

Projektdokumentation



„Das Webportal zum Erstellen von JavaScript-basierten Minispielen“

Projektarbeit von:

Mike Dobrinski, Dennis Kohlmann, Sebastian Schröder und Lukas Trenkner

Schule: Europaschule Schulzentrum Utbremen

Ausbildungsberuf: Fachinformatiker für Anwendungsentwicklung

Ausbildungsjahr: 2

Zeitraum: Februar 2015 – Mai 2015

Inhaltsverzeichnis

Inhaltsverzeichnis.....	2
1. Vorwort	4
2. Projektdefinition.....	5
2.1. Ausgangssituation.....	5
2.2. Ideensammlung / Brainstorming	5
2.3. Firmengründung	6
2.4. Namensfindung.....	7
2.5. Entscheidungsfindung.....	7
3. Projektplanung	8
3.1. Detaillierung	8
3.2. Strukturierung.....	10
3.3. Verantwortlichkeiten	12
4. Marketingkonzepte	13
4.1. Marktanalyse & Verfassung des aktuellen Marktes.....	13
4.2. Marketingziele	13
4.3. Marketingstrategien.....	13
4.4. Marketing-Mix	14
5. Projektdurchführung.....	15
5.1. Erstellen der Datenbankstruktur und der Datenbank	15
5.2. Programmierung des Webservice.....	16
5.3. Rechtesystem.....	17
5.4. Programmierung der Benutzeroberfläche.....	18
5.5. Programmierung der Produktfunktion	19
6. Qualitätssicherung	22
6.1. Erstellung der Testszenarien.....	22
6.2. Testdurchführung	22
6.3. Bugfixing	22
6.4. Codereview.....	23
6.5. Datenschutz.....	23
7. Fazit	24

7.1. Soll- / Ist-Vergleich.....	24
7.2. Problemdarstellung.....	24
7.3. Zukünftige Entwicklung und weitere Vorgehensweise.....	25
8. Anhang	27

1. Vorwort

Im Rahmen unseres zweiten Ausbildungsjahres führen wir ein Mittelstufen (Gruppen-)Projekt in den Fächern Anwendungsentwicklung und Geschäftsprozesse durch. Mit selbst gewählten Zielen, sowie Technologien versuchten wir unsere Projektidee umzusetzen. Diesen Prozess haben wir im Folgenden festgehalten. Wir gehen dabei zunächst auf die Ausgangssituation ein, kommen dann von der Planung, über die Entwicklung zum Endprodukt, und möchten auch Aufschluss darüber geben welche Entscheidungen wir gefällt haben, welche sich als ideal herausstellten sowie – im Nachhinein betrachtet – welche in Zukunft anders ausfallen würden.

2. Projektdefinition

2.1. Ausgangssituation

Durch unsere verschiedenen, technischen Kenntnisstände waren wir in unserer Wahl sehr eingeschränkt. Dies ist der Grund weshalb wir versuchten eine Lösung zu finden bei der möglichst alle Teammitglieder die Technik bereits erlernt haben. Denn nur so kann man eine gleichmäßige und gerechte Aufteilung der Aufgaben durchführen.

- Mike: COBOL, HTML, CSS, .net, (Java)
- Dennis: SQL, JavaScript, HTML, CSS, C#, (Java), (Java EE)
- Sebastian: SQL, JavaScript, HTML, CSS, Objective-C, (Java), (Java EE)
- Lukas: SQL, JavaScript, HTML, CSS, (Java), (Java EE), PHP

Wie man an der Auflistung sehen kann, liegt unsere größte Übereinstimmung bei HTML und CSS. Dazu haben wir uns entschieden diese Funktionalität mit JavaScript zu erweitern. Dem entsprechend stand fest: Wir entwickeln eine Website / Webapp!

Da ein Großteil des Teams bereits Erfahrung mit dem NodeJS Framework gesammelt hat, haben wir uns dazu entschlossen auf diese Erfahrung aufzubauen. So können auch die Mitglieder, die wenig bis gar keine Vorkenntnisse haben, relativ effizient und zeitnah an das Framework herangeführt werden.

2.2. Ideensammlung / Brainstorming

Unsere Ideensammlung haben wir bei einem gemeinsamen Brainstorming zusammengetragen und stetig ergänzt, sowie detaillierter ausgearbeitet. Einige dieser Ideen lauteten wie folgt:

- Lagerverwaltung
- Zeiterfassung

- Veranstaltungsplanung
- Raumverwaltung
- Empfehlungsplattform
- Spieleplattform
 - Anzeige und spielen von JavaScript basierten Spielen in einem Canvas Element
 - Benutzer können eigene Spiele entwerfen und anlegen
 - Spielbaukasten
- Asset Marketplace
 - Ähnliche Projekte bereits auf dem Markt (z.B. „RPG-Maker“, jedoch noch keine Framework unabhängige Lösung)
 - Von jedem bedienbar -> große Zielgruppe
 - Mögliches Problem: zeitlicher Rahmen nicht ausreichend

Jede dieser Ideen kann mit Hilfe den zuvor ausgewählten Technologien und Frameworks umgesetzt und als Website angesteuert werden.

Bei Recherchen zu dem Angebot von den oben genannten Ideen fiel uns auf, dass die Spieleplattform – so wie wir sie uns vorstellten – eine Marktinnoation darstellte. Wir beschlossen die Idee der Spieleplattform weiter auszubauen und mehr Informationen sowie Vorschläge zu suchen.

2.3.Firmengründung

Bei der Bestimmung der Firmenart stellte sich die Frage wie genau unser Unternehmen aufgebaut werden sollte. Wir sind vier gleichberechtigte Gründungsmitglieder, was sich

langfristig nicht ändern sollte. Außerdem sollte das Unternehmen ohne eine persönliche Haftung unsererseits auskommen.

