KONZEPTIONIERUNG UND ENTWICKLUNG EINES ADMINISTRATOR-BACKENDS FÜR EIN LOCATION-BASED-GAME-FRAMEWORK FÜR MOBILE APPS

Martin Heller

Technische Hochschule Mittelhessen
Wilhelm-Leuschner-Straße 13
61169 Friedberg

ABSTRACT

Im Rahmen dieser Arbeit wurde ein Administrationswerkzeug für ein Framework eines Location-Based Games konzeptioniert und entwickelt. Es wurde in Form einer Web-App realisiert. Die zentrale Aufgabe des Werkzeugs ist das Einpflegen von Spielinhalten in ein Cloud-Backend und das Ändern und Löschen solcher Inhalte. Neben der technischen Umsetzung war ein weiteres Ziel der Arbeit die Bedienung möglichst effektiv, komfortabel und unkompliziert zu gestalten, sodass es auch von Benutzern ohne spezielle technische Kenntnisse und mit möglichst geringem Schulungsaufwand verwendet werden kann. Um dies sicherzustellen, hat man die Entwicklung so gestaltet, dass die Usability des Werkzeugs an mehreren sinnvollen Zeitpunkten u. a. von und mit zukünftigen Benutzern evaluiert wurde. Dadurch hatten sie direkten Einfluss auf die Entwicklung und konnten diese somit aktiv mitgestalten. Der Schwerpunkt dieser Arbeit lag also im Software- und Usability-Engineering.

***Index Terms*—** Usability-Engineering, Software-/Web-Engineering, Parse, React, Parse+React, Framework for Location-Based Games

1. EINLEITUNG

Das Projekt Games@THM der Technischen Hochschule Mittelhessen (THM) am Standort Friedberg entwickelt ein Framework für Location-Based Games, welches G.E.O.R.G.E. genannt wird. Eine Instanz dieses Frameworks ist das Serious Game *Secret Science Society*, kurz *3S*, das entwickelt wurde, um Studierende bei der Orientierung am Hochschulstandort zu unterstützen, die Kommunikation zwischen ihnen zu fördern und ihnen damit deren Studieneinstieg zu erleichtern.

Für dieses Framework benötigt Games@THM ein Administrationswerkzeug, mit dem Inhalte für solche Spiele eingepflegt, bearbeitet und gelöscht werden können. Ein solches Werkzeug sollte unkompliziert, intuitiv und leicht zu bedienen sein und die Arbeit damit sollte möglichst effektiv, effizient und zufriedenstellend erledigt werden können.

Das Framework verwendet als Datenquelle für dessen Inhalte das Backend-as-a-Service[1] (BaaS) von Parse.com, im Folgenden *Parse* genannt, welches diese in einer Cloud speichert. Es ist zwar bereits möglich Inhalte einzupflegen, allerdings ist dies sehr kompliziert und ineffizient.

Was gebraucht wird, ist ein auf die Inhalte von G.E.O.R.G.E. angepasstes Werkzeug, mit welchem sich Inhalte effizienter und zufriedenstellender administrieren lassen, als das bisher möglich ist. Dafür muss ein Weg gefunden werden, das Werkzeug mit *Parse* zu verbinden und Daten aus der Cloud abrufen, anzeigen und ändern zu können. Des Weiteren muss sich überlegt werden, wie eine gute Usability gewährleistet werden kann.

Games@THM möchte, dass das Projekt in Form einer Web-App realisiert und sich auf die Usability konzentriert wird. Dazu wird zusätzlich zum Software-Engineering in einem angemessenen Umfang auch Usability-Engineering betrieben. Dabei wird sich am Prozessmodell von Sardonick und Brau orientiert (s. Abb. 1). Das Modell stellt dar, dass ein Projekt in mehreren Zyklen in der Entwicklung evaluiert werden sollte und dass die Evaluation im Mittelpunkt der Entwicklung steht. Der Softwareentwicklungsprozess wird daher so gestaltet, dass das Projekt in verschiedenen Entwicklungsstadien sinnvoll evaluiert werden kann.

