt in hoher Qualität fertigzustellen. Diese Aufgaben haben wir weiter unterteilt, bis eine detaillierte Übersicht entstand. In der Tabelle sind mehrere Spalten enthalten, die dabei helfen, jeden Task richtig einzuschätzen. Für jeden Task haben wir diese Felder ausgefüllt, sodass am Ende eine vollständige und strukturierte Liste vorliegt.



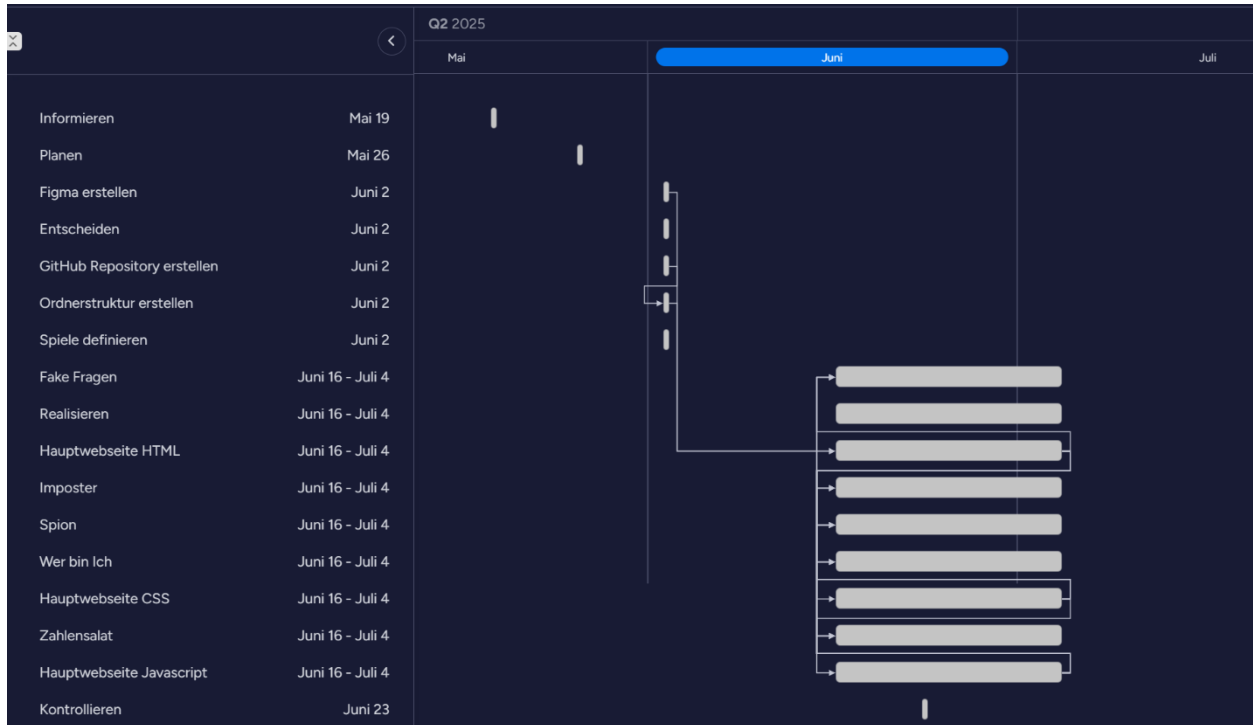
Aufgabe	IPERKA	Verantwort...	Status	Priorität	Zeitaufwand	Zeitleiste	Abhängig von
Figma erstellen	Entscheiden	AK	In Bearbeitung	Hoch	12h	Juni 2	
GitHub Repository erstellen	Entscheiden	L	In Bearbeitung	Mittel	1h	Juni 2	
Ordnerstruktur erstellen	Entscheiden	D	Start ausstehend	Hoch	1h	Juni 2	GitHub Repository er...
> Spiele definieren	Entscheiden		Start ausstehend	Hoch	4h	Juni 2	
Hauptwebseite HTML	Realisieren	L +2	Start ausstehend	Hoch	4h	Juni 16	Figma erstellen +1
Hauptwebseite CSS	Realisieren	L +2	Start ausstehend	Hoch	4h	Juni 16	Hauptwebseite HTML
Hauptwebseite Javascript	Realisieren	L D	Start ausstehend	Kritisch	4h	Juni 16	Hauptwebseite HTML
Entscheiden			In Bearbeitung			Juni 2	
Kontrollieren	Kontrollieren		Start ausstehend	Hoch	4h	Juni 23	