Dies spiegelt sich am ehesten in der Gründung einer GmbH wieder. Einerseits ist die Haftung gedeckelt, andererseits kann hierbei der Gesellschaftsvertrag nach unseren Wünschen gestaltet werden.

2.4.Namensfindung

Auf der Suche nach einem passenden Namen für unser Projekt, war der Bezug zum Thema 'Online Spiele' für uns von elementarer Bedeutung. Beim Brainstorming sind wir zu Titeln wie „Canvas24“ und „JSGames“ gekommen. Wir wollten jedoch eine Bezeichnung finden die den potenziellen Anwender ein wenig mehr mit einschließt und anspricht. Außerdem wollten wir klar machen, dass es bei unserem Portal auch persönliche Interessen und „Herzblut“ mit eingeflossen sind.

Des Weiteren sollte eine möglichst große und junge Zielgruppe angesprochen werden, wobei ein einprägsamer, englischer Name hilfreich ist. Ein weiterer Vorteil eines englischen Namens ist, dass er einen eventuellen globalen Start vereinfacht. So haben wir uns letztendlich für den Namen „**We Love Games**“ entschieden.

2.5.Entscheidungsfindung

Unser erklärtes Ziel lautet dem Entsprechend eine Spieleplattform in Form einer Website zu entwickeln, auf der Kunden Spiele hochladen, erstellen und spielen können, ohne dabei besondere Kenntnisse, oder Hardware zu besitzen. Jeder soll in der Lage sein, sich auf der Website zurechtzufinden sowie sie zu bedienen. Die Umsetzung wird mit dem Framework NodeJS realisiert.

3. Projektplanung

3.1. Detaillierung

Die Umsetzung des Projektes werden wir mit Hilfe von verschiedenen Frameworks realisieren. Wie schon erwähnt, haben einige Teammitglieder bereits Erfahrungen mit dem NodeJS¹ Framework gesammelt. NodeJS eignet sich besonders gut, schnell und ohne besonderen Aufwand einen Webservice aufzusetzen. Man schreibt den Quellcode in der Programmiersprache JavaScript², welcher beim Starten von Googles JavaScript Engine V8³ interpretiert wird. Die Vorteile von NodeJS liegen auf der Hand:

- Tasks werden asynchron verarbeitet und bieten so eine deutlich bessere Performance gegenüber Konkurrenten in diesem Segment (wie z.B. PHP)
- Die Community von NodeJS ist sehr agil und bietet mit dem Packet Manager NPM⁴ eine unvorstellbare Menge an Paketen und Frameworks, die für jeden möglichen Anwendungsfall eine Lösung anbieten.
- Die Serveranwendung ist plattformunabhängig, sowie Open Source und auf so gut wie jedem Rechner lauffähig – egal ob professioneller Server oder Microcomputer (wie zum Beispiel einem „raspberry pi“).

Des Weiteren nutzen wir eine MySQL Datenbank zum Speichern von Daten, da wir auch hier ein gewisses Maß an Vorwissen nutzen konnten um effizient zu Arbeiten. Auch hier ist ein großer Vorteil, dass die Datenbank plattformunabhängig sowie Open Source ist.

In Zeiten von immer verschiedenere Endgeräten ist es für eine Website sehr wichtig ‚responsive‘ zu sein. Das heißt die Seite muss sich den verschiedensten Bildschirmauflösungen anpassen können und sorgt so auch auf Smartphones für eine gut

¹ <https://nodejs.org> (20.05.2015)

² <http://de.wikipedia.org/wiki/JavaScript> (20.05.2015)

³ <https://code.google.com/p/v8/> (20.05.2015)

⁴ <https://www.npmjs.com> (20.05.2015)

lesbare Darstellung. Aus diesem Grund nutzen wir das Frontend-Framework Bootstrap⁵ - ursprünglich von Twitter entwickelt, nun als Open Source Framework erhältlich – als Hilfstemplate.

Zu diesen Technologien nutzen wir einige Module aus dem oben genannten Paket Manager. Ein paar Beispiele:

- **Jade Template Engine**⁶: Jade ist eine Template Engine die HTML-Code ausgibt. Sie gibt uns die Möglichkeit unsere Templates generisch anzupassen und so zum Beispiel jeweils die richtigen Anzeigebilder oder Spieleparameter, wie den Punktestand einzubinden.
- **Passport**⁷: Da wir eine Nutzerverwaltung benötigen um kontrollieren zu können wer, auf welche Funktionen zurückgreifen darf haben wir uns für das Passport Modul entschieden welches sich als Middleware nutzen lässt. Das gibt uns die Möglichkeit die Authentifizierungsfunktion bei jeder Anfrage aufzurufen, bevor Datenbank oder sonstige Daten geladen werden. Man kann außerdem auch die Konten von Facebook oder Google zur Authentifikation nutzen, wir haben uns jedoch dazu entschlossen unsere eigene Nutzerverwaltung zu implementieren.
- **Bcrypt-NodeJS**: Dieses Modul regelt zusammen mit Passport die Authentifizierung. Wir haben uns bei der Verschlüsselung der Nutzerpasswörter für den Bcrypt-Algorithmus⁸ entschieden, um für eine verstärkte Sicherheit der Nutzerdaten zu sorgen. Bei jedem Login wird aus dem vom User eingegebenen Passwort ein Hash generiert und mit dem in der Datenbank gespeicherten Hash verglichen. Bcrypt bietet den großen Vorteil gegenüber anderen Hash Algorithmen, dass Entschlüsselungsdauer vergleichsweise hoch und damit die Rechenleistung relativ hoch ausfällt. Dem entsprechend ist ein ‚knacken‘ der Passwörter selbst mit professioneller Hardware *deutlich* (zeit-)aufwändiger. Beim täglichen Einsatz ist diese Tatsache jedoch so gut wie gar nicht zu spüren.

