

Realisierung eines Rechners als Einzelseiten-Webapplikation

Individualprojekt

Cindy Hinze | Fachinformatiker Anwendungsentwicklung

Inhaltsverzeichnis

[Kapitel 1: Einleitung 1](#_Toc120275837)

[1.1: Projektumfeld 1](#_Toc120275838)

[1.2: Projektziel 1](#_Toc120275839)

[1.3: Projektbegründung 1](#_Toc120275840)

[1.4: Projektschnittstellen 2](#_Toc120275841)

[1.5: Projektabgrenzung 2](#_Toc120275842)

[Kapitel 2: Projektplanung 2](#_Toc120275843)

[2.1 Projektphasen 2](#_Toc120275844)

[2.2 Abweichungen von der Projektkarte 2](#_Toc120275845)

[2.3 Ressourcenplanung 3](#_Toc120275846)

[2.4 Entwicklungsprozess 3](#_Toc120275847)

[Kapitel 3: Analysephase 3](#_Toc120275848)

[3.1 IST-Analyse 3](#_Toc120275849)

[3.3 Projektkosten 3](#_Toc120275850)

[3.4 Nutzwertanalyse 3](#_Toc120275851)

[3.5 Anwendungsfälle 4](#_Toc120275852)

[3.6 Qualitätsanforderungen 4](#_Toc120275853)

[3.7 Lastenheft 4](#_Toc120275854)

[Kapitel 4: Entwurfsphase 4](#_Toc120275855)

[4.1 SOLL-Konzept 4](#_Toc120275856)

[4.2 Maßnahmen zur Qualitätssicherung 4](#_Toc120275857)

[4.3 Pflichtenheft 4](#_Toc120275858)

[Kapitel 5: Implementationsphase 4](#_Toc120275859)

[5.1 Implementierungsschritte 4](#_Toc120275860)

[Kapitel 6: Abnahmephase 4](#_Toc120275861)

[6.1 Abnahmephase 4](#_Toc120275862)

[Kapitel 7: Fazit 4](#_Toc120275863)

[7.1 SOLL/IST-Vergleich 4](#_Toc120275864)

[7.2 Lesson learned 4](#_Toc120275865)

[7.3 Ausblick 4](#_Toc120275866)

[Anhang 5](#_Toc120275867)

[A.1 Ressourcenplan und verwendete Hardware 5](#_Toc120275868)

# Kapitel 1: Einleitung

## 1.1: Projektumfeld

Das Projekt wurde von mir in der Annedore-Leber-Oberschule in Berlin durchgeführt. Die Annedore-Leber-Oberschule befindet sich im Annedore-Leber-Berufsbildungswerk und wurde im Jahre 1979 gegründet. Hier werden junge Menschen mit Behinderungen und besonderem Förderbedarf in über 35 Berufen ausgebildet.

Im Rahmen der Berufsschule, im Lernfeld 12a, erhielten wir den Auftrag ein Individualprojekt durchzuführen. Ein direkter Kunde existiert daher nicht. In meinem Individualprojekt erstellte ich einen Rechner für ein Brettspiel in Form einer Webbrowser-Applikation.

## 1.2: Projektziel

In einem Webbrowser seiner Wahl kann der Benutzer die entsprechende Webseite aufrufen und somit auf die Applikation zugreifen. Die Webseiten-Applikation besteht aus HTML-Formularelementen, in welche die erreichte Punktzahl für zum Beispiel Sterne eingetragen wird. Zudem werden alle Highscores von Ressourcen, Territorien, Münzen, Bonus- und Gesamtpunkte angezeigt. Dies geschieht schon beim Aufrufen der Webseite.

Nach dem Eintragen aller Punktzahlen kann der Nutzer auf den Button(„berechnen“) klicken. Die eingetragenen Punkte werden zusammengerechnet und im Input-Feld der Gesamtpunkte als Ergebnis angezeigt. Eine weitere Funktion des Buttons ist das Speichern der Daten in eine Datenbank. Hierbei werden die eingetragenen Werte per „POST“ an das Back-end (PHP) geschickt und in einer Datenbank hinterlegt. Aus derselben Datenbank werden die Highscores gelesen, die beim Aufrufen der Webseite zu sehen sind.

## 1.3: Projektbegründung

Bei dem Brettspiel des Projektes erlangt man für gewisse Sachen eine bestimmte Anzahl an Punkten. Die gesamten Punkte müssen am Ende zusammengerechnet und aufgeschrieben werden, um den Sieger zu ermitteln. Normalerweise geschieht dies schriftlich. Eine einfache Webapplikation erspart die Zeit beim Rechnen, sowie Papier, welches jede Runde verwendet werden würde. Zudem benötigt die Applikation keine zahlungspflichtige Software um Implementiert zu werden. Sollte man die Web-Applikation online stellen wollen, wären die Finanzierung eines Web-Host die einzigen Kosten bei diesem Projekt.

## 1.4: Projektschnittstellen

Die Webapplikation kann in jedem aktuellen Webbrowser auf Geräten wie PCs, Tablets und Smartphones genutzt werden.

Aktuell läuft die Applikation auf einem Apache-Server XAMPP (Localhost) mit MariaDB und PHP.

## 1.5: Projektabgrenzung

In dem Brettspiel des Projektes befinden sich sieben spielbare Fraktionen zur Auswahl. Das Projekt wurde auf eine runtergebrochen und umfasst dementsprechend nur die Allgemeine Eingabe und Berechnung. Das führt außerdem dazu, dass keine Liste benötigt wird, in der die erzielten Punkte in sortierter Reihenfolge aufgelistet werden, mit der man den Gewinner ermitteln könnte.

# Kapitel 2: Projektplanung

## 2.1 Projektphasen

|  |  |
| --- | --- |
| **Projektphase** | **Stunden** |
| 1 Analyse | 4 |
| 2 Entwurf | 9 |
| 3 Implementierung | 40 |
| 4 Test | 5 |
| 5 Dokumentation | 12 |
| **Gesamt** | **70** |

## 2.2 Abweichungen von der Projektkarte

Die Verfügbarkeit einer Fraktion von sieben wurde abgeändert. Statt eine der sieben Fraktionen zu sehen wurde eine allgemeine Ansicht umgesetzt.

Die geplante Umsetzung des MVC-Entwurfsmusters hat nicht stattgefunden. Grund dafür ist das beschränkt verfügbare Wissen zur Implementierung dieses Musters.

## 2.3 Ressourcenplanung

Die benötigten Ressourcen, sowie die verwendete Hard- und Software befinden sich im Anhang unter A.1 Ressourcenplan und verwendete Hard- und Software.

## 2.4 Entwicklungsprozess

Da es sich um ein Projekt mit einem kleineren Umfang handelt, habe ich mich für das Wasserfallmodell als Entwicklungsprozess entschieden. In diesem Fall fand zu Beginn des Projektes eine Anforderungsanalyse statt, in der festgestellt wurde, welche Ressourcen und Software für dieses Projekt benötigt werden. Außerdem wurde festgehalten, welche Anforderungen die Web-Applikation erfüllen sollte. In der darauffolgenden Design-Phase entwarf ich ein Mockup der Front-End-Ansicht, sowie ein ER-Diagramm für die benötigte Datenbank.

# Kapitel 3: Analysephase

## 3.1 IST-Analyse

Um den Gewinner einer Scythe-Runde ermitteln zu können, werden die erlangten Punktzahlen benötigt. Dieses Spielprinzip ist im Bereich der Brettspiele überwiegend standardisiert.

Aktuell ist es so, dass man zu jeder Runde Scythe einen Stift und einen Zettel zur Verfügung hat, auf welchem am Ende eines Spieldurchlaufs die Punkte der jeweiligen Spieler ausgerechnet und notiert werden. Sollte man nach einer gewissen Zeit wissen wollen, wer im Gesamten die meisten Punkte einer bestimmten Kategorie erreicht hat, ist ein Aufheben der Punktezettel aller vergangenen Spielrunden erforderlich.

Da der gesamte Vorgang zeitaufwendig ist sowie Papierressourcen und Platz für die gesammelten Punktezettel verbraucht, wurde sich eine Digitalisierung der Berechnung und eine Datenverwaltung gewünscht.

## 3.3 Projektkosten

Die verwendete Software Visual Studio Code, sowie der lokal eingerichtete Server XAMPP sind Open Source verfügbar. Für die verwendete Hardware sind ebenfalls keine weiteren Kosten angefallen, da diese von der Berufsschule bereitgestellt wurden. Da ich aufgrund dessen einen Stundensatz von 0€ hatte, blieben nur die Kosten für den Arbeitsplatz sowie Internet und Strom. Dafür wird von einem pauschalen Stundensatz von 10€ ausgegangen. Somit ergeben sich Projektkosten von 700,00€, da die Durchführungszeit von 70 Stunden x 10€ pro Stunde berechnet wurde. Dieser Betrag ist für das aktuelle Projekt allerdings nur ein theoretischer Wert.

## 3.5 Anwendungsfälle

Die Web-Applikation richtig sich an Brettspiel-Spielern des Brettspiels „Scythe“. Am Ende einer Spielrunde werden die Punkte für jeden Mitspieler einzeln ausgerechnet. Die Applikation kann über einen beliebigen Browser auf einem Smartphone oder Tablet genutzt werden. Bei Bedarf ist die Nutzung auf einem Desktop-PC ebenfalls möglich.

## 3.6 Qualitätsanforderungen

In der Web-Applikation werden Daten eingetragen, welche einerseits zur Berechnung dienen und andererseits in die Datenbank eingespeichert und auch wieder ausgelesen werden sollen. Hierbei ist es wichtig, dass die eingetragenen Daten ausschließlich Zahlen sind und auch vom Programm als Zahlen erkannt werden. Zudem sollen diese Zahlen eine maximale Länge von 1-3 Zeichen besitzen, abhängig vom erforderlichen Wert. Diese Werte dürfen ebenso nicht negativ sein und es darf keine Kommazahl bei der Berechnung rauskommen. Zuletzt soll die Applikation sicherstellen, dass alle Felder einen Wert erhalten und erst dann mit der Berechnung und der Datenspeicherung begonnen wird.

# Kapitel 4: Entwurfsphase

## 4.1 SOLL-Konzept

Um die Berechnung der im Spiel erreichten Werte erreichen zu können, wurden innerhalb eines HTML-Formelements Input-Felder erstellt, in welche diese Werte eingetragen werden können. Ausgenommen für Popularität, Fabrik und Sterne entschied ich mich für Select-Elemente, da diese Werte beschränkt sind. Nach dem Eintragen klickt man auf den „berechnen“-Button. Dieser enthält zum Teil die Formel zum Berechnen der gesamten Punktzahl und speichert diese dann in das Input-Feld „Gesamt“. Somit ist eine Darstellung der aktuell erreichten Punktzahl möglich. Der andere Teil des Buttons arbeitet mit einer PHP-Datei zusammen, welche per POST die eingetragenen Werte mithilfe eines SQL-Befehls in die Datenbank speichert. Eine weitere GET-Methode enthält eine SQL-Abfrage, welche beim Aufrufen der Webseite die erreichten Highscores ausgewählter Werte anzeigt.

## 4.2 Maßnahmen zur Qualitätssicherung

Bei den einzutragenden Zahlen darf es sich nur um Zahlen handeln. Um dies sicher zu stellen wurde allen Input-Elementen ein Type-Attribut zugeteilt, welches den Wert „number“ erhielt. Wenn man nun versucht Buchstaben in eins der Felder einzutippen, wird darin nichts zu sehen sein. Damit das Programm selbst die Werte ebenfalls als Zahlen erkennt, wurde in der Javascript-Datei den Werten die Funktion „ParseInt“ beigefügt. Kommazahlen können nur aufgrund der Tatsache, dass alle 2 Ressourcen Punkte bekommen, entstehen. Daher wurde dieselbe Funktion „ParseInt“ ebenfalls in die Berechnung der Territorienpunkte hinzugefügt. Innerhalb der Input-Elemente befinden sich unter anderem die Attribute „max“ und „min“. Diese definieren den Zahlenbereich, in welchem sich die eingetragenen Zahlen befinden dürfen. Alle beginnen mindestens mit „0“, was dafür sorgt, dass keine negativen Werte eingetragen werden können.