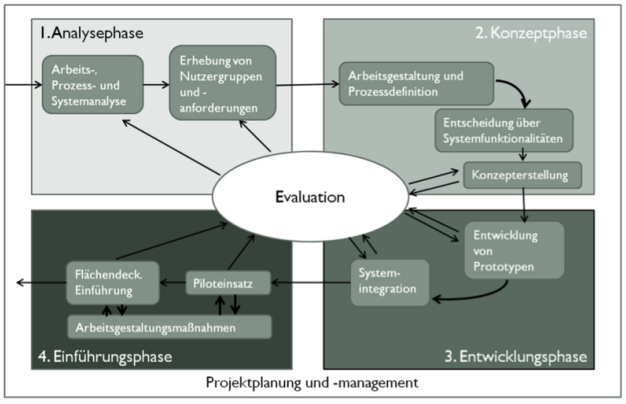

Abb. 1. Prozessmodell *Usability-Engineering* [1]

[1] Unter einem BaaS versteht man eine Backend-Infrastruktur (Server mit Datenbank, APIs, etc.), die von einer externen Firma gehostet wird.

2. VERWANDTE ARBEITEN

Im entfernteren Sinne können Content Management Systeme (CMS), wie z. B. *Wordpress*[2] und *Joomla*[3] als verwandt betrachtet werden, da diese auch die Möglichkeit bieten Inhalte effektiv, effizient und zufriedenstellend zu administrieren. Was diese Systeme von dieser Arbeit unterscheidet, sind die zugrundeliegenden Technologien, wie PHP und relationale Datenbanken[2, 3] im Gegensatz zu den hier zugrundeliegenden Technologien, wie JavaScript und dokumentenbasierte Datenbanken bzw. *Parse*.

Im näheren Sinne verwandt ist das *Parse Dashboard*. Dies ist die bereits erwähnte aktuelle Lösung in Form einer Web-App um Inhalte in die Cloud einzupflegen. Mit diesem können die Inhalte der Datenbank direkt angesehen und Änderungen vorgenommen werden. Allerdings ist dafür einiges an technischem Wissen erforderlich, u. a. Kenntnisse über Pointer, Arrays und die JavaScript-Object-Notation-(JSON)-Syntax. Außerdem sind eigentlich zusammengehörende Daten aus technischen Gründen auf mehrere Tabellen aufgeteilt, sodass dadurch die Übersicht bzw. die Konzentration auf das Wesentliche fehlt. Aus ergonomischer Sicht gibt es also noch viel Verbesserungspotenzial. Und genau an diesem Punkt setzt diese Arbeit an.

3. GRUNDLAGEN

Eine wichtige Grundlage für diese Arbeit ist die Arbeit von Benedikt Kusemann [4]. In dieser Arbeit wurde die Basis des Location-Based-Game-Frameworks entwickelt. Zum einen in Form eines Android-App-Prototypen, der eine Instanz eines solchen Spiels (hier: *3S*) repräsentiert und zum anderen in Form der Modellierung des Schemas der *Parse*-Datenbank, aus der dieser Prototyp die Inhalte für das Spiel bezieht.

Im Spiel *3S* gibt es drei *Fraktionen*, von denen sich ein Studierender einer anschließt: Secret, Science und Society. Des Weiteren gibt es Standorte, auch *Point of Interests (POIs)* genannt, die dem Spieler auf einer *Karte (vom Campus)* angezeigt werden. Standorte können sich auf unterschiedlichen *Stockwerken* befinden. Manche Standorte sind angreifbar, sodass Spieler diese mit Energie angreifen können, um sie für ihre *Fraktion* zu erobern. Dadurch können Spieler Punkte für sich und ihre *Fraktion* sammeln. Manche Standorte sind nicht angreifbar, sondern Vorraussetzung zum Abschließen von *Aufgaben (Quests)* bzw. *Unteraufgaben (Jobs)*. Durch das Abschließen solcher Aufgaben erhält der Spieler sowohl Punkte für seine Highscore, als auch Energie.

4. VERWENDETE TECHNOLOGIEN

Für die Entwicklung des Administrationswerkzeugs wurden u. a. die Standard Web-Technologien, wie HTML5, JavaScript und CSS verwendet. Aufgrund der Verwendung *Parse* wurde das *Parse JavaScript SDK* [4] verwendet, um eine Schnittstelle zur Cloud zu haben.

Darüber hinaus hat *Parse* noch eine weitere JavaScript-Bibliothek zur Verfügung gestellt, welche ebenfalls verwendet wird: die *Parse+React* [5] Bibliothek [5]. Diese verbindet das *Parse SDK* mit der JavaScript Bibliothek *React* [6].