⁵ <http://getbootstrap.com> (20.05.2015)

⁶ <http://jade-lang.com> (20.05.2015)

⁷ <http://passportjs.org/docs/> (20.05.2015)

⁸ <http://de.wikipedia.org/wiki/Bcrypt> (20.05.2015)

- **Nodemailer**⁹: Zur ersten Anmeldung benötigt der Nutzer eine gültige Emailadresse. Zur Bestätigung dieser versenden wir eine Bestätigungsemail mit einem Aktivierungslink an die vom Nutzer angegebene Mailadresse. Hier erlaubt uns das Nodemailer Modul eine einfache Möglichkeit hübsche HTML-E-mails zu versenden.

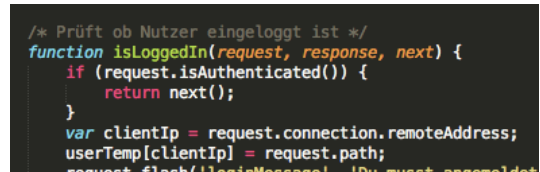
3.2. Strukturierung

Um einen möglichst einheitlichen Code-Stil, trotz mehrerer Programmierer zu entwickeln, haben wir eine Naming-& Code-Convention erstellt, die von allen Programmierern eingehalten werden sollte.

Einige dieser Konventionen lauten wie folgt:

- CamelCase: Variablen und Funktionen werden nach der CamelCase-Schreibweise benannt. Damit wird die allgemeine Lesbarkeit erhöht. Anbei in Abbildung 1 eine Funktion und darin enthaltene Variablen mit CamelCase.

- Englische Namen: Variablen und Funktionen sollten einheitlich englische Bezeichnungen haben.



```
/* Prüft ob Nutzer eingeloggt ist */  
function isLoggedIn(request, response, next) {  
  if (request.isAuthenticated()) {  
    return next();  
  }  
  var clientId = request.connection.remoteAddress;  
  userTemp[clientId] = request.path;  
  request.flash('loginMessage', 'Du musst angemeldet
```

Abbildung 1: Funktion zur Überprüfung des Users, mit genannten Konventionen

- LC-First: LowerCase-First legt fest, dass der erste Buchstabe von Variablen und Funktionen klein geschrieben wird.
- Auslagerung von Einstellungen: Um es möglichst einfach zu gestalten, den Webservice auf verschiedenen Systemen starten zu können bzw. Datenbanken auszutauschen, sollten Einstellungen wie Port, Datenbankname, Tabellennamen, Datenbankuser und Passwort separat gespeichert und eingebunden werden. Dies hat

⁹ <http://adilapapaya.com/docs/nodemailer/> (20.05.2015)

den Vorteil, dass Einstellungen bei eventuellen Änderungen nur einmal angepasst werden müssen.

```
// config/database.js
module.exports = {
  'connection': {
    'host': 'localhost',
    'user': 'root',
    'password': '*****'
  },
  'database': 'browsegame_node',
  'tableUsers': 'users',
  'tableGames': 'games',
  'tableHighscores': 'highscores'
};
```

Abbildung 2: Beispiel der Auslagerung
anhand unserer Datenbankeinstellungen

Die Ordner- und Dateistruktur ist für die Übersichtlichkeit ebenso wichtig wie das Halten an Namingconventions. Bei der Wahl dieser Struktur haben wir uns an bestehenden NodeJS Projekten – aus unseren Unternehmen – sowie aus Internetquellen bedient und nutzen eine Funktion von NodeJS namens Modules¹⁰. Diese ermöglicht das Erfüllen eines der fundamentalen Gebote der objektorientierten Programmierung, der Kapselung. Funktionen und Variablen sind untereinander nur sichtbar, wenn dies von der Funktionalität gefordert wird. Dies ist einerseits übersichtlicher, andererseits sorgt es für eine höhere Sicherheit des Systems. Eine Abbildung der Datei und Ordnerstruktur ist im Anhang zu finden.

Technologie ist im stetigen Wandel. So auch Datenbanken und deren Software. Um auf alle zukünftigen Eventualitäten vorbereitet zu sein und uns nicht dauerhaft an die jetzt gewählte Datenbank MySQL zu binden haben wir uns dazu entschlossen das Design Pattern Data Access Object (DAO)¹¹ zu implementieren. Durch diese Entscheidung ist der Aufwand, den ein Wechsel – egal ob gezwungener Maßen oder aus anderen Gründen – verursachen würde, minimiert. Es müsste lediglich ein neues Data Access Object implementiert und eingebunden werden. Unsere Software ist so konzipiert, dass nur an einer einzigen Stelle das alte DAO durch das neue ersetzt werden müsste.

¹⁰ <https://nodejs.org/api/modules.html> (20.05.2015)

¹¹ http://de.wikipedia.org/wiki/Data_Access_Object (20.05.2015)

3.3. Verantwortlichkeiten

Da wir das Versionskontrollsystem „Git“ benutzen ist jeder für seinen eigenen Code verantwortlich. Wer etwas verfasst, geändert oder gelöscht hat, kann dank Git ohne große Umwege nachvollzogen werden. Außerdem ist es sehr einfach einen zuvor gemachten Fehler durch einen so genannten ‚Revert‘ im Git rückgängig zu machen. Zudem haben wir die Möglichkeit mit so genannten ‚Branches‘ einzelne Arbeitsschritte in Zweigen zu unterteilen und verschiedene Versionen getrennt voneinander zu bearbeiten.

Zu diesem Sicherheitsaspekt haben wir eine enge Zusammenarbeit angestrebt, damit möglichst jedes Teammitglied einen Rundumblick vom Projekt erhält. Neben fast täglichen Besprechungen, wer gerade was über welchen Zeitraum erledigt, haben wir uns in regelmäßigen Abständen zusammengefunden und über die weitere Planung und den allgemeinen Zustand des Projektes beraten.

Des Weiteren haben wir Lukas Trenkner zum Projektleiter ernannt.

4. Marketingkonzepte

4.1. Marktanalyse & Verfassung des aktuellen Marktes

Bevor wir unser Projekt gewählt haben, haben wir uns – wie schon erwähnt – auf dem aktuellen Markt umgesehen. Bei vielen unserer Ideen gibt es bereits einen übersättigten Markt (z.B. die Zeiterfassung). Bei dem Baukasten für JavaScript Spiele haben wir nichts Vergleichbares gefunden. Die einzigen fertigen Lösungen in diesem Bereich setzten ein Grundwissen von Programmierung voraus, um sie zu bedienen. Somit erschließen wir mit unserem Projekt einen neuen Markt, da wir auch unerfahrenen Nutzern die Möglichkeit geben kreativ zu sein.

4.2. Marketingziele

Jeder Mensch spielt Spiele. Egal ob das Computer- oder Konsolenspiel, ein kleines Minispiel auf dem Smartphone, ein Kreuzworträtsel in der Zeitung oder ganz klassisch der Spieleabend mit der Familie oder Freunden an dem die bekannten Brettspiele aus dem Regal genommen werden. Jeder spielt in verschiedenen Situationen Spiele. Wir haben das Ziel eine neue Option in diesem vielfältigen Angebot zu sein. Wir wollen den oben genannten neuen Markt zunächst einmal erschließen, gerade durch unser Alleinstellungsmerkmal, der Interaktion. Jeder kann sich selbst eine Spieleidee überlegen und es mit Hilfe unserer Webseite ganz einfach global spielbar machen. Die Marktetablierung spielt dabei eine große Rolle, aber natürlich müssen wir mittel- bis langfristig auch an unsere Plattform als Marke denken und diese möglichst bekannt machen.

4.3. Marketingstrategien

Dem vorausgehend sind unsere Marketingstrategien klar verteilt. Wir setzten auf Wachstum unserer Nutzerzahlen und erreichen dies durch Differenzierung von unseren Konkurrenten. Unsere Entwicklung ist ein weiterer wichtiger Punkt, denn die ständige Anpassung an den

aktuellen Markt, beziehungsweise die Bedürfnisse unserer Kunden bedarf einer nie endenden Innovationsspirale. So ist die Weiterentwicklung hin zu mobilen Lösungen der nächste, mögliche Schritt unseres Unternehmens.

4.4. Marketing-Mix

Die Marketingstrategien lassen sich nur umsetzen, wenn wir all unsere Möglichkeiten in Marketinginstrumenten voll ausschöpfen. Dazu ist, gerade bei unserer potenziell jungen Zielgruppe, eine auffällige Kommunikationspolitik von Nöten. So werden wir uns zunächst auf allen gängigen sozialen Netzwerken (u.a. Facebook, Twitter, Google+, xing) etablieren um so neue Nutzer zu gewinnen.

Wir sind von unserem Produkt überzeugt und zuversichtlich, dass auch unsere Kunden davon begeistert sein werden. Unsere Plattform ist sicher, und läuft stabil da wir ab der ersten Zeile Code auf Qualität und Sicherheit setzten. Nun liegt es an unseren Kunden kreativ zu sein und sich somit – zu den bereits bestehenden – weitere Spiele auszudenken und somit eine große Produkttiefe zu entwickeln.

Des Weiteren wollen wir uns als Marke etablieren. Erstellt ein Kunde ein Spiel, so hat er die Möglichkeit das Spiel zu verlinken (beispielsweise bei Facebook zu teilen), sowie sich das Spiel herunterzuladen, um es zum Beispiel auf seiner eignen Website hinzuzufügen. Bei heruntergeladenen Spielen wird ein Wasserzeichen mit unserem Namen, sowie eine Verlinkung, auf unsere eigene Website in das Spiel eingebaut, um somit noch mehr Nutzer erreichen zu können.

Bei unserer Preispolitik setzen wir auf eine kostenlose Website, die jedem zur Verfügung steht. Unsere Finanzierung wird mit Hilfe von unspezifischer Werbung, die an verschiedenen Stellen der Website geschaltet wird, umgesetzt. Je mehr Nutzer wir für uns gewinnen, und je öfter sie auf unserer Plattform surfen, desto mehr Werbeeinnahmen könne generiert werden.

5. Projektdurchführung

5.1. Erstellen der Datenbankstruktur und der Datenbank

Beim Entwickeln der Datenbankstruktur wollten wir sie der dritten Normalform konform erstellen. Dem entsprechend sind pro Datenbanktabelle keine redundanten Felder vorhanden, beziehungsweise werden diese durch Fremdschlüssel aufgelöst.

Wir nutzen drei Datenbanktabellen:

1. Benutzertabelle:

Für unsere Nutzerverwaltung ist es essenziell die Anmeldedaten und weitere Einstellungen zu speichern. Primärschlüssel ist klassisch die ID, dazu kommen Benutzernamen, Emailadresse, Passworthash und zwei boolesche Werte ‚inactive‘, welcher angibt ob der Nutzer seine Emailadresse schon bestätigt hat, und ‚isAdmin‘, der eine Unterscheidung zwischen normalen Nutzern und Administratoren zulässt.

users	
PK	ID int(11)
	username varchar(45)
	email varchar(100)
	password varchar(200)
	inactive bool
	isAdmin bool

2. Spieletabelle:

Zur Speicherung von Spielen nutzen wir zwei Speichermedien. Einerseits eine Datenbanktabelle, andererseits eine Filesystemspeicherung auf der Festplatte des Servers. Die Spieletabelle hat auch einen eigenen Primärschlüssel, dazu den Spielnamen, die Spielbeschreibung, den Uploader als Fremdschlüssel, ein ‚imageEnc‘, welches definiert, welches Encoding (mit dem Entsprechender Dateiendung) für das Anzeigebild genutzt wird, ein Feld ‚javascript‘, über das eigens hochgeladene JavaScript Dateien hochgeladen werden können und ein Boolean ‚inactive‘ welcher das Spiel deaktivierbar macht.

games	
PK	ID int(11)
	gamename varchar(45)
	description varchar(100)
FK	user int(11)
	inactive bool
	imageEnc varchar(4)
	javascript varchar(100)

3. Highscoretabelle:

Um den Spielen einen sozialen Touch zu geben und die Nutzer dazu zu animieren oft zu Spielen haben wir ein Highscoresystem eingebaut. Dieses wird durch eine Weitere Tabelle in der Datenbank realisiert. Auch hier gibt es einen Primärschlüssel in Form einer ID, die Fremdschlüssel Spiel und Nutzer, den Score selbst und einen Zeitstempel.

highscores	
PK	ID int(11)
FK	user int(11)
FK	game int(11)
	socre int(11)
	tStamp DATETIME

Die Datenbank, genauer gesagt das Datenbankschema ist so konzipiert, dass beim Start der Anwendung überprüft wird, ob sich ein Schema mit richtigem Namen auf dem Rechner befindet, auf dem die Anwendung gestartet wird (localhost). Ist ein Schema mit allen nötigen Tabellen vorhanden, wird dies genutzt. Fehlt das komplette Schema, eine oder mehrere Tabellen, so kann die Anwendung die komplette oder nur Teile der Datenbankstruktur vollautomatisch erstellen. Ein Migrieren sowie Testen der Anwendung auf einem anderen System ist somit sehr einfach. Es muss lediglich sichergestellt sein, dass MySQL auf dem Localhost installiert und der Service gestartet ist (und die richtigen Benutzereinstellungen in der Anwendung stehen). Dies wird mit der MySQL Funktion „CREATE IF NOT EXIST“¹² gelöst.

5.2. Programmierung des Webservice

Der Webservice ist die wichtigste Entwicklung unseres Projektes. Er steht in Verbindung mit der Datenbank, empfängt und beantwortet Anfragen von Nutzer. Trotzdem ist die Entwicklung mit Hilfe des Node Frameworks denkbar einfach. Bereits wenige Zeilen Code reichen aus um einen startfähigen Server zu programmieren.

¹² <https://dev.mysql.com/doc/refman/5.5/en/replication-features-create-if-not-exists.html>
(20.05.2015)

Die folgenden sechs Zeilen (Abbildung 3) sorgen dafür, dass der gestartete Server auf eine http-get-Anfrage, auf seiner IP-Adresse mit dem angegebenen Port mit „Hello World“ antwortet:

```
1 var http = require('http');
2 http.createServer(function (req, res) {
3   res.writeHead(200, {'Content-Type': 'text/plain'});
4   res.end('Hello World\n');
5 }).listen(1337, '127.0.0.1');
6 console.log('Server running at http://127.0.0.1:1337/');
```

Abbildung 3: Einfache Gestaltung eines Webservices mit NodeJS

Genau genommen ist unser Webservice – in der Datei ‚server.js‘ – bis zum Projektabschluss nicht länger als 65 Zeilen geworden. Natürlich ist das auch der Tatsache zu verdanken, dass wir sehr vieles auslagerten.

Um den Server zu starten muss man das Terminal / die Komandozeile öffnen, zum Projekt navigieren und den Befehl „\$ node server.js“ ausführen. Die Anwendung wurde ohne Fehler gestartet, wenn daraufhin diese Ausgabe erscheint:



Abbildung 4: Betriebsbereiter Webservice im Terminal

Ab diesem Zeitpunkt ist der Server bereit Anfragen von Nutzern entgegenzunehmen und zu beantworten.

5.3.Rechtesystem

Bei der Entwicklung unserer Spieleplattform haben wir drei verschiedene Benutzergruppen vorgesehen:

- Gastbenutzer: Jeder Nutzer, der die Website zum ersten Mal aufruft wird als Gast eingestuft. Er hat damit die Möglichkeit aktive Spiele zu sehen und zu Spielen.

Außerdem kann er sich mit Hilfe der Anmeldemaske ein neues Benutzerkonto erstellen, welches er per Klick auf den ihm zugesendeten Link bestätigen muss.

- Angemeldeter Nutzer: Meldet sich ein Nutzer über den Login an, so kann er zu den Funktionen des Gastes noch eigene Spiele erstellen und hochladen, seine eigenen Spiele verwalten, also deaktivieren und aktivieren, sowie seine Benutzereinstellungen, wie Passwort und Email, ändern. Außerdem kann er beim Spielen Highscores erzielen und nimmt somit an einem globalen Wettbewerb teil. Des Weiteren hat er die Möglichkeit seine eigenen Spiele herunterzuladen.
- Administratoren: Dieser Nutzer hat alle Rechte vom angemeldeten Nutzer und noch zusätzliche Funktionen. Er kann alle Spiele verwalten, also deaktivieren und aktivieren, alle Benutzer bearbeiten, anderen Nutzern Adminrechte geben und entziehen sowie neue Nutzer hinzufügen ohne einen Bestätigungsprozess zu vollziehen.

5.4. Programmierung der Benutzeroberfläche

Die GUI, also das **grafische User Interface**, muss unserer Zielgruppe entsprechend nicht nur für erfahrene Computerprofis, sondern auch für Normalnutzer bedienbar sein. Die Vereinfachung von Funktionen war aus diesem Grund die primäre Aufgabe.

Durch die verwendete Jade Templating Engine wurde auch das Schreiben der Templates vereinfacht. Die Schreibweise von Jade macht das Erstellen von HTML Seiten deutlich angenehmer. So kann man sich die spitzen Klammern sowie die End-Tags beim Schreiben sparen, da Jade die Einrückung zum Erstellen des DOM¹³-Baums verwendet. Links ist der

¹³ Document Object Model siehe dazu: http://de.wikipedia.org/wiki/Document_Object_Model (20.05.2015)

geschriebene Jade Code zu sehen, in der Mitte der daraus entstandene DOM-Baum, und rechts die fertige Anzeige im Webbrowser (Abbildung 5).



Abbildung 5: Vom Jade Code, über HTML, hin zur fertigen Darstellung im Browser

5.5. Programmierung der Produktfunktion

Die Hauptfunktion unserer Plattform ist natürlich das Spielen und Erstellen von JavaScript Spielen. Beim Erstellen bieten wir dem Nutzer zwei verschiedene Optionen. Die erste Option ist die einfachste, Nutzer können über ein Formular mit Vorschauansicht ihr eigenes Spiel zusammenklicken. Somit kann jeder seine Spieleidee mit wenigen Mausklicks umsetzen. Die zweite Option richtet sich an erfahrenere Nutzer. Sie erlaubt es fertige Spiele mit Hilfe eines Uploads hochzuladen. Dazu müssen einige Grundvoraussetzungen erfüllt werden, damit das Spiel auf unserer Seite spielbar ist:

- Es muss ein Canvas-Element für die Zeichnung des Spiels benutzt werden.
- Alle Ressourcen, die im Spiel benutzt werden, wie zum Beispiel Bilder, müssen als Pfad „path + den eigentlichen Pfad“ nutzen, da der Pfad erst nach speichern in der Datenbank zur Verfügung steht.

- Um den Highscore abzuschicken sollte ein XMLHttpRequest¹⁴ benutzt werden. Dieser sendet einen POST-Request, mit Score und Spielinformation, an den Server, welcher dann die Anfrage bearbeitet, Rechte des Nutzers überprüft und den Highscore dann abspeichert.
- Beim Hochladen muss darauf geachtet werden, sowohl die JavaScript Datei, als auch alle nötigen Ressourcen abzuschicken

Da wir unsere Plattform möglichst allen zugänglich machen wollten haben wir ein großes Augenmerk auf die Browserkompatibilität gesetzt. Wir verwenden das HTML 5 Element Canvas, welches von allen gängigen Browsern unterstützt wird¹⁵. Dazu kommt eine JavaScript Funktion mit der Bezeichnung „Window.requestAnimationFrame()“. Diese Funktion sorgt dafür, dass, je nach Aktivität, mit sich ändernden Werten, in gewissen Abständen das Canvas neu gezeichnet wird. So entsteht der Eindruck man sehe einen flüssigen Bewegungsablauf. Leider gehört diese Funktion noch nicht sehr lange zum Standard, weshalb nur neuere Versionen der Browser diese unterstützen. Jedoch haben die Browserhersteller auch vor der Standardisierung diese Funktion in ihre jeweiligen Interpreter aufgenommen, wodurch wir eine mit Hilfe des folgenden Aufrufes (Abbildung 6) deutlich mehr Browser nutzen können:

```
var requestAnimationFrame = window.requestAnimationFrame ||  
                             window.mozRequestAnimationFrame ||  
                             window.webkitRequestAnimationFrame ||  
                             window.msRequestAnimationFrame;
```

Abbildung 6: Browserkompatibilität durch generischen Aufruf

Beim Anmelden von Nutzern überprüft der Webservice zudem, ob ein Benutzerordner vorhanden ist. Wenn nicht wird ein neuer Ordner angelegt um ein Erstellen von Spielen zu ermöglichen. Wir haben uns folgende Ordnerstruktur für das Speichern von Spielerressourcen überlegt:

¹⁴ <https://developer.mozilla.org/de/docs/Web/API/XMLHttpRequest> (20.05.2015)

¹⁵ siehe hierzu: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/canvas> (20.05.2015)

- Jeder Nutzer hat einen eigenen Benutzerordner, der mit Hilfe seiner ID bezeichnet wird.
- Lädt ein Nutzer ein Spiel hoch, so werden alle dazugehörigen Ressourcen in einem Spieleordner innerhalb des Nutzerordners unter der Bezeichnung ID.Spielenamen abgespeichert.

Bei der Entwicklung haben wir uns gegen ein konkretes Löschen von Nutzern und Spielen entschieden, um Problematiken, wie zum Beispiel Datenverlust beim versehentlichen Löschen, zu Umgehen. Trotzdem besteht die Möglichkeit Nutzer und Spiele zu deaktivieren und sie so von der Plattform auszuschließen. Das Deaktivieren kann jedoch jederzeit (von Administratoren) rückgängig gemacht werden.

6. Qualitätssicherung

Neben der eigentlichen Entwicklung ist die Qualitätssicherung von großer Bedeutung für unser Projekt. Das Testen wurde in regelmäßigen Abständen durchgeführt um jederzeit zu wissen, auf welchem Stand sich das Projekt befindet, und vor allem wo noch Arbeit zu leisten ist.

6.1. Erstellung der Testszenarien

Dazu haben wir über dreißig Testszenarien entwickelt, die alle Funktionen der Website, sowie des Webservices abdecken (siehe Anhang).

6.2. Testdurchführung

Zur Durchführung der Tests wurde der Webservice mit intakter Datenbank gestartet. Dazu wurden verschiedene Browser, aktueller Versionen, gestartet und die Testfälle nach und nach ausgeführt.

Es wurde ebenfalls getestet wie sich der Webservice ohne Datenbankschema verhält, und ob er autonom Schema und Tabellen korrekt erstellen kann.

Besonders intensiv wurde unser Rechtesystem getestet um Missbrauch zu verhindern.

6.3. Bugfixing

Beim Entwickeln von Softwareprodukten bleibt das Auftreten von so genannten Bugs nicht aus. Damit sind Fehler in der Software gemeint, bei denen unerwartete Aktionen auftreten oder Produktfunktionen nicht erfüllt werden. Durch testen und viele Besprechungen wurden viele Bugs definiert und danach gelöst

6.4.Codereview

Die Arbeit von uns Entwicklern muss im Idealfall unter ständiger Kontrolle stehen um eine gleichbleibend gute Qualität des Quellcodes zu gewährleisten. Unser Versionskontrollsystem „Git“ hat dazu beigetragen, dass jedes Teammitglied in regelmäßigen Abständen über die Änderungen von anderen informiert wurde. Dem entsprechend waren klassische Code Reviews, also eine dedizierte Besprechung, bei der nur der Code Stil betrachtet und bewertet wird, nicht zwingend notwendig, was auch daran liegt, dass sich jeder an die Code Convention gehalten hat.

6.5.Datenschutz

Wir nehmen Datenschutz sehr Ernst. Aus diesem Grund haben wir uns für die Implementierung des Rechtesystems entschieden. Nur berechtigte Nutzer sollten auf Personenbezogene Daten zugreifen, diese Ändern und Erstellen dürfen. Des Weiteren haben wir uns zum Schutz unserer Nutzerdatenbanktabelle für den Hash- beziehungsweise Verschlüsselungs-Algorithmus Bcrypt entschieden. Selbst in dem unwahrscheinlichen Fall, dass diese Tabelle in den Besitz Dritter gelangt ist es sehr schwer, die Passwörter daraus zu generieren.

Auch interne Angriffe auf die Datenbank mit Hilfe von so genannten SQL-Injektionen haben wir durch Kapselung unterbunden.

7. Fazit

7.1. Soll- / Ist-Vergleich

Das Ziel unseres Mittelstufenprojektes war die Entwicklung einer Spieleplattform, die sowohl von Menschen mit Entwicklungserfahrung, wie auch solchen die keinerlei Kenntnisse von Softwareprogrammierung besitzen, bedienbar ist. Beide Nutzergruppen sollten in der Lage sein kreativ zu werden und zusammen eine vielseitige Plattform möglich zu machen.

Im Allgemeinen wurden unsere Erwartungen und Vorstellungen erfüllt. Die Plattform ist betriebsbereit und so bedienfreundlich wie wir es uns erhofften.

7.2. Problemdarstellung

Nichtsdestotrotz ist es auch zu Problemen und Komplikationen gekommen, die vorher nicht Bedacht beziehungsweise nicht mit eingeplant wurden.

- Die Infrastruktur der Schule erlaubt es weder neue Software auf den Rechnern zu installieren, noch eigene Betriebssysteme (zum Beispiel von einem USB Stick) zu Nutzen. Da wir von der NodeJS-Installation abhängig waren mussten wir auf eigene Rechner zurückgreifen. Nicht allen steht ein portables Laptop zur Verfügung, weshalb wir diese Schwierigkeit mit einem Laptop aus Firmenbesitz sowie eines ausrangierten Laptops eines Projektmitgliedes lösten.
- Die kabellose Internetverbindung der Schule weist *extreme* Schwankungen in der Performance auf. So war es teilweise nicht möglich eine einfache Google-Suche zu vollziehen, was die Entwicklungsarbeit deutlich erschwerte. Des Weiteren ist es nicht ersichtlich, warum der Zugang zu privaten „Git-Repositories“ sowie dem Packetmanager NPM blockiert wird. Auch deshalb mussten wir viele Arbeitsstunden, die wir in der Schule nicht produktiv Nutzen konnten auf den Nachmittag

beziehungsweise Abend – teilweise sogar in den Betrieb – verlegen, was an sich nicht im Sinne der Projektanforderung sein kann.

- Der Upload von Spielerressourcen stellte sich als deutlich umfangreicher heraus, als wir einplanten. Dies lässt sich unter anderem durch fehlendes Know-How unsererseits erklären. Durch die Hilfe von Herrn Heidmann gelang es uns jedoch den Upload (mit Einschränkungen) fertigzustellen.
- Wie bei vielen Softwareprojekten bestand die Gefahr des Paretoprinzips¹⁶. Dies besagt, dass 80% des Projektes in nur 20% der Zeit erledigt werden, aber die restlichen 20% des Projektes ganze 80% der Zeit beanspruchen. Dies ist vor allem bei voneinander unabhängiger Entwicklung zu beobachten. Obwohl wir in ständigem Austausch miteinander standen, konnten wir eine ähnliche Verteilung unserer Arbeitsressourcen beobachten. So waren wir mit dem Grundgerüst der Plattform bereits nach einigen wenigen Wochen fertig, benötigten jedoch sehr viel Zeit und Arbeit um die Detailarbeit zu vollenden, weshalb die Zeitplanung beim Fortschreiten des Projektes zunehmend unter Druck geriet.

7.3. Zukünftige Entwicklung und weitere Vorgehensweise

Die Entwicklung einer Software im allgemeinen, in unserem Fall einer Website, darf nicht zum Stillstand kommen. Der Markt verändert sich stetig und damit auch die Erwartungen und Wünsche der Nutzer. Überdies ist die Ausbildung einer mobil bedienbaren Plattform eine mögliche Weiterentwicklung, die einen essenziellen Mehrwert bieten würde. Ferner bietet die Auswahl an Spieleoptionen eine gute Möglichkeit die User-Experience weiter zu verbessern.

¹⁶ <http://de.m.wikipedia.org/wiki/Paretoprinzip> (20.05.2015)

Doch auch ohne die aktive Weiterentwicklung ist eine laufende Supportarbeit nötig, um möglicherweise auftretende „Bugs“ und Fehlfunktionen im laufenden Betrieb zu erkennen sowie möglichst zeitnah zu beheben.

Wir möchten an dieser Stelle darauf aufmerksam machen, dass die Zusammenarbeit untereinander und der Projektverlauf – trotz kleinerer Schwierigkeiten – gut geglückt sind. Auch das Format einer autonomen Gruppenarbeit sehen wir als gute Übung und Grundlage für die Abschlussprüfung, sowie unseren zukünftigen Ausbildungsberuf.

8. Anhang

Um die Installation von allen Modulen möglichst einfach zu gestalten haben wir eine für Node typische Package.json¹⁷-Datei angelegt.

```
{
  "name": "weLoveGames",
  "description": "weLoveGames is super awesome!",
  "private": true,
  "version": "0.1.1",
  "dependencies": {
    "bcrypt-nodejs": "0.x",
    "body-parser": "1.x",
    "cookie-parser": "1.x",
    "express": "4.x",
    "jade": "1.x",
    "multiparty": "3.x",
    "mysql": "2.x",
    "passport": "0.2.x",
    "passport-local": "1.x",
    "nodemailer": "1.x",
    "node-zip": "1.1.x"
  }
}
```

¹⁷ JavaScriptObjectNotation

Die Orderstruktur unseres Projektes:



Testprotokoll:

Testobjekt	WeLoveGames (1.0.0)
Tester	Lukas Trenkner
Testdatum	19.05.2015

Nr.	Erwartetes Ergebnis	Tatsächliches Ergebnis	Testergebnis
1	Gast kann sich registrieren	Gast kann sich registrieren	Ja
2	Benutzer kann sich anmelden	Benutzer kann sich anmelden	Ja
3	Benutzer kann sich abmelden	Benutzer kann sich abmelden	Ja
4	Gast kann Spiele spielen	Gast kann Spiele spielen	Ja
5	Benutzer kann Spiele spielen	Benutzer kann Spiele spielen	Ja
6	Gast kann keine Highscores übermitteln	Gast kann keine Highscores übermitteln	Ja
7	Benutzer kann highscores übermitteln	Benutzer kann Highscore übermitteln	Ja
8	Gast kann kein fertiges Spiel hochladen	Gast kann keine Spiele hochladen	Ja
9	Benutzer kann ein fertiges Spiel hochladen	Benutzer kann ein fertiges Spiel hochladen	Ja
10	Gast kann kein Spiel erstellen	Gast kann kein Spiel erstellen	Ja
11	Benutzer kann ein Spiel erstellen	Benutzer kann Spiel erstellen	Ja
12	Gast kann kein Spiel freischalten	Gast kann kein Spiel erstellen	Ja
13	Benutzer kann eigenes Spiel aktivieren	Benutzer kann eigenes Spiel aktivieren	Ja
14	Benutzer kann eigenes Spiel deaktivieren	Benutzer kann eigenes Spiel deaktivieren	Ja
15	Gast kann Spiel kein herunterladen	Gast kann kein Spiel herunterladen	Ja
16	Benutzer kann eigenes Spiel herunterladen	Benutzer kann eigenes Spiel herunterladen	Ja
17	Benutzer kann kein anderes Spiel herunterladen	Benutzer kann kein anderes Spiel herunterladen	Ja
18	Gast kann keine eigenen Spiele sehen	Gast kann kann keine eigenen Spiele sehen	Ja

19	Benutzer kann eigene Spiele sehen	Benutzer kann eigene Spiele sehen	Ja
20	Benutzer kann Profil bearbeiten	Benutzer kann Profil bearbeiten	NEIN (Fehler behoben)
21	Benutzer kann seinen Benutzernamen nicht bearbeiten	Benutzer kann seinen Benutzernamen nicht ändern	Ja
22	Admin können alle Benutzer sehen	Admin kann alle Benutzer sehen	Ja
23	Admin kann Benutzer deaktivieren	Admin kann Benutzer deaktivieren	Ja
24	Admin kann Benutzer aktivieren	Admin kann Benutzer aktivieren	Ja
25	Admin kann Benutzer zum Admin machen	Admin kann Benutzer zum Admin machen	Ja
26	Admin kann Admin zum Benutzer machen	Admin kann Admin zum Benutzer machen	Ja
27	Benutzer kann keine Rechte verändern	Benutzer kann keine Rechte verändern	Ja
28	Gast kann keine rechte verändern	Gast kann keine Rechte verändern	Ja
29	Admin kann Spiel aktivieren	Admin kann Spiel aktivieren	Ja
30	Admin kann Spiel deaktivieren	Admin kann Spiel deaktivieren	Ja
31	Webservice kann Datenbankschema autonom anlegen	Webservice konnte Schema anlegen	Ja