# Kapitel 5: Implementationsphase

## 5.1 Implementierung des Front-Ends

### 5.1.1 Implementierung der HTML-Seiten

Als erstes begann ich mit der Erstellung der *HTML*-Seite *index.html*. Als Grundlage dient ein *div*-Element, welches den Bereich des Inhaltes definieren soll, welcher sich im Zentrum des Bildschirms befinden soll. Innerhalb dieses Elements ist das *form*-Element. Hier sind alle Eingabefelder und der Button zu finden. Die *form*-Attribute *method* und *action* stellen eine Verbindung zur PHP-Datei her, welche es ermöglicht, dass die eingetragenen Werte weiterverwendet werden können. Die Eingabefelder wurden in einzelne *div*-Elemente eingesetzt, um sie dann mit *CSS* einfacher in ein Grid-Layout umsetzen zu können. Um das Anzeigen der Highscore möglich zu machen, wurde zu den ausgewählten Eingabefeldern ein normaler Textabschnitt beigefügt.

### 5.1.2 Implementierung von Javascript

Bei der Implementierung von Javascript wurde eine *app.js* erstellt. Hier wurden zu Beginn Variablen erschaffen, welche den Zugriff auf die Eingabefelder in der HTML-Datei ermöglichen. Im nächsten Schritt habe ich die Funktion *addEventListener* für den Button genutzt, welcher nun den Code ausführt, sobald der Button geklickt wird. Folgendes befindet sich in der Funktion:

Um die eingegebenen Daten für die Datenbank nutzen zu können, habe ich ein *FormData*-Objekt erstellt, welches automatisch die Werte der *form*-Elemente erfasst und diese dann mithilfe einer Netzwerkmethode übermitteln kann. In diesem Fall handelt es sich um die *fetch*-Methode. Da das Gesamtergebnis der Berechnung ebenfalls in die Datenbank gespeichert werden soll, wurde im nächsten Schritt der Rechenweg festgelegt und die entsprechenden Werte in der *FormData* aktualisiert.

Die Highscore-Werte sollen bereits beim Aufrufen der Webseite geladen werden. Ermöglicht wird dies durch eine weitere *fetch*-Methode außerhalb des Buttons. Hierbei wird durch die *scoreboard.php*-Datei ein Zugriff auf die Höchstwerte innerhalb der Datenbank ermöglicht und mithilfe der zu Beginn gesetzten Variablen in den Textabschnitten der *index.html* angezeigt. Sollte es noch keinen Highscore geben, weil es noch keine Einträge in der Datenbank gibt, wird der Wert *0* angezeigt.

Der Quellcode der *addEventListene*r-Funktion und der *fetch*- Methoden befindet sich im Anhang unter A.2 Implementierung von Javascript.

### 5.1.3 Implementierung des Layouts

Für das Layout habe ich eine *style.css* erstellt. In dieser legte ich die Schriftart *Papyrus* für die gesamte Seite fest, da ich eine einheitliche und zum Brettspiel passende Schriftart wollte. Passend zum Spiel suchte ich mir ein Bild aus, welches ich als Hintergrundbild verwendete. Damit es sich jeder Bildschirmgröße anpasst, habe ich die Hintergrundgröße auf *cover* gesetzt.

Um das gesamte Layout responsiv zu halten, habe ich bei allen Größen-Werten mit Prozenten gearbeitet. Ausgenommen die Schriftgröße. Hier habe ich mit der Methode *clamp* gearbeitet. Bei dieser Methode wird ein minimal-, maximal- und ein bevorzugter Wert festgelegt. Die Schriftgröße passt sich dabei der Größe des Fensters an, wobei diese weder unter den Minimalwert noch über den Maximalert gehen kann. Zudem habe ich mit dem *grid*-Display gearbeitet, bei welchem ich die Spalten und teilweise die Reihen des Gitters für verschiedene Displaygrößen festgelegt habe. Die verschiedenen Displaygrößen konnte ich mit der *@media*-Methode festlegen und so nun bestimmte Größen oder Anordnungen entsprechend anpassen.

Da die Eingabefelder den Typ *number* besitzen, haben diese standardgemäß an ihrer rechten Seite Pfeil zum Hoch- und Runterzählen. Diese wurden durch CSS entfernt.

Der Quellcode der *clamp*-Methode und der zum Entfernen der Pfeile der Eingabefelder befindet sich im Anhang unter A.3 Implementierung des Layouts.

## 5.2 Implementierung des Back-Ends

### 5.2.1 Implementierung des PHP-Programmteils

Da eine Verbindung zur Datenbank benötigt wird, habe ich die Datei connector.php erstellt. Diese Datei enthält folgenden Code:

$conn = mysqli\_connect("localhost","root","","scoreboard");

Hierbei erhält die Methode den Host, das Passwort und den Namen der Datenbank, welche sie zur Verbindung benötigt. Falls ein Fehler bei der Verbindung entsteht, wird durch eine weitere Methode eine Fehlermeldung ausgegeben. Ich habe mich dazu entschieden, den Connector in eine separate Datei zu schreiben, damit dieser in Zukunft bei Bedarf eingebunden werden kann und somit Platz im Code spart.

Im Anschluss erstellte ich die scoreboard.php. Zu Beginn des Codes wird mithilfe des eingebundenen Connectors eine Verbindung zur Datenbank hergestellt. Die aktuelle Version von PHP (8.1.6) enthält den Bug, dass selbst nicht-fatale Fehler eben als solche angesehen werden. Das führt dazu, dass der gesamte Code nicht ausgeführt wird. Um das zu verhindern habe ich die Methode *mysqli\_report(MYSQLI\_REPORT\_ERROR);* eingefügt. Sie unterdrückt den fatalen Fehler und der Code wird wieder ausgeführt.

Als nächstes findet eine if-Funktion statt. Diese überprüft die Requestmethoden, welche ausgeführt wurden. Als erstes wird die POST-Methode überprüft.

### 5.2.2 Implementierung der Datenbank

# Kapitel 6: Abnahmephase

## 6.1 Abnahmephase

# Kapitel 7: Fazit

## 7.1 SOLL/IST-Vergleich

## 7.2 Lesson learned

## 7.3 Ausblick

# Glossar

# Anhang

|  |  |  |
| --- | --- | --- |
| Typ | Bezeichnung | Rolle |
| Personal | | |
|  | Entwickler und Autor | Projektausführung und Dokumentation |
| Hardware | | |
|  | <PC von zuhause> | Arbeits- und Testgerät |
|  | Samsung Galaxy S20 FE | Testgerät |
|  | Samsung Galaxy Tab 6 | Testgerät |
| Software | | |
| Betriebssysteme | Microsoft Windows 10 | Arbeits-/Test-Betriebssystem |
|  | Google Android (verschiedene Versionen) | Test-Betriebssystem |
| Browser | Google Chrome (verschiedene Versionen) | Arbeits-/Test-Browser |
|  | Samsung Internet  Version 19.0.1.2 | Test-Browser |
|  | Microsoft Edge  Version 107.0.1418.42 | Test-Browser |
| Entwicklungs-Tools | Visual Studio Code | Code-Editor |
|  | Google Chrome DevTools | Debugging |
|  | phpmyadmin | Browseranwendung zum Erstellen und Bearbeiten der Datenbank |
|  | GitHub | Versionsverwaltung |
| Dokumentations-Tools | Microsoft Word | Programm zur Erstellung der Dokumentation |
|  | yEd | Programm zum Erstellen von Diagrammen |
| Server-Software | MariaDB  Version 10.4.24 | Datenbank-Server |
|  | Apache  Version 2.4.53 | Webserver |
|  | PHP  Version 8.1.6 | Serverseitige Programmiersprache |

## A.1 Ressourcenplan und verwendete Hardware