React wiederum ist eine Bibliothek, die sich rein um die konsistente Anzeige von Daten kümmert und mit *Das V in MVC* beschrieben wird. *React* verändert die Art und Weise der Web-Entwicklung drastisch. Webseiten werden nicht mehr mit HTML5 beschrieben, sondern in JavaScript mit *React*. Des Weiteren ist es von *React* vorgesehen, das Web-Projekt nach Komponenten zu entwickeln. Diese haben einen Lebenszyklus, in dem sie ihren Status fortlaufend aktualisieren. Bei der Entwicklung musste daher ein besonderes Augenmerk darauf gelegt werden, dass der Status auch wie gewünscht aktualisiert wird.

Es wurde sich des Weiteren dazu entschieden den *Node.js Package Manager*[7] (*NPM*) zu verwenden, da für diesen einige *React*-Pakete veröffentlicht wurden, die in diesem Projekt verwendet wurden.

Um das Werkzeug leichter ausliefern zu können, wurde die JavaScript Bibliothek *Browserify*[8] verwendet. Diese lies sich über *NPM* installieren. Mit dieser Bibliothek ist es möglich die untereinander referenzierten Node.js-Module in ein JavaScript-Paket zu packen, welches dann direkt im Web-Browser ausgeführt werden kann.

5. METHODIK

5.1. Usability-Engineering

Zur Umsetzung des Usability Engineerings wurden zum einen die zukünftigen Anwender des zu entwickelnden Werkzeugs in die Entwicklung miteinbezogen, sodass sie aktiv an ihr teilhaben und mitwirken konnten. Dadurch konnten u. a. Usability-Probleme frühzeitig erkannt und behoben werden, was zu Zeitersparnis in der Entwicklung und zur Verbesserung der Usability führte. Zum anderen wurde gegen Ende der Entwicklung ein Usability-Test durchgeführt, in dem Benutzer bei der Verwendung des entwickelten Prototypen beobachtet und interviewt wurden. Zusätzlich wurde eine Zeitmessung durchgeführt, um die Effizienz des entwickelten Werkzeugs messen, bewerten und mit der vorhandenen Lösung vergleichen zu können.

[2] https://wordpress.org
[3] https://joomla.org
[4] https://github.com/ParsePlatform/Parse-SDK-JS
[5] https://github.com/ParsePlatform/ParseReact
[6] http://facebook.github.io/react/index.html
[7] https://npmjs.com
[8] http://browserify.org

5.1.1. Anforderungsanalyse

Die Anforderungsanalyse wurde sowohl mit späteren Nutzern als auch den Entwicklern von *G.E.O.R.G.E.* durchgeführt. Dabei kam heraus, dass man Administrationsmöglichkeiten für die im Grundlagenkapitel bereits erwähnten *Fraktionen*, *Karten (Campus)*, *Stockwerke*, *Point of Interests (POIs)*, *Aufgaben (Quests)* und *Unteraufgaben (Jobs)* mit hoher Priorität benötigt. Des Weiteren wurden Statistiken über *Fraktionen*, *Quests* und *Spieler* gewünscht, um das Spiel besser analysieren und ggf. anpassen zu können. Dafür wurde eine mittlere Priorität gesetzt. Optional wurden u. a. Previews gewünscht, die darstellen sollen, wie Inhalte im Spiel aussehen würden.

5.1.2. Anfertigung und Evaluation einer Skizze

Auch im weiteren Verlauf wurden die Benutzer an geeigneten Zeitpunkten mit einbezogen. Vor der Entwicklung eines Prototypen, wurde zuerst eine Skizze erstellt. In dieser Skizze wurden die Seiten zur Administration der o. g. Kategorien entworfen. Um diese Skizze einfacher überarbeiten und mit anderen teilen zu können, wurde die Skizze nicht per Hand, sondern mit der Web-App *Moqups* [9] angefertigt.

Die Skizze wurde insgesamt drei mal von und mit Benutzern evaluiert und entsprechend den Ergebnissen überarbeitet. Durch die Evaluationen konnten zum einen Usability-Probleme aufgedeckt werden und zum anderen haben sie auch zu grundsätzlichen Design-Änderungen geführt.

Die wichtigsten Erkenntnisse seien im Folgenden erwähnt: Es wurde eine Eingabe für Web-Elemente (HMTL-Seiten) und eine Möglichkeit für das Hochladen von Bildern benötigt. Des Weiteren wurde ein Benutzer- und Rollenkonzept gewünscht, um die Bedienung des Administrationswerkzeugs mit eingeschränkter Nutzung auch anderen überlassen zu können und die Verschachtelungstiefe der Administrationsseiten musste reduziert werden.

