Projektbericht WEBopoly

Kitzbichler Jasmin, Stich Andrea, Treffer Manuel, Wilhelm Marco

1. Einleitung

Mit diesem Projektbericht beschreiben wir die Vorgehensweise, die aufgetretenen Probleme und Ideen und Lösungsansätze bei unserem Projekt „WEBopoly“. WEBopoly ist eine Web-Anwendung, die man von der grundlegenden Idee mit Monopoly vergleichen kann (zumindest auf Grund der grafischen Gestaltung).

Zu Beginn sollte das Spielfeld wie folgt Aussehen:

* 40 Felder, also gleich wie ein Monopoly-Spielfeld
* 4 Feldkategorien, immer gleich nacheinander angeordnet:
  + Alkohol – Felder
  + Activity – Felder
  + Fragen – Felder
  + Überraschungs – Felder

Es war geplant, dass insgesamt 10 Spieler oder weniger am selben PC/Laptop spielen können. Optionale Verbesserungsmöglichkeiten waren ein Youtube Tutorial und ein Pop-Up für die Aufgaben/Fragen.

1. Vorgehensweise
   1. Github

Die grundlegende Planung unseres Projektes fand im Zuge der Hausaufgaben für Web-Development statt.

Zu Beginn haben wir auf GitHub ein neues Repository erstellt und alle Projektteilnehmer hinzugefügt. In diesem Repository haben wir unseren Projektordner angelegt. Die ganze Abwicklung in unserem Projekt wurde über GitHub erledigt.

* 1. Scribbles / Wireframe

Anschließend haben wir im Projektteam Scribbles bzw. Wireframes erstellt, um das erste Konzept darzustellen und der geplanten Web-Anwendung ein grundlegendes Aussehen zu geben. (vgl. Abgabeordner)

* 1. ER – Diagramm

Da für unsere Web-Anwendung eine Datenbank benötigt wird, haben wir ein ER – Diagramm erstellt. (vgl. Abgabeordner)

* 1. HTML und CSS

Für den HTML und CSS Teil haben wir Bootstrap zur Unterstützung verwendet. Zu Beginn haben wir die Hauptseite, also die Website mit der eigentlichen Web-Anwendung erstellt. Dazu haben wir die Navbar, die Buttons, und die Punktetabelle von Bootstrap übernommen und angepasst. Das „Spielbrett“ haben wir mittels Div-Boxen erstellt.

* 1. Logo und Spielfiguren

Das Logo und die Spielfiguren haben wir selbst mit dem Programm Illustratorerstellt. Die Spielfiguren werden für das Spiel selbst benötigt, jeder Spieler bekommt einen Emoji als Spielfigur zugewiesen.

* 1. Würfel

Den Würfel haben wir mithilfe von JavaScript erstellt. Es gibt einen Button und ein Bild mit der Würfelanzahl. Die JavaScript Funktion „myFunction()“ bewirkt, dass eine Zufallszahl von eins bis sechs generiert wird. Diese Zufallszahl ist dann der gewürfelte Wert. Des Weiteren bewirkt die Funktion, dass eines von sechs Bildern ausgewählt wird, welches einen Würfel mit dem jeweils generierten Wert anzeigen lässt.

* 1. Spieleranzahl und Spielernamen

Weiterführend haben wir 2 weitere HTML Seiten erstellt, spieleranzahl.html und spielernamen.html. Die Spieleranzahl-Seite ist die erste Seite, die nach dem Login aufgerufen wird. Sie besteht aus unserem Logo, einer Überschrift, einem Input-Feld, bei dem nur Zahlenwerte von 1 bis 8 eingegeben werden können und einem Button um weiterzuspringen. Nachdem die Spieleranzahl eingegeben und der Button gedrückt wurde kommt man auf die Spielernamen-Seite. Die Seite sieht gleich aus wie spieleranzahl.html, nur gibt es anstatt dem Input-Feld für Zahlenwerte bis zu 8 Input-Felder, in der jeder Spieler seinen Namen eingeben kann und sieht, welche Spielfigur ihm zugewiesen wird.

* 1. Fragen bzw. Aufgaben

Durch den Spielmechanismus muss immer dann, wenn ein Spieler gewürfelt und sich auf ein anderes Feld bewegt hat eine Aufgabe/Frage, abhängig von der Feldkategorie, erscheinen. Dies haben wir mit zwei Overlays gelöst. Am Anfang haben wir einen Button auf der spielfeld.html Seite implementiert. Durch das Klicken auf diesen Button soll ein Overlay angezeigt werden, auf dem der Zugspieler und die Aufgabe/Frage ersichtlich ist. Das zweite Overlay dient lediglich dazu, dass der äußere Bereich verdunkelt wird. Weiters gibt es auf dem Overlay zwei Buttons mit der Überschrift „Aufgabe erledigt?“. Wenn die Aufgabe erledigt wurde, kann man den „Ja“ Button, ansonsten den „Nein“ Button anklicken. Durch das Anklicken verschwindet das Overlay wieder.

Anschließend haben wir als Datenquelle für die Aufgaben/Fragen für jede Feldkategorie ein JavaScript Array erstellt, in dem verschiedene Testaufgaben/Testfragen erfasst sind. Zu diesem Zeitpunkt wurde nur ein Array verwendet, die Implementierung der weiteren JavaScript Methoden haben wir später vorgenommen.

* 1. Implementierung der Spiel-Logik

Auch der Code für die Funktionalität des Spieles wurde mit JavaScript programmiert. Um einen grundlegenden Lösungsansatz zu finden und da es noch nicht möglich war, die Spieleranzahl zu übergeben, haben wir die Funktionalität anfangs nur für einen Spieler programmiert.

Jedes Feld auf unserem Spielbrett (insgesamt gibt es 12) hat eine ID, das erste Feld hat die ID 1, das Zweite 2, usw. Die Funktion „movePlayer()“ macht es uns möglich, eine Spielfigur (also ein .png Bild, das am Beginn des Spieles im Feld mit der ID 1 steht) auf das richtige Feld nach dem Würfeln zu bewegen.

Es gibt ein Array, in dem jeder Eintrag die Position eines Spielers darstellt, also:

Array[0] = Position von Spieler 1

Array[1] = Position von Spieler 2

…

Da zu Beginn des Spieles jeder Mitspieler auf der Position 1 steht, werden die Einträge mit dem Wert 1 initialisiert. Des Weiteren gibt es eine Variable, in der der aktuelle Spieler, also der Zugspieler abgespeichert wird, beginnend mit 1. Wenn der letzte Spieler an der Reihe war, wird die Variable „Zugspieler“ wieder auf 1 gesetzt, damit wieder der erste Spieler an der Reihe ist.

Da es nur 12 Spielfelder gibt, muss nach dem zwölften Feld wieder bei 1 begonnen werden, was mittels if-Anweisung implementiert wurde. Die if-Anweisung überprüft, ob die aktuelle Position des Spielers plus die gewürfelte Zufallszahl größer als 12 ist. Falls ja, werden die Zufallszahl und die alte Position zusammengezählt und der Wert 12 wird abgezogen, damit wird gewährleistet, dass nach 12 bei 1 weitergezählt wird.

Es müssen natürlich auch die Spielfiguren auf das richtige Feld bewegt werden. Dazu wird die Spielfigur mithilfe von „document.getElementById“ angesprochen und durch die Befehle „.appendChild“ und „.innerHTML“ auf das richtige Feld verschoben.

Anschließend werden die Funktionen „on()“, „getTask()“ und „highlightPlayer()“ aufgerufen.

Die Funktion „on()“ bewirkt, dass nach dem würfeln das Overlay mit den Aufgaben/Fragen eingeblendet wird. Die Funktion „getTask()“ sorgt dafür, dass je nach Feldtyp eine zufällige Alkohol, Frage oder, Activity-Aufgabe im Overlay eingeblendet wird. Mit „highlightPlayer()“ zeigen wir die Spielfigur des aktuellen Spielers, also dem Spieler der die Aufgabe/Frage erledigen muss, an.

* 1. Enstand

Um das Spiel zu beenden, gibt es in der Navbar einen Button „Spiel beenden“. Wenn der Button angeklickt wird, gelangt man auf die Seite endstand.php. Auf dieser befindet sich eine Punktetabelle, auf der die besten vier Spieler angezeigt werden. Weiters kann man mit dem Button „Neues Spiel“ ein neues Spiel starten.

* 1. Datenbank

Der nächste Schritt war es, die Datenbank mit „phpmyadmin“ zu implementieren. Insgesamt gibt es 2 Tabellen, „user“ und „webopoly“. In der user-Tabelle gibt es folgende Attribute:

* id: INT AUTO INCREMENT, ein fortlaufender Primärschlüssel
* name: hier werden die Benutzernamen der User abgespeichert
* password: hier werden die Passwörter der User verschlüsselt abgespeichert

Die Tabelle „webopoly“ hat folgende Attribute:

* id: INT AUTO INCREMENT, ein fortlaufender Primärschlüssel
* player: hier werden die Namen der Spieler von spielernamen.php abgespeichert (Typ: VARCHAR)
* position: hier wird die aktuelle Position des Spielers auf dem Spielbrett gespeichert (Typ: INT) (wird aktuell nicht mehr verwendet)
* points: hier werden die Punkte der Spieler abgespeichert (Typ: INT)
  1. Datenbank Einbindung

Die Datenbank haben wir mittels PHP eingebunden. Dazu haben wir die Dateien config.php, .htaccess sowie dbconfig.php angepasst. Die gesamten SQL Statements haben wir anschließend im WebopolyModel.php definiert und den jeweiligen Funktionen zugeordnet.

* 1. Übergabe von Werten mittels PHP

Unsere erste Überlegung war es, die Spieleranzahl über GET bzw. POST an die weiteren Seiten zu übergeben. Anschließend haben wir allerdings auf die $\_SESSION zugegriffen, da diese es erlaubt, Variablen auch über mehrere Seiten hin zu übergeben. Dieses Prinzip haben wir für die Spieleranzahl von index.php auf playername.php, von index.php auf game.php sowie von der index.php auf die endgame.php angewendet. Die restlichen Werte wurden zuerst in die Datenbank gespeichert und anschließend aufgerufen und ausgegeben.

* 1. JSON und AJAX

(Nachmittag mit Herr Prof. Hoover)

Da wir das Problem hatten, dass JavaScript für das Frontend und PHP für das Backend bestimmt ist, haben wir eine Schnittstelle benötigt, die es ermöglicht, Daten der einen Programmiersprache an die andere zu übergeben (in unserem Fall, um die Punkte der einzelnen Spieler zu setzen und den neuen Punktestand anschließend an die Datenbank zu übergeben)

* 1. Integration PHP mit MVC

(Nachmittag mit Herr Prof. Hoover)

(Leidergottes) als einer der letzten (und bis dato auch schwersten) Punkte stand die Implementierung unseres bisher verfassten Codes in das „Baseproject“ an.

Dazu mussten wir die einzelnen Seiten in die Views übertragen, neue Controller anlegen, die Routes ändern, evtl. das WebopolyModel anpassen.

* 1. SCSS

Unser bisheriger CSS Code wurde noch mittels SCSS einfacher gestaltet.

* 1. Kommentare

Zum Schluss haben wir noch den gesamten Code kommentiert.

* 1. Video und finaler Schliff

Zur Anleitung des Spiels haben wir noch ein Video produziert und das gesamten Spiel auf noch etwaige Fehler überprüft.