Screenshot aus unserem Monday nach der Ersten Planungsphase.

Für jeden Task haben wir ein Enddatum festgelegt. Dadurch konnte Monday automatisch ein Gantt-Diagramm erstellen, das den zeitlichen Ablauf der Aufgaben übersichtlich darstellt. Wir haben ausserdem die nötigen Abhängigkeiten zwischen den Tasks eingetragen, sodass ersichtlich ist, welche Aufgaben aufeinander aufbauen und in welcher Reihenfolge sie erledigt werden sollen.

Die Darstellung des Gantt-Diagramms in Monday ist jedoch nicht ganz optimal. Deshalb kann es an einigen Stellen etwas unübersichtlich wirken und die Übersicht erschweren. Trotzdem ist das Diagramm eine hilfreiche Unterstützung bei der Planung und Übersicht unseres Projekts.

3.3 Gantt Diagramm



Screenshot aus unserem Monday von unserem Gantt Diagramm.

3.4 Ressourcen Aufteilung

Um unser Projekt sinnvoll umzusetzen, haben wir für jeden Task eine verantwortliche Person festgelegt. Das bedeutet nicht, dass die anderen keine Beiträge zu diesem Task leisten dürfen, aber so konnten wir die Arbeit besser aufteilen und möglichst effizient zusammenarbeiten. Grundsätzlich haben wir folgende Aufgabenverteilung vereinbart:

- **Andres:** Design und Präsentation
- **Len:** Code und GitHub
- **Dominik:** Dokumentation und Code

3.5 Ablauf

Im Gantt-Diagramm ist der zeitliche Ablauf unserer Tasks gut ersichtlich. Wir haben als Erstes das GitHub-Repository erstellt und eine sinnvolle Ordnerstruktur aufgebaut. Damit war die Grundlage für eine organisierte Zusammenarbeit gelegt. Anschliessend starteten wir mit den ersten Aufgaben in den Bereichen Code, Design und Dokumentation.

4 Entscheiden

4.1 Mutliplayer vs Singleplayer

Multiplayer vs Singleplayer							
Kriterien		Multiplayer			Singleplayer		
Kriterium	Gewichtung	Eigenschaft	Punkte	Ergebnis	Eigenschaft	Punkte	Ergebnis
Umsetzbarkeit	40%	Schwierig	4	1.6	Flexibel	8	3.2
Nutzbarkeit	25%	Cool aber Problematisch	8	2	Einfach	6	1.5
Lernfaktor	15%	Hoch	7	1.05	Mittelmässig	5	0.75
Benötigte Ressourcen	5%	Viele / Teilweise Teuer	3	0.15	Wenig/Gratis	9	0.45
Hintergrundleistung	5%	Starke Leistung für Server benötigt	2	0.1	Fast keine	6	0.3
Herausforderung	10%	Hoch	7	0.7	Mittelmässig	5	0.5
Total	100.00%	Resultat: 5.6 Im Vergleich: 45.53%			Resultat: 6.7 Im Vergleich: 54.47%		

4.1.1 Kriterien Auswahl

Wir haben uns ausfolgendem Grund für die oben aufgelisteten Kriterien entschieden. Da diese Entscheidung sehr fest an der Umsetzbarkeit abhängt, ist dieses Kriterium auch am stärksten gewichtet. Ein weiteres Kriterium ist die Nutzbarkeit, welche aus unserer Sicht wichtig ist, um ein gutes finales Produkt zu entwickeln. Die weiteren Kriterien sind eher weniger gewichtet, da diese eher im Hintergrund sind welche das Produkt nicht stark beeinträchtigen. Sie haben demnach eine niedrigere Gewichtung.

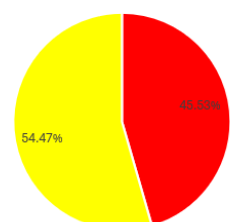
4.1.2 Entscheidungen

Wir haben die Punkte anhand des Schwierigkeitsgrades aufgeteilt und nicht wie gerne wir dies hätten zum Beispiel bei «Nutzbarkeit» von Multiplayer. Bei der «Umsetzbarkeit» hat klar der Singleplayer die Nase vorn. Auch die weniger gewichteten Kriterien sprechen eher für den Singleplayer. Einzig der «Lernfaktor» und die «Nutzbarkeit» sprechen für den Multiplayer.

4.1.3 Ergebnis

Es kam zum Ergebnis, dass wir Singleplayer nehmen, da dies einfacher ist, um aufzubauen und nicht noch mehr Ressourcen benötigt wird, für das Hosten einer Lobby. Das Resultat war eher knapp, jedoch ein guter Anhaltspunkt, von dem wir nun ausgehen können.

Singleplayer vs Multiplayer



4.2 HTML, CSS & Java Script vs React

HTML, CSS & Java Script vs React							
Kriterien		HTML, CSS & Java Script			React		
Kriterium	Gewichtung	Eigenschaft	Punkte	Ergebnis	Eigenschaft	Punkte	Ergebnis
Lernaufwand	25%	gering	5	1.25	Hoch	2	0.5
Umsetzung	25%	Simpel	4	1	Simpel	3	0.75
Unsere Erfahrung	20%	grosse Erfahrung	5	1	keine	0	0
Flexibilität	15%	Flexibel	3	0.45	Hoch	5	0.75
Problemebehebungsmöglichkeiten	15%	viele	5	0.75	viele	4	0.6

Total	100.00%	Resultat: 4.45	Resultat: 2.6
		Im Vergleich: 63.12%	Im Vergleich: 36.88%

4.2.1 Kriterien Auswahl

Die oben genannten Kriterien haben wir gewählt, um eine gute Entscheidung für die Programmiersprachen Umgebung zu wählen. Die beiden Kriterien «*Lernaufwand*» und «*unsere Erfahrung*» sind relativ hoch gewichtet, da wir in dieser kurzen Realisierungszeit nicht viel neues Lernen können. Die «*Flexibilität*» ist auch eher wichtig, da wir viel Freiheiten benötigen beim Umsetzen dieses Projektes.

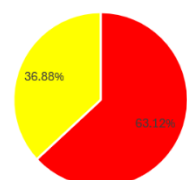
4.2.2 Entscheidungen

Mit HTML, CSS & JavaScript haben wir mehr «*Erfahrung*» als mit React, deshalb haben wir dort mehr Punkte vergeben haben. React ist «*flexibler*», da es ein vorgefertigtes Framework ist, welches spezifische Möglichkeiten bietet. Der «*Lernaufwand*» ist bei React viel höher, deshalb haben wir dort weniger Punkte vergeben. Der Lernaufwand und daraus der Lernfaktor ist eigentlich etwas Positives, da für dieses Projekt aber eher das Ergebnis das ist was zählt, ist die Entscheidung bei diesem Kriterium zu HTML, CSS & Java Script gefallen.

4.2.3 Ergebnis

Das finale Ergebnis ist klar, die klassische Variante mit HTML, CSS und JavaScript hat mit 2/3 gewonnen. Besonders der niedrigere Lernaufwand und unsere Erfahrung waren ausschlaggebend. Mit diesem Ergebnis können wir nun in die Realisierungsphase gehen.

React vs HTML, CSS, Java Script



5 Realisieren

Nach der Planung sind wir nun beim Realisieren, den eigentlich spannendsten Teil des Projekts. Da wir durch GitHub sehr gut zu unterschiedlichen Zeiten arbeiten können, wollten wir das auch unbedingt nutzen. Dieses Ziel haben wir auch ständig verfolgt.

5.1 GitHub-Repository und Projektstruktur

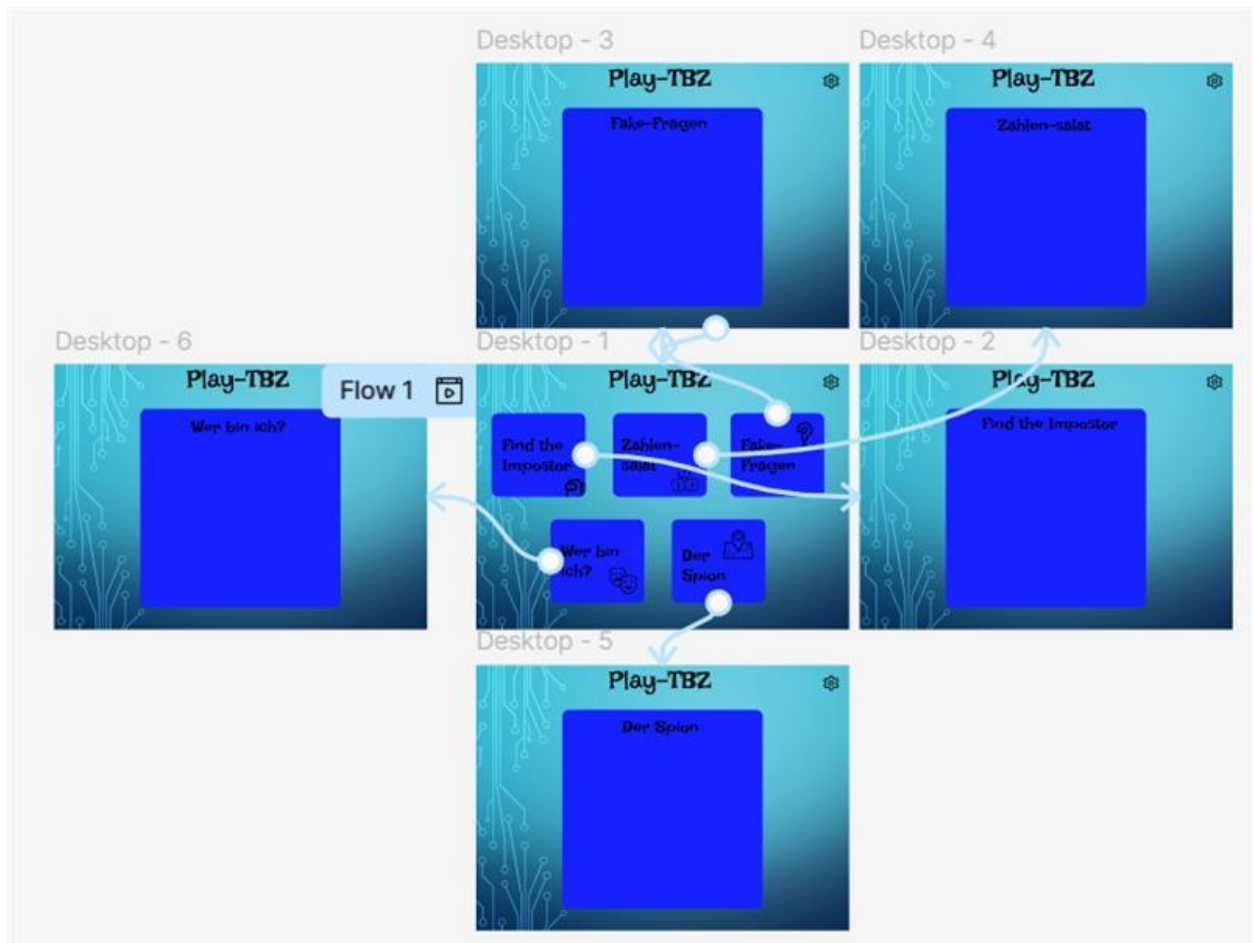
Als Erstes haben wir ein GitHub-Repository erstellt und alle Gruppenmitglieder als Administratoren hinzugefügt. So kann jedes Gruppenmitglied von zu Hause aus am Projekt arbeiten. Dies hat uns sehr geholfen, da wir meist nicht alle gleichzeitig Zeit hatten.

Anschliessend haben wir eine Ordnerstruktur angelegt, welche wir als sinnvoll angesehen haben. Die Ordnerstruktur ist funktional ausgerichtet, das heisst, wir haben für jedes Spiel ein Ordner erstellt. Zusätzlich haben wir die wichtigsten Dateien index.html, style.css und script.js bereits in den ersten Commits angelegt und mit Platzhalter-Inhalten gefüllt. So konnten wir nun mit dem Ausprogrammieren der Spiele beginnen.

5.2 Design mit Figma

Bevor wir mit dem eigentlichen Programmieren beginnen können, haben wir uns mit dem Design der Website beschäftigt. Dafür haben wir das Tool „Figma“ genutzt und verschiedene Entwürfe erstellt mit mehreren Layout-Varianten. Danach haben wir uns für ein finales Design entschieden, welches modern und benutzerfreundlich ist.

Wir haben Farben, Schriftarten, Abstände und Strukturen der Website gewählt. Dieses Figma-Design hat uns während der Umsetzung als visuelle Vorlage gedient, so mussten wir beim Implementieren nicht mehr gross auf das Design achten.



Screenshot aus dem Figma unserer Website.

5.3 Umsetzung der Hauptwebsite

Als das Design endlich fertig war, haben wir mit der Implementierung der Hauptseite begonnen. Wir haben die Startseite mit HTML aufgebaut und mit CSS gestaltet. Dazu gehörten Navigation, Header, Footer, Inhaltsbereiche und responsive Layouts.

Im weiteren Verlauf haben wir JavaScript verwendet, um erste interaktive Elemente umzusetzen, z. B. Buttons, einfache Ladefunktionen und das Einlesen von Inhalten aus unseren JSON-Dateien. Die Navigation zwischen den Seiten und das Einbinden der Spiele haben wir ebenfalls direkt zu Beginn eingerichtet.

5.4 Entwicklung der Spiele

Neben der Hauptwebsite haben wir uns an die Implementierung der ersten drei Spiele gemacht. Wir haben uns bewusst für Spiele entschieden, die leicht verständlich sind, aber trotzdem Spass machen. Danach haben wir Spiel für Spiel vollständig mit HTML, CSS und JavaScript programmiert.

Um unsere Spiele schon vor der effektiven Testphase zu prüfen haben wir während den Pausen in der Schule immer wieder ein Spiel zusammengespielt und direkt das Feedback aufgenommen. Diese Punkte haben wir dann gleich im Code umgesetzt. Dadurch haben wir frühzeitig Fehler erkannt und die Bedienung deutlich optimiert.

Als die ersten drei Mini-Spiele implementiert gewesen sind und auf GitHub dokumentiert, haben wir die zwei weiteren Spiele umgesetzt. Dabei sind wir genau gleich vorgegangen und haben das Spiel, mit einem Menü, kurzer Anleitung und den Steuerungselementen implementiert. Insgesamt haben wir nun fünf funktionierende Spiele in unserem Projekt.



5.5 Optimierungen und Tests

Im Laufe der Entwicklung haben wir viele Anpassungen vorgenommen: Wir haben die Performance verbessert, Animationen eingebaut, Rückmeldungen bei Aktionen eingebaut und auf ein sauberes visuelles Feedback geachtet. Einige Spiele haben wir mehrmals überarbeitet, bis sie stabil und unterhaltsam waren.