Die Ergebnisse der Evaluationen flossen in die Entwicklung des Prototypen ein.

5.1.3. Evaluation der Basis-Komponenten

Um die Benutzer möglichst zeitnah wieder in die Entwicklung und Entscheidungen miteinbeziehen zu können, wurden nicht direkt die Administrationsseiten umgesetzt, sondern zuerst die einzelnen Komponenten entwickelt, aus denen diese Seiten zusammengesetzt werden.

Nach der Fertigstellung der einzelnen Komponenten, wurden, diese zur Evaluation zusammen auf einer Webseite aufgeführt. Die Komponenten wurden nach der Fokusgruppenmethode in einem Meeting mit mehreren Personen evaluiert. Die Evaluation hat Folgendes ergeben: Die Speichern- und Abbrechen-Buttons sollen oben rechts in die Navigationsleiste, für eine bessere Navigation soll eine Sidebar erstellt werden, bei der Farbauswahl soll in einem Mini-Preview zu sehen sein, wie Haupt- und Textfarbe zueinanderpassen, Bilder sollen nach *neueste zuerst* geordnet werden, der Bildname soll angezeigt werden, die Anzahl der Bilder pro Seite soll auswählbar sein und die Lokalisierung soll auf einer extra Seite erfolgen. Bis auf die letzteren beiden Punkte wurden die Wünsche bereits umgesetzt.

5.1.4. Evaluation einer erweiterten Prototyp-Version

Die nächste Evaluation wurde durchgeführt, nachdem die Administrationsseiten aus den Komponenten zusammengestellt und diese entsprechend mit der Datenbank verknüpft wurden, zuerst wieder ohne Änderungen zu ermöglichen. In dieser Prototyp-Version konnte das Look & Feel des Werkzeugs zum ersten Mal richtig evaluiert werden. Zum einen, weil es interaktiv bedient werden konnte, zum anderen, weil in der Skizze aufgrund von Limitierungen manche Administrationsseiten nicht bzw. nicht so dargestellt werden konnten, wie sie später aussehen sollten. Auch diese Usability-Evaluation wurde wieder mit mehreren Personen durchgeführt. Das wichtigste Ergebnis war, dass die Seite zur Administration der Stockwerke direkt in die Seite zur Verwaltung der Campus eingebettet werden sollte. Zum einen hat ein Stockwerk nur zwei Eingabefelder, zum anderen wird durch die Einbettung der Zusammenhang zwischen Campus und Stockwerk direkt hergestellt. Des Weiteren wurde vorgeschlagen bei Texteingaben zu prüfen, ob der Text schon vorhanden ist und wenn dem so ist eine Warnung anzuzeigen. Beides wurde entsprechend umgesetzt.

5.1.5. Test des Prototypen im Piloteinsatz

Nach der Fertigstellung eines Prototypen mit allen nötigen Funktionalitäten, wurde ein Usability-Test durchgeführt. Verschiedene Nutzer haben unter Beobachtung den Prototypen ausprobiert. Sowohl das Anlegen neuer als auch das Ändern von existierenden Datensätzen wurde getestet. Dabei wurden Bugs entdeckt, die sich schnell beheben ließen. Es wurde beobachtet, dass die Sidebar nicht gut wahrgenommen und dass der Speichern-Button nicht auf Anhieb gefunden wurde. Von den Nutzern kam durchweg positives Feedback. Im Vergleich zum *Parse Dashboard* ist das Eintragen von Datensätzen sehr viel einfacher, effizienter und zufriedenstellender. Um dies zu untermauern, wurde eine Zeitmessung durchgeführt. Dafür wurde die Zeit, die man für das Anlegen eines Quest-Datensatzes braucht, ein Mal mit dem *Parse Dashboard* und ein Mal mit dem Prototypen gemessen. Mit dem *Parse Dashboard* hat es knapp 19 Minuten, mit dem Prototypen knapp 5 Minuten. Die Effizienz für diese Aufgabe wurde also durch den Prototypen knapp vervierfacht.

[9] https://moqups.com

5.2. Software-/Web-Engineering

Aus der Anfertigung und Evaluation der Skizze ging hervor, welche Komponenten benötigt wurden. Dies sind Eingabe- bzw. Auswahlmöglichkeiten für Lokalisierungstexte, Web-Inhalte, Bilder, Daten und Uