PROJEKTARBEIT MODUL 288

Mariano Pfister und Philipp Heer

Inhaltsverzeichnis

Aufgabenstellung:

Was ist unser Auftrag?

Wir erstellen eine individuelle Webapplikation, benutzerfreundlich, schnell und sicher in HTML, CSS und JavaScript. Aus welchem Bereich wir die Aufgabenstellung wählen ist dabei frei.

Vorgegeben Aufgabenstellung aus dem Arbeitsplatt:

Welche Eingaben verlangt das Programm?

- Mausklick/Fingertippen

Welche Verarbeitung, Berechnung übernimmt das Programm?

- Ein simples Klick-Spiel, welches die Anzahl Klicks auf einen bestimmten Bereich mitzählt und die Anzahl an Tipps wieder ausgibt. Nach dem Erreichen einer gewissen vordefinierten Zahl, fängt das Spiel selbst an mit zu klicken. Das Spiel übernimmt also die Rolle als Zähler, Verarbeiter und Ausgabe der Klicks.

Inspiration: https://orteil.dashnet.org/cookieclicker/

Welche Ausgaben liefert das Programm?

- Neuberechnung an Cookies die man hat

Was sind die Aufgaben?

Aufgabe 1: Aufgabestellung formulieren

- Welche Eingaben verlangt das Programm?
- Erstellen Sie eine Liste mit diesen Daten und definieren Sie auch den Datentyp der einzelnen Eingaben.
- Welche Verarbeitung, Berechnung übernimmt das Programm?
- Welche Ausgaben liefert das Programm?

Bevor weitergearbeitet werden kann, muss die Aufgabestellung durch den

Projektleiter genehmigt werden.

Aufgabe 2: Codieren

- Beim Codieren sind Teilaufgaben in Funktionen auszulagern.
- Der Code ist mit erklärenden Kommentaren zu dokumentieren.

Aufgabe 3: Test durchführen und dokumentieren

- Einige sinnvolle Testfälle mit den erwarteten Resultaten sind durchzuführen und in einem Office-Dokument (Word oder Excel) festzuhalten (Screenshots einfügen).
- Ungelöste Probleme und allfällige Fehler sind festzuhalten.

Mindestanforderungen:

Die Applikation umfasst …

- … HTML- und CSS- und JavaScript-Code mit inline Dokumentation (Kommentar)
- … beim Testen sind keine Fehler erkennbar
- … Benutzereingaben führen zu interaktiver Verarbeitung (Zahlen, Texte, Bilder)
- … ist einfach, intuitiv zu bedienen, da gute Benutzeroberfläche
- … Verarbeitung (Berechnung oder Darstellungsänderung) vorhanden
- … ist korrekt und sauber codiert.
- … wird gut erklärt mit Kommentar (Inline-Dokumentation)
- … Qualität und Arbeitsmenge entspricht den Anforderungen

Unsere Planung:

Erste Überlegungen/Thematik:

Nach dem wir den Arbeitsauftrag zum Projekt gelesen haben, machten wir uns bereits die ersten Gedanken, wie unsere Applikation aussehen bzw. welche Funktionen sie haben könnte. So begann unsere Brainstorming-Phase und wir überlegten uns, was wollen wir machen. Uns war er aber von Anfang an klar, normales ist verboten. Einen einfachen Taschenrechner oder ähnliches, wozu es schon aus dem Unterricht Vorlagen und hunderte Lösungswege im Internet zu finden gibt, kommt für uns nicht in Frage. Wir wollten mit unserem Wissen etwas Aussergewöhnliches erschaffen. Nach langen Überlegungen und das Durchsuchen von Webseiten nach Inspiration, haben wir uns entschlossen, ein Browsergame zu entwickeln, welches wir aus alten Schulzeiten schon gespielt und geliebt haben, als unsere Projektarbeit für dieses Modul zu wählen.

Es geht um das beliebte Browsergame Cookie Clicker von Orteil. Cookie Clicker gilt als eines der Spiele mit dem höchsten Suchtpotential der vergangenen Jahre. Das Spielprinzip ist dabei relativ einfach gehalten, nichtsdestotrotz fesselt es auch heute noch hunderttausende von Spielern vor den Monitor. Ziel des Spiels liegt darin, so viele Cookies (Kekse) wie möglich zu generieren.

Der Spieler klickt anfangs wiederholt auf einen grossen Keks und erhält für jeden Klick ein Cookie. Hat man genug gesammelt, kann man sich für diese z. B. zusätzliche Klicks kaufen oder Upgrades besorgen, die alle paar Sekunden automatisch für den Spieler klicken. So generieren Spieler immer mehr Cookies, um sich bald immer effektivere, automatische Cookie-Produzenten leisten zu können.

Wir machen es aber anders:

Das Zusammenspiel zwischen HTML, CSS und JavaScript ist extrem wichtig. HTML bildet das Grundgerüst der Seite und stellt Inhalte, Überschriften, Listen, Tabellen, usw. dar. CSS ist die Stylesheet Language, um die Webseite zu gestalten und Farbe, Schriftart, Layout und vieles mehr zu definieren. JavaScript ist für die «Dynamik» auf Webseiten verantwortlich. Immer wenn sich etwas verändern soll, kann man mittels JavaScript definieren, was sich wie verändern soll. Dabei verändert man entweder den Inhalt oder die Struktur (also HTML) der Webseite oder bzw. das Layout (also CSS) einzelner HTML-Elemente. Da wir aber mit JavaScript nicht so gerne arbeiten und wir von unseren Betrieben schon etwas mehr Erfahrung in der Webprogrammierung mitbringen, wollen wir mit iQuery arbeiten.

Es gibt in JavaScript Gegebenheiten, die einem das Leben als Programmierer erschweren:

Man muss jeden Schritt (also jede Anweisung) einzeln ausformulieren.

In JavaScript muss man jedem Element einzeln sagen, was man von ihm will, auch wenn ich es einem anderen schon gesagt habe.

Und genau hier setzt jQuery ein. jQuery ist nämlich eigentlich nichts anderes als eine mittlerweile sehr umfangreiche Sammlung von Programmteilen für das Manipulieren von Elementen auf Webseiten.

Was ist jQuery?

jQuery ist eine JavaScript-library (Bibliothek), das umfangreiche Funktionen zur Navigation und Manipulation der Inhalte auf einer Webseite bereitstellt.

Eine Javascript-library ist ein bestehender JavaScript Code das man bei seiner Seite implementieren kann, dies vereinfacht hilft beim Entwickeln mit JavaScript.

Der Vorteil von jQuery liegt klar bei der Anzahl an zu schreibenden Code und es erweitert bisherigen JavaScript-Anwendungen. Diese ist in einer einzigen .js-Datei enthalten. Einfache Möglichkeiten, Elemente auf der Website auszuwählen (um dann später damit Sachen zu machen wie Aussehen ändern, Animationen oder Aktionen). Was bisher in JavaScript umständlich über „getElementById“ und „getElementsBy“ gemacht wurde, geht jetzt schnell und problemlos über $("#bereich1") und $(".farbe1"). Beim Auswählen der Elemente ist man sehr nahe an der Schreibweise wie bei CSS.

jQuery bietet zudem noch Vorteile, wenn es über Google geladen wird. jQuery ist unter Umständen bereits geladen, weil eine andere Website auch über Google jQuery eingebunden hat. Somit ist jQuery bereits im Cache des Besuchers und sofort verfügbar. Es wird schneller von Google geladen. Normalerweise wird es vom Google-Server schneller geladen, da es eine Beschränkung der gleichzeitig zu ladenden Dateien von einem Webserver gibt.

jQuery oder JavaScript:

Die Frage «jQuery oder JavaScript» muss man sich eigentlich gar nicht mehr stellen, weil jQuery sich im Laufe der Erläuterungen als einfacher und effizienter herausstellt. Nun, um JavaScript wird man dennoch kaum herumkommen, denn jQuery ist in JavaScript geschrieben. Wenn man also jQuery ohne JavaScript-Kenntnisse nutzen wolle, wäre das so, wie wenn man Texte ohne Sprache schreiben wolle.

.

Konzeption der Applikation:

Nun brauchen wir einen Plan. Wir mussten unsere Überlegungen irgendwo festhalten können. Eine Skizze von Hand war da für uns nicht das richtige. Wir erstellen uns also ein Screendesign mit Adobe InDesign.

Unsere Überlegungen:

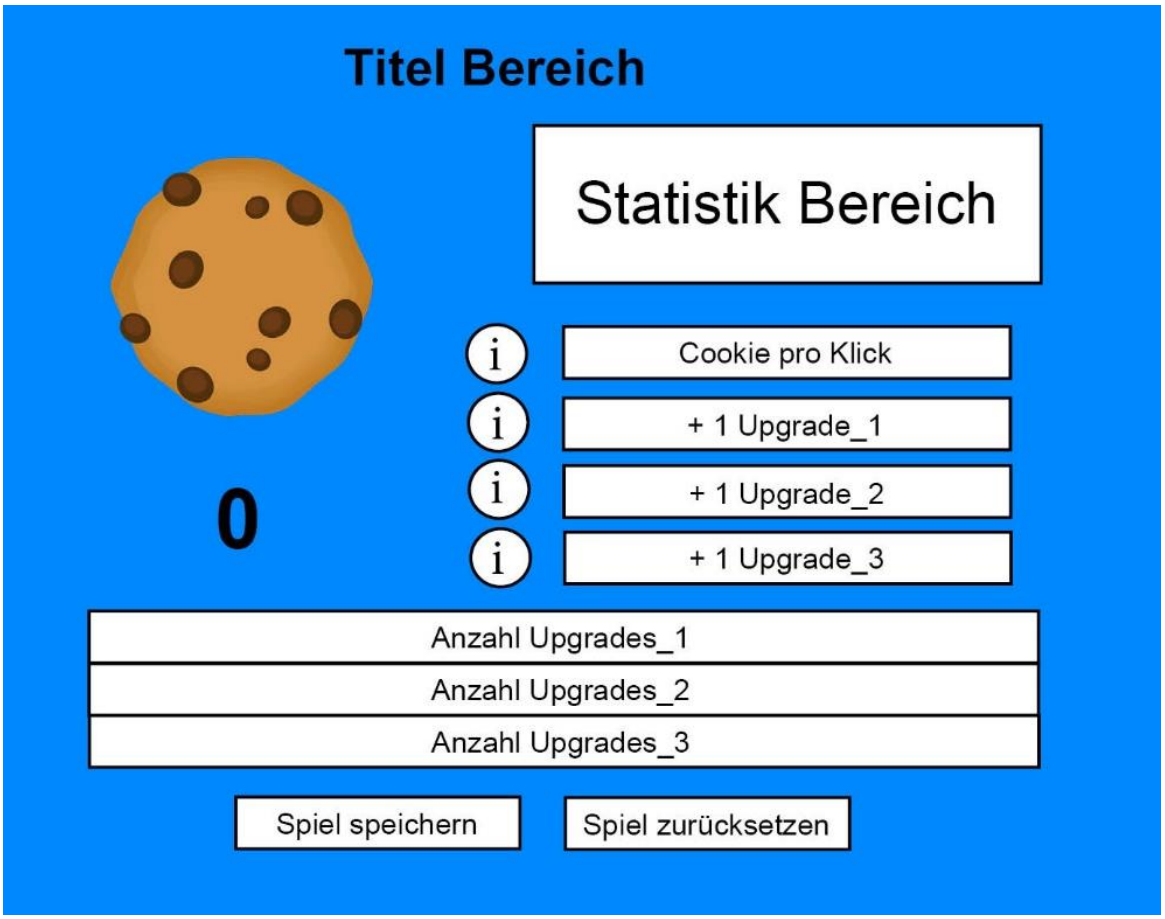
Ganz oben auf der Seite muss unser beider Name stehen, ist ja unser Projekt. Der Klick-Bereich, also dort wo man dann auf unserem Cookie klickt, muss gross und präsent auf dem Screen angezeigt werden. Die aktuelle Anzahl an Cookies wird dabei gleich unterhalb vom Cookie angezeigt und passt sich dann entsprechend der aktuellen Werten an. Ebenfalls soll die Möglichkeit bestehen, eine Statistik der gesammelten Cookies, präsent auf dem Screen zu sehen. Diese Statistik soll unter anderem die aktuelle Anzahl an Cookies, die Anzahl an totalen Cookies, welche man durch einen Click bekommt und die pro Sekunde erhaltenen Cookies der Upgrades wiedergeben.

Gleich unterhalb der der Statistik sollen die Upgrade-Möglichkeiten angezeigt werden und mit einem einfach klick ausgeführt bzw. entsprechend der Statistik und den Cookie-Zähler anpassen. Es soll auch die Möglichkeit bestehen, an Infos zu kommen wie viel die Upgrades kosten und die Kosten der Upgrades danach, die sich pro Benutzen jeweils

erhöhen sollen. Die Anzahl der gesammelten Upgrades werden grafisch, wie auf der originalen Cookie Clicker Seite, mit kleinen Bildern wiedergeben.

Wir wollen es dem Benutzer ermöglichen, seine Spielstände abzuspeichern und auch diese komplett zurück zu stellen. Die ganze Seite sollte auch responsive sein.

Unser Screendesign:



Unser Terminplan:

Datum	Arbeit	Ziel
10.11.2020	**Start Projektarbeit** **Gruppeneinteilung / Infos sammeln / erstes Dokumentieren** **--Benötigte Zeit: 1.5 Stunden**	• **Infos zur Projektarbeit erhalten (evtl. Fragen klären)** • **Mit Dokumentation beginnen**
14.11.2020	**Gruppenaustausch und erstellen Konzept** **Screendesign erstellen und gemeinsam Besprechen** **--Benötigte Zeit: 1.5 Stunden**	• **Abschluss Konzept** • **Screendesign erstellen und abschliessen**
17.11.2020	**Beginn Programmierung Applikation / erstes Layout setzen und Funktionen einbinden** **Dokumentation weiterführen** **--Benötigte Zeit: 1.5 Stunden**	• **Mit der Programmierung anfangen (index.html, style.css und script.js erstellen und bearbeiten)**
20.11.2020	**Austausch zu bereits erreichtem und klären von Problemen / Umsetzung gemäss Screendesign** **--Benötigte Zeit: 3 Stunden**	• **Applikationsumstellung gemäss Screendesign so weit wie möglich kommen**
24.11.2020	**Dokumentation weiterführen** **An Applikation weitermachen / Bilder setzen, Variablen definieren und Funktionen testen** **--Benötigte Zeit: 1.5 Stunden**	• **Dokumentation weiterführen** • **Auf Funktionen Testen**
29.11.2020	**Dokumentation weiterführen** **An Applikation weitermachen / Variablen definieren und Funktionen testen** **Gruppenaustausch (Stand abklären)** **--Benötigte Zeit: 4 Stunden**	• **Dokumentation weiterführen** • **Auf Funktionen Testen** • **Abklärung nächster Ziele und Ziele/Erweiterungen festlegen**
01.12.2020	**Dokumentation weiterführen** **An Applikation weitermach / neue Funktionen einbinden und testen evtl. abschliessen** **--Benötigte Zeit: 1.5 Stunden**	• **Dokumentation weiterführen** • **Auf Funktionen Testen und allgemeine Probleme lösen**
05.12.2020	**Browsertests durchführen** **Dokumentation weiterführen** **--Benötigte Zeit: 1 Stunde**	• **Webseite auf Chrome, Firefox, Microsoft Edge und Internet Explorer testen** • **Dokumentation weiterühren**

12.12.2020	**Projektarbeit «finishen» / Alles nochmal durchgehen / Abschlusstests** **Dokumentation weiterführen** **--Benötigte Zeit: 3.5 Stunden**	• **Applikation abschliessen** • **Dokumentation weiterführen**
15.12.2020	**Dokumentation abschliessen und Abgabe Projektarbeit** **--Benötigte Zeit: 1.5 Stunden**	• **Dokumentation beenden und Projekt abgeben**
Gesamtzeit	**--Benötigte Gesamtzeit: 20.5 Stunden**	

Realisation:

Erstellen der Applikation:

Jetzt konnten wir mit unserer Webapplikation anfangen und erstellten nach unserer Idee die HTML, CSS und JavaScript files und füllten diese mit dem Inhalt ab bzw. den Funktionen ab.

Aufbau head Bereich:

```
<head>
    <!--  Meta Einstellungen  -->
    <title>Cookie Clicker</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!--  Files integriegren  -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <link rel="stylesheet" href="css/style.css">
    <script type="text/javascript" src="script.js"></script>

    <link rel="apple-touch-icon" sizes="180x180" href="favicons/apple-touch-icon.png">
    <link rel="icon" type="image/png" sizes="32x32" href="favicons/favicon-32x32.png">
    <link rel="icon" type="image/png" sizes="16x16" href="favicons/favicon-16x16.png">
    <link rel="manifest" href="favicons/site.webmanifest">
    <link rel="mask-icon" href="favicons/safari-pinned-tab.svg" color="#5bbad5">
    <meta name="msapplication-TileColor" content="#da532c">
    <meta name="theme-color" content="#ffffff">

</head>
```

Im head Bereich werden einzelne Meta Einstellungen und die zu integrierten Files wiedergeben bzw. abgelegt. Dies ist nötig da wir darüber jQuery über eine externe Quelle einbinden und auch unsere Favicons Settings ablegen.

Aufbau Cookies und Statistiken:

```
    <!-- Cookie -->
    <div class="cookie_img">
        <img class="cookie" src="img/cookie_v3.png" onclick="playAudio('audio/audio.mp3')">
    </div>

    <!-- Anzahl Cookies -->
    <div class="amount"></div>
</div>
<div class="bodyWrapperRight">

    <!-- Statistiken -->
    <div class="overview">
        <div class="stats_title">Statistik</div>
        <p class="statsCategories">
            <div>
                <span>Aktuelle Anzahl Cookies: </span><span class="spanCookies">0</span>
            </div>
            <br>
            <div>
                <span>Total Cookies pro Klick: </span><span class="spanClicksValue">1</span><br>
            </div>
            <div>
                <span>Cookies pro Sekunde durch Bauern: </span><span class="spanBauernValue">0</span><br>
            </div>
            <div>
                <span>Cookies pro Sekunde durch Fabriken: </span><span class="spanFabrikValue">0</span><br>
            </div>
            <div>
                <span>Cookies pro Sekunde durch Portale: </span><span class="spanPortalValue">0</span><br>
            </div>
        </p>
    </div>
```

Im darauffolgenden Code wird das Cookie als Bild, die Anzahl Cookies und die Statistik aufgelistet.

Aufbau Upgrades:

```
<!--  Upgrades  -->
<div class="upgrades">
    <div class="clickUpgradeWrapper">
        <div class="infoTriggerButton">?</div>
        <div class="upgradeNode">+1 Cookie pro Klick</div>
        <div class="infoNode clickInfo">
            <div class="infoTriangle"></div>
            <p class="kostenUpgrade">Kosten: <span>10</span> cookies</p>
            <p class="info">Du erhälst 1 Cookie pro Klick mehr wen du dieses Upgrade machst</p>
        </div>
    </div>
    <div class="bauerUpgradeWrapper">
        <div class="infoTriggerButton">?</div>
        <div class="upgradeNode">+1 Bauer </div>
        <div class="infoNode bauerInfo">
            <div class="infoTriangle"></div>
            <p class="kostenUpgrade">Kosten: <span>50</span> cookies</p>
            <p class="info">Produziert jede Sekunde 1 Cookie</p>
        </div>
    </div>
    <div class="fabrikUpgradeWrapper">
        <div class="infoTriggerButton">?</div>
        <div class="upgradeNode">+1 Fabrik</div>
        <div class="infoNode fabrikInfo">
            <div class="infoTriangle"></div>
            <p class="kostenUpgrade">Kosten: <span>200</span> cookies</p>
            <p class="info">Produziert jede Sekunde 5 Cookies</p>
        </div>
    </div>
    <div class="portalUpgradeWrapper">
        <div class="infoTriggerButton">?</div>
        <div class="upgradeNode">+1 Portal</div>
        <div class="infoNode portalInfo">
            <div class="infoTriangle"></div>
            <p class="kostenUpgrade">Kosten: <span>1000</span> cookies</p>
            <p class="info">Produziert jede Sekunde 20 Cookies</p>
        </div>
    </div>
</div>
```

In diesem Bereich werden die Upgrades aufgelistet sowie der Infobereich.

Aufbau Fehler Konsole, gesammelte Upgrades und einbinden der Buttons:

```
<h2>Fehler:</h2>
<div class="console">

</div>
<div class="map">
    <h2>Gesammelte Upgrades:</h2>
    <div class="bauernKarte">

    </div>
    <div class="fabrikKarte">

    </div>
    <div class="portalKarte">

    </div>
</div>
<div class="buttonWrapper">
    <div id="saveButton">Status Speichern</div>
    <div id="resetButton">Spiel zurücksetzen</div>
    <div id="colorButton">Hintergrundfarbe ändern</div>
</div>
```

Am Schluss wird die Fehler Konsole und die gesammelten Upgrades aufgelistet und darauffolgend die Buttons mit den jeweiligen Funktionen zum Speichern, zurücksetzen zum Ändern der Hintergrundfarbe. Den dritten Button erklären wir in einem Test.

Unser «Secret Level»:

Wir haben uns noch eine coole Funktion einfallen lassen, welches aktiviert wird, wenn man bestimmte Voraussetzungen im Spiel erfüllt und korrekt abspeichert. Aber um dieses «Secret Level» freizuschalten muss entweder einfach Versuchen die Voraussetzungen zu erraten oder man untersucht die Seite und findet den Hinweis im Code.

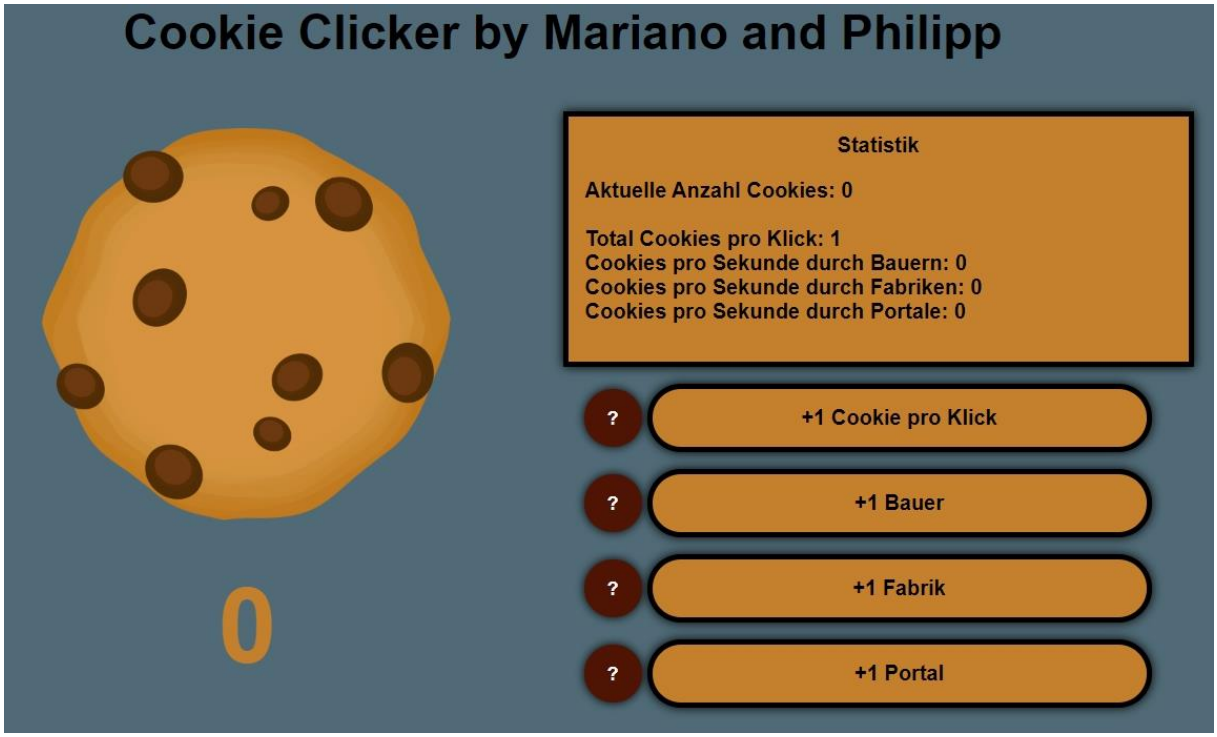
Cookie Clicker auf dem HOMM Interactive Server:

Um besser arbeiten zu können wurde Mariano gestattet das Projekt auf dem Homm Interactive Server zu programmieren. Dies bedeutet das Mariano alles auf dem Server so eingerichtet hat, dass wir beide mit einer FTP Verbindung zugriff darauf haben und so die Daten hoch und runterladen können. Dies hilft uns, dass wir nicht die ganze Zeit die Dateien uns gegenseitig schicken müssen, sondern diese einfach hochladen können damit der andere dies dann runterlädt. Dazu kommt noch, dass wir das Thema Tod-flexen wollten und JavaScript durchgespielt haben.

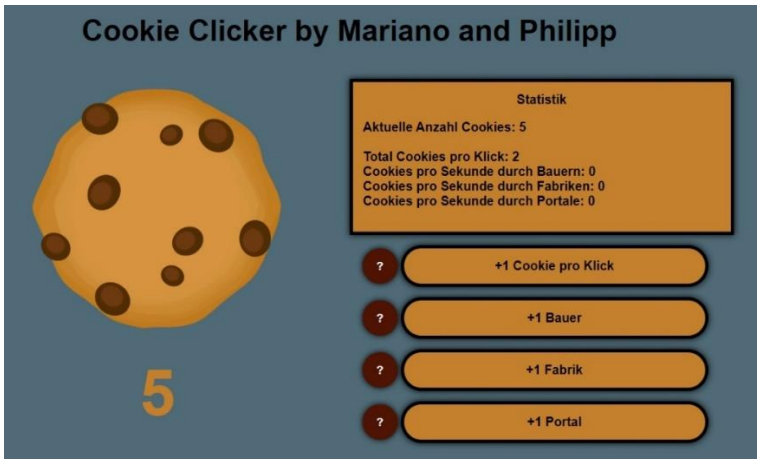
Tests:

Statistiken anzeigen und Variablen in Local Storage speichern:

```
// Statistiken
setInterval(function () {
    document.getElementsByClassName("amount")[0].innerHTML = cookiecount.toLocaleString();
    document.getElementsByClassName("spanCookies")[0].innerHTML = cookiecount.toLocaleString();
    document.title = cookiecount + " cookies | Cookie Clicker by P&M"; // Tab Text updaten
}, 0);

// Variabeln in Local Storage speichern
$('#saveButton').click(function () {
    saveCurrentStatus()
});
```



Zustand vor dem Test. Man erkennt das noch keine Cookies generiert und noch kein Upgrade hinzugefügt wurde.



Zustand nach dem Test. Ich habe für diesen Test 15-mal auf dem Cookie geklickt und mir dann das erste Upgrade gekauft. Es sollten also von den 15 Cookies 10 abgezogen werden und in der Statistik sowohl die aktuelle Anzahl an Cookies als auch den Totalwert der Cookies pro Klick wiedergeben werden. Mit dem Klick auf den «Status Speichern» Button wurden dann die Werte abgespeichert und beim neu laden der Seite sind die Werte trotzdem gleichgeblieben. Der Test war also erfolgreich.

Zufällige Hintergrundfarbe beim Klicken auf Button erzeugen:

```
// Zufällige Farbe erzeugen und diese per Klick auf den Hintergrund setzen (16.8 Millionen möglichkeiten da RGB);
$("#colorButton").click(function () {
    // Zufällige Farbe aus RED;
    var r = Math.floor(Math.random() * 256);
    // Zufällige Farbe aus GREEN;
    var g = Math.floor(Math.random() * 256);
    // Zufällige Farbe aus BLUE;
    var b = Math.floor(Math.random() * 256);
    console.log(r, g, b);
    var color = "rgb" + "(" + r + "," + g + "," + b + ")";
    console.log(color);
    $('body').css('background-color', color);
});
```

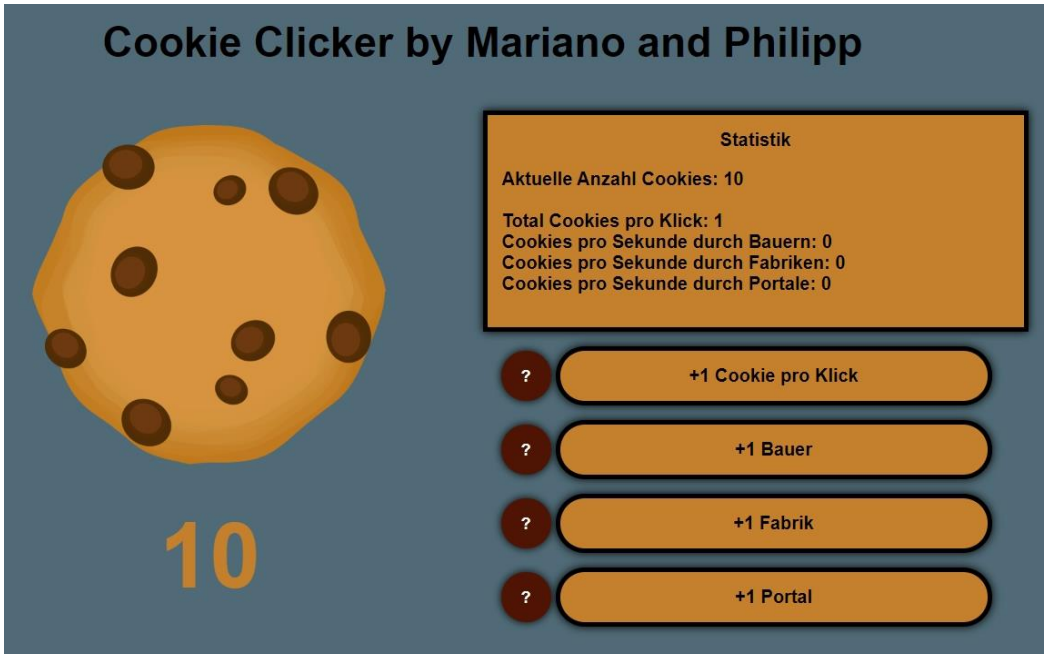
Diese Funktion haben wir im Nachhinein hinzugefügt da wir uns nie richtig einig waren, welche Hintergrundfarbe eigentlich richtig passen würde. So entschlossen wir uns mit einem Script eine zufällige Hintergrundfarbe basierend auf RGB zu generieren. So können nach dem Zufallsprinzip (in der Theorie) bis zu 16.8 Millionen Farben generiert werden.

Bei diesem Test funktionierte ebenfalls alles. Die Hintergrundfarbe änderte sich beim Klicken auf den Button in eine zufällige Farbe.

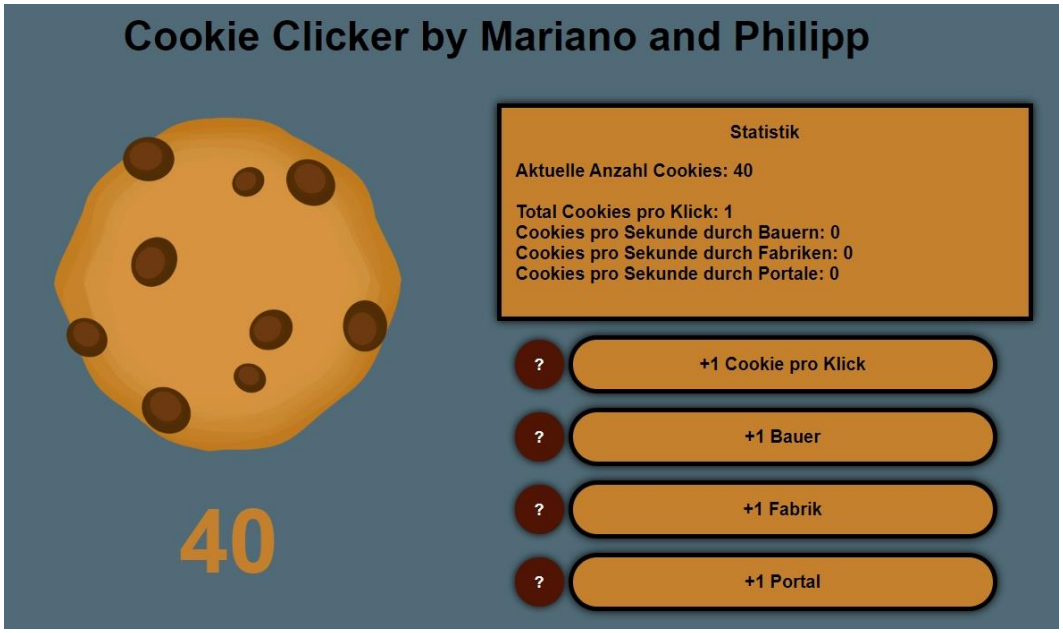
Cookies beim Klicken hinzufügen:

```
// Cookie hinzufügen bei Klick
$('.cookie_img img').click(function () {
    cookiecount = cookiecount + addCountClick; // Cookie bei Klick hinzufügen
    $("#cookieCount").value = cookiecount; // Cookie Zähler updaten
})
```

Ziel dieses Tests ist es, dass beim Klicken auf dem Cookie (also der Bilddatei) ein Cookie hinzugefügt wird und der Cookie-Zähler in der Statistik dabei angepasst wird.



Zustand vorher

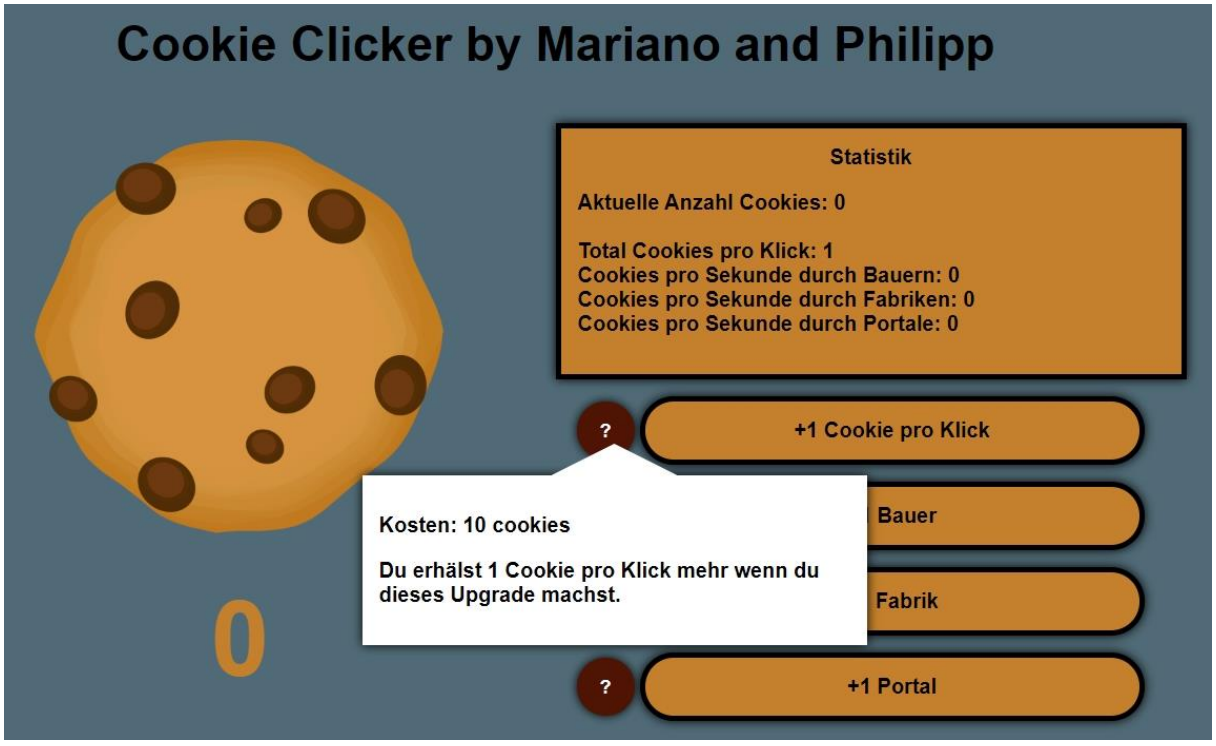


Zustand, nachdem 40-mal auf dem Cookie (Bilddatei) geklickt wurde. Der Test war auch dieses Mal erfolgreich.

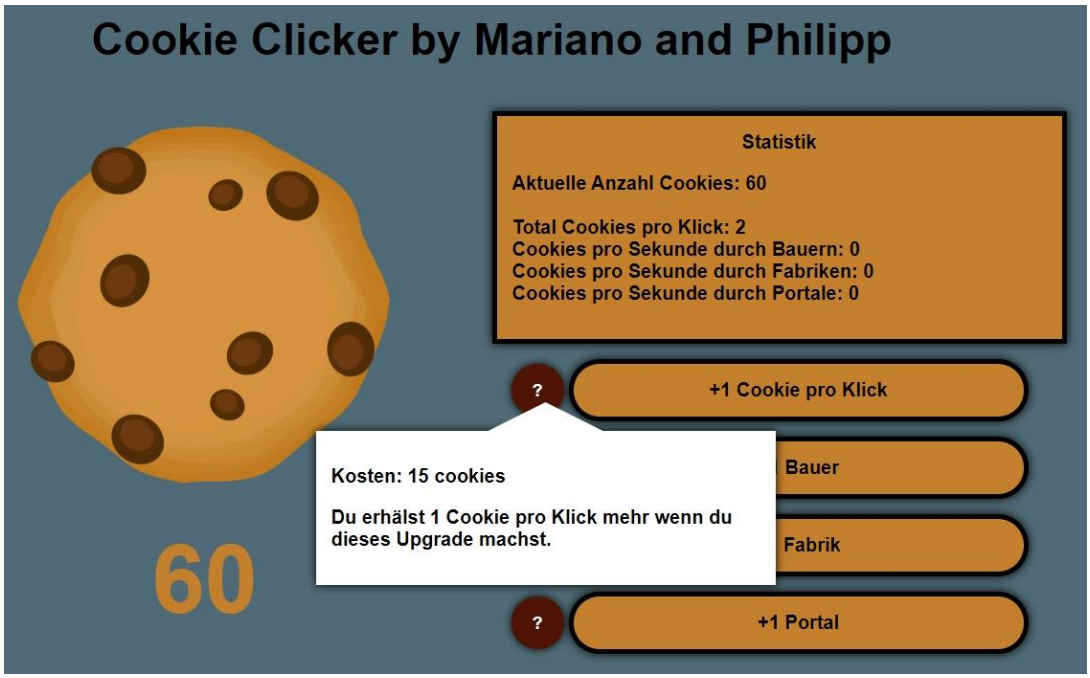
Erstes Upgrade testen mit if else Funktion:

```
// erstes Upgrade
// Cookie pro Klick + 1
$('.spanClicksValue').text(addCountClick);
$('.clickUpgradeWrapper .upgradeNode').click(function () {
    if (cookiecount >= upgradeClickCountCost) { // wen man genug Cookies besitzt
        addCountClick = addCountClick + 1; // Cookie pro Klick + 1
        cookiecount = cookiecount - upgradeClickCountCost; // Kosten berechnen

        upgradeClickCountCost = upgradeClickCountCost * 1.5; // kosten erhöhen
        upgradeClickCountCost = Math.round(upgradeClickCountCost / 5) * 5; // kosten immer auf 0.5 oder ganze Zahlen runden

        $('.clickUpgradeWrapper .kostenUpgrade span').text(upgradeClickCountCost); // Statistik updaten
        $('.spanClicksValue').text(addCountClick); // Kosten in der Info Box updaten
    } else { // wen man zu wenig Cookies
        var cookieDiffrence = upgradeClickCountCost - cookiecount;
        // Konsole gibt meldung ("Du hast nicht genügend Cookies: dir fehlen " + cookieDiffrence + " Cookies")
        var hours = (new Date()).getHours();
        var minutes = (new Date()).getMinutes();
        var seconds = (new Date()).getSeconds();
        $('.console').append('<p><span>' + hours + ':' + minutes + ':' + seconds + ' ></span> Du hast nicht genügend Cookies: dir fehlen ' + cookieDiffrence + ' Cookies um ein Click-Upgrade zu kaufen.</p>')
    }
})
```

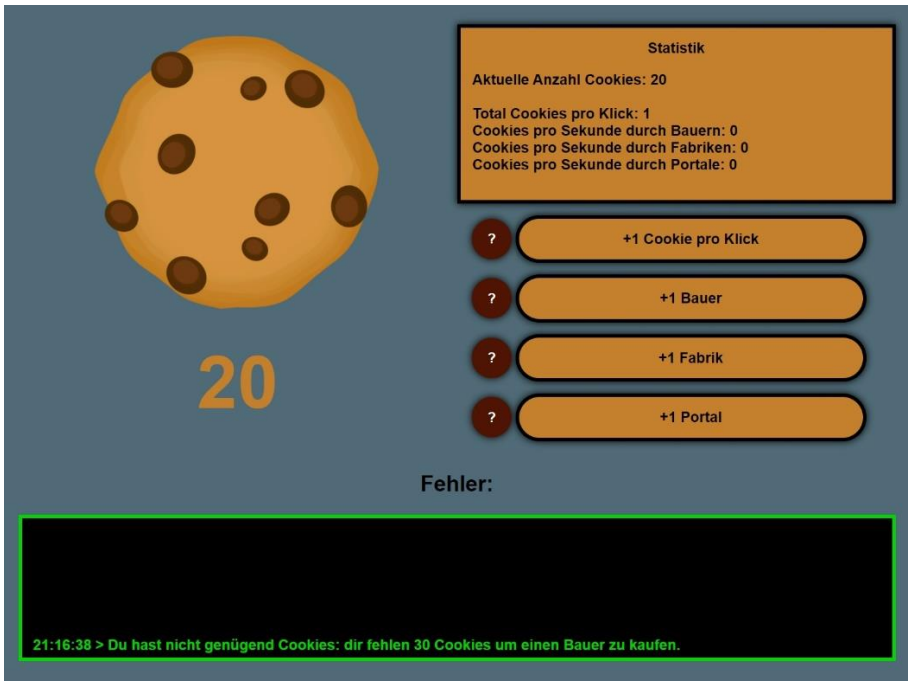
Bei diesem Test soll es möglich sein das erste Upgrade betätigen zu können. Es wird überprüft, ob man genügend Cookies für das Upgrade zur Verfügung hat und wenn ja, wird jeweils ein Cookie hinzugefügt und die neuen Kosten für ein weiteres Upgrade berechnet. Die Kosten werden dabei mit dem Wert mal 1.5 erhöht und dabei immer auf 0.5 oder ganze Zahlen gerundet. Wenn dieser Wert nicht erreicht wird gibt die Konsole eine Meldung und berechnet den noch zu benötigenden Restwert für das Upgrade mit aktueller Zeitangabe.



Zustand beim starten des Spiels mit Auflistung der Kosten und Infos für das erste Upgrade.