Besonders hilfreich war es, dass wir die Spiele im Unterricht regelmässig testen konnten. So hatten wir auch Feedback von anderen Personen. So haben wir ein gutes Gefühl dafür bekommen, was funktioniert und unsere Mitschüler mögen und was noch verbessert werden musste.

5.6 Tools und Arbeitsweise

Während unserer gesamten Umsetzung haben wir die IDE Visual Studio Code verwendet. Durch die Extension „Live Server“ konnten wir unsere Website direkt im lokalen Browser testen. Nach jeder Änderung haben wir einen neuen Commit gemacht und diesen auf GitHub hochgeladen.

6 Kontrollieren

Testfall	Erwartet	Datum	Effektiv	Tester	Status	Zuständig	Ursache
Website funktioniert	Website startet auf Localhost	23.06.2025	Die Website startet wie erwartet	Len	Erfolgreich	Ist Erfolgreich	/
Daten aus JSON	Daten werden aus dem JSON gelesen	30.06.2025	Die Daten werden ins Programm eingelesen	Dominik	Erfolgreich	Ist Erfolgreich	/
Layout entspricht Figma	Website Layout sieht aus wie im Figma	23.06.2025	Die Website sieht aus, wie im Figma	Andres	Erfolgreich	Ist Erfolgreich	/
Responsive Design	Das Website Design ist Responsiv	30.06.2025	Website ist Responsive und passt sich an	Dominik	Erfolgreich	Ist Erfolgreich	/
Durchgezogenen Style	Style ist auf jeder Teilseite gleich	23.06.2025	Es ist ein gemeinsamer Style erkennbar	Andres	Erfolgreich	Ist Erfolgreich	/
Navigation der Website	Die Navigation funktioniert korrekt	30.06.2025	Navigation funktioniert, weiterleitung stimmt	Len	Erfolgreich	Ist Erfolgreich	/
Verständlichkeit / Intuitiv	Die ist Verständlich und man findet sich zurecht	30.06.2025	Kein Tutorial vorhanden bei den Spielen, sonst gut	Lazar	Behoben	Len Dominik	Wir konnten unsere Spiele und brauchten keine Anleitung
Build der Website funktioniert	Der Build der Website läuft Fehlerfrei durch	30.06.2025	Der Build ist vollständig und korrekt durchgelaufen	Dominik	Erfolgreich	Ist Erfolgreich	/

Nach der Umsetzung haben wir unsere Website in mehreren Testschritten überprüft.

Zuerst haben wir getestet, ob sie lokal auf dem eigenen Gerät startet. Wie erwartet hat das hat problemlos funktioniert. Die Seite wurde wie erwartet unter «localhost:8000» geladen.

Anschliessend haben wir kontrolliert, ob die Daten aus den JSON-Dateien korrekt in die Website eingelesen werden. Die Daten wurden an den Stellen im Projekt korrekt angezeigt.

Beim Layout haben wir unsere Website mit dem Figma-Design verglichen. Das Ergebnis war sehr nah am Original, kleinere Abweichungen gab es dennoch da es nicht so einfach umsetzbar war, diese waren aber minimal.

Auch das responsive Verhalten auf verschiedenen Bildschirmgrössen hat funktioniert. Die Seite war auf dem Handy, Tablet und PC gut nutzbar. Diesen Testfall konnten wir mit den Firefox Entwickler-Tools prüfen.

Der Style ist auf jeder Unterseite gleich. Wir haben immer die gleichen Farben und das gleiche Hintergrundbild verwendet. So sah das ganze Projekt in sich stimmig aus.

Die Navigation zwischen den Seiten hat zuverlässig funktioniert, alle Links haben zur richtigen Seite geführt. Man kann auch immer zurück zur Startseite wechseln.

Auch die Benutzerfreundlichkeit wurde getestet. Für das haben wir die Spiele mit Mitschülern gespielt, dabei wurde angemerkt, dass eine Anleitung zu den einzelnen Spielen nicht schlecht wäre. Dies haben wir danach noch implementiert, es war eine sehr gute Idee.