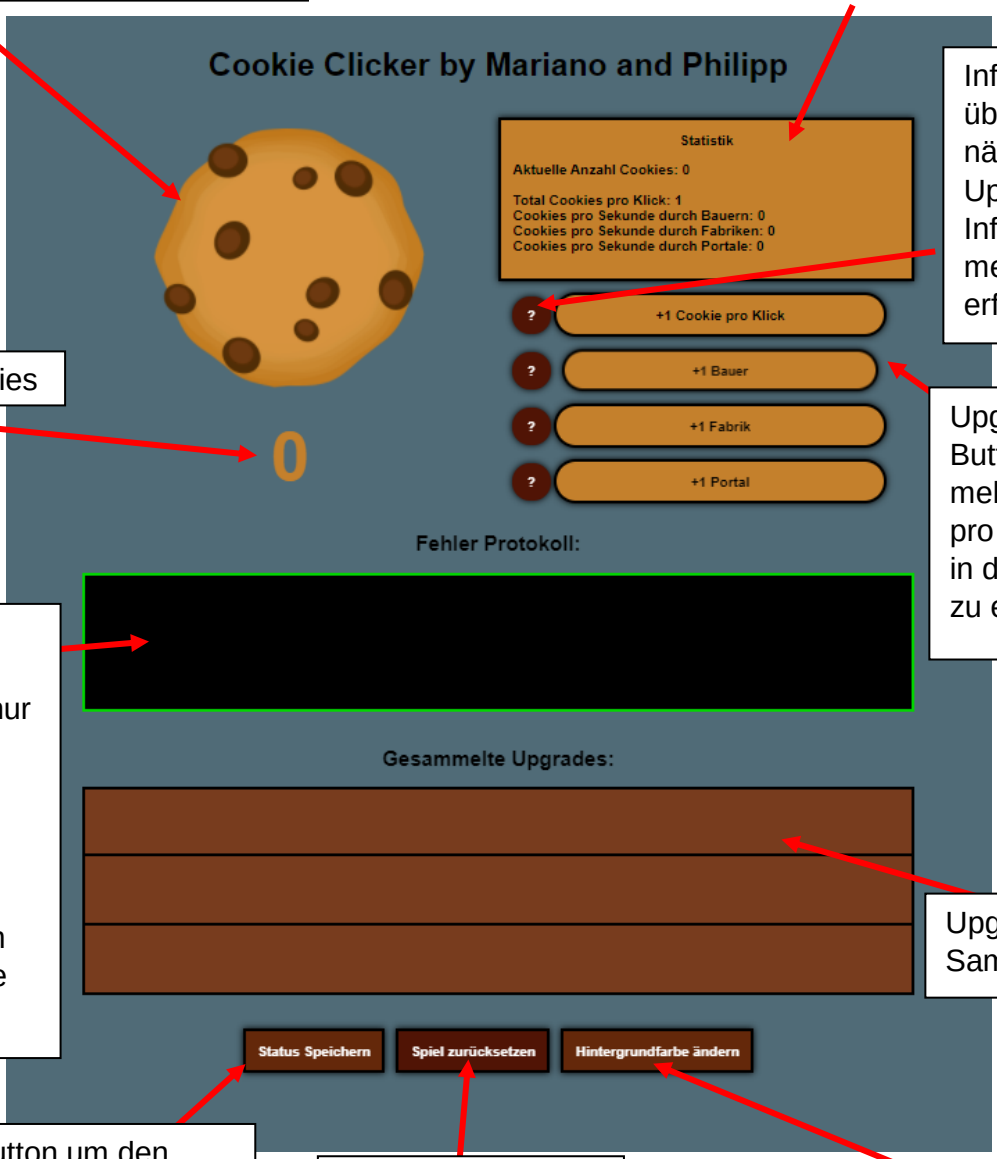
Zustand nach dem ersten Upgrade. Bei genügender Anzahl an Cookies konnte man das erste Upgrade kaufen und die Statistik wurde entsprechend angepasst. Will man nun das gleiche Upgrade nochmal machen, so werden in der Infobox die Kosten mal 1.5 erhöht (von 10 auf 15 in diesem Beispiel).



Sollte ich beim Klicken des Upgrade Buttons nicht genügend Cookies zur Verfügung haben, gibt mir der Browser folgende Meldung und rechnet mir die Anzahl fehlender Cookies nach. In diesem Beispiel hat man das erste Upgrade bereits getätigt und braucht nun 50 Cookies, um das Upgrade nochmals machen zu können. Da man aber nur insgesamt 20 Cookies gesammelt hat (weil diese durch das Bauer-Upgrade abgerechnet wurden) hat man entsprechend 30 Cookies zu wenig, um das Upgrade erneut machen zu können.

Kurzanleitung:

Wie funktioniert unsere Webapplikation genau?



Cookie zum Klicken um Cookies zu bekommen

Statistik-Bereich mit Auflistung der aktuellen Anzahl an Cookies und Upgrades

Info-Button um über die nächsten Upgrade Informationen mehr zu erfahren

Anzahl Cookies

Upgrade-Buttons um mehr Cookies pro Klick oder in der Sekunde zu erhalten

Fehler Protokoll (Inhalt wird nur bei nicht genügender Anzahl an Cookies angezeigt. Wird nicht im local Storage gespeichert)

Upgrade Sammlung

Button um den Spielstand im local Storage abzuspeichern

Button um das Spiel zurückzusetzen

Button um Hintergrundfarbe zu ändern

Reflexion:

Was ging gut? Was schlecht?

Unser Team war motiviert und bereit dieses Projekt in Angriff zu nehmen. Die ersten Gespräche untereinander verliefen gut und es gab in der Startphase keine grösseren Probleme. Daher nahmen wir es mit der Zeitplanung nicht so genau wie wir es eigentlich sollten. Daraus entstanden nacheinander viele kleine Probleme und wir mussten uns mehrmals um Änderungen des Zeitplans bemühen. Ging es dann an die Arbeit, verlief alles gut. In Gruppenarbeiten hat jedes Mitglied so seine Stärken und Schwächen. Das machten wir uns auch zunutze verteilten die Aufgaben entsprechend der jeweiligen Stärken. Jeder kam zu Wort und wir waren auch meist der gleichen Meinung.

Wo gab es warum Probleme / Schwierigkeiten?

Wir müssen zugeben, dass wir die Zeitaufwände nicht wirklich gut geplant hatten. Die Recherchen gingen länger als zuerst gedacht. Dadurch mussten viele der zuerst geplanten Termine verschoben werden und es entstand grosser Zeitdruck innerhalb des Teams. Es ist daher wichtig das für solche Projekte fixe Termine und Aufgaben eingehalten werden. Auch wenn es sich hier nicht um ein riesiges Projekt handelt, haben wir uns zu Beginn völlig mit der Zeiteinteilung verschätzt und konnten erst später diese beheben. Wir können uns nicht einfach zwischendurch zurücklehnen und manches aufs Wochenende schieben. Wenn wir eine Arbeit jetzt erledigen können, sollten wir es auch dann erledigen.

Wie haben wir diese gelöst und was würden wir das nächste Mal anders machen?

Wenn es in einer Gruppenarbeit Probleme gibt sollte man gemeinsame Lösungswege durchgehen. Neben der Zeit in der Schule benutzten wir für Zuhause das Programm «Discord», um gemeinsam über die Vorgehensweise oder der gemachten Arbeit reden zu können und auch zum Datenaustausch. Auch das Problem mit dem Fokussiert bleiben regelt man am besten so, dass man sich gegenseitig Ziele bis zu einem bestimmten Zeitpunkt gibt und sich auch gegenseitig kontrolliert. Das Schlüsselwort hierbei ist natürlich wieder Kommunikation, denn ohne Kommunikation können keine guten Gruppenprojekte entstehen. Beim nächsten Mal müssen wir deutlich fokussierter und engagierter an solche Projektarbeiten rann gehen.

Unsere Meinung können wir aber stolz auf unser Projekt sein, denn uns gefällt die Webapplikation und wir sind schon auf das nächste gespannt.