Insgesamt haben alle Testfälle ein gutes Ergebnis gezeigt, und die Website war stabil, funktional und optisch stimmig.

7 Auswerten

7.1 Kommunikation und Zusammenarbeit

7.1.1 Beteiligte Personen

An unserem Projekt waren wir als Team und zu teils unsere Lehrperson beteiligt. Die Lehrperson hat uns bei Fragen unterstützt und wichtige Hinweise gegeben. Ausserdem haben wir uns gegenseitig im Team immer wieder ausgetauscht.

7.1.2 Ablauf und Vorgehen

Den in der Planungsphase gemachten Plan wollten wir so gut es ging verfolgen. Teilweise mussten wir den Plan jedoch anpassen, weil manche Aufgaben mehr Zeit gebraucht haben als gedacht oder technische Schwierigkeiten aufgetreten sind. Als Beispiel hatten wir Probleme mit GitHub Pages, weshalb wir die Seite nicht direkt auf GitHub hosten konnten.

7.1.3 Problemfälle

Ein Problem war, das wir die Spiele selbst kannten und deshalb keine Anleitung oder Ähnliches geschrieben haben. Dies ist dann aber zum Glück beim Testen aufgefallen.

7.1.4 Erkenntnisse und neue Erfahrungen

Wir haben gelernt, wie wichtig gute Planung und regelmässige Kommunikation sind. Ausserdem haben wir erfahren, dass man flexibel sein muss, wenn nicht alles genau nach Plan läuft. Der Umgang mit GitHub und dem Design-Tool Figma wurde mit der Zeit einfacher.

7.1.5 Ressourceneinsatz

Wir haben die Zeit und unsere Fähigkeiten gut genutzt, indem wir die Aufgaben entsprechend unseren Stärken verteilt haben. Da jeder von uns daran arbeiten konnte, wann er gerade Zeit hatte, konnten wir effizient zusammenarbeiten.

7.1.6 Noch offene Probleme

Der Host auf GitHub Pager wäre noch sehr cool gewesen, da man dann den ganzen source Code nicht mehr Lokal auf seinem PC braucht. Dieses Feature hätten wir gerne umgesetzt, war aber aus mehreren Gründen nicht so einfach möglich.

7.2 Arbeitsergebnis und Würdigung

7.2.1 Qualität

Unsere Website funktioniert gut, sieht ansprechend aus und ist einfach zu bedienen. Die Spiele laufen stabil und machen Spaß.

7.2.2 Dank und Anerkennung

Wir danken uns gegenseitig im Team für die gute Zusammenarbeit. Jeder hat sich in unterschiedlichen Bereichen eingebracht: Andres hat sich vor allem um das Design gekümmert, Len um den Code und GitHub, und Dominik hat viel an der Dokumentation gearbeitet.

7.2.3 Erkenntnisse

Wir haben viel Neues gelernt, besonders wie man ein Projekt plant und im Team arbeitet. Die Erfahrungen mit Figma und GitHub wollen wir bei zukünftigen Projekten wieder nutzen. Wir möchten unsere Zeitplanung noch besser machen und uns in andere Sichtweisen versetzen.

8 Sonstiges

8.1 Anhänge

Projekt lokal ausführen

Voraussetzungen

- Python 3.x (für den lokalen Webserver)
- Git

Schritte zum Klonen und Starten

Repository klonen

```
git clone https://github.com/Dominik-TBZ/Modul-431-Projektarbeit.git
```



In den Projektordner wechseln

```
cd Modul-431-Projektarbeit
```



Lokalen Webserver starten (Port 8000)

```
python -m http.server 8000
```



8.2 Quellen

8.2.1 GitHub

<https://github.com/>

8.2.2 Monday

<https://monday.com>

8.2.3 ChatGPT

<https://chatgpt.com>

8.2.4 Stack Overflow

<https://stackoverflow.com/>

8.2.5 Figma

<https://figma.com>

8.2.6 Crazy Games

<https://de.crazygames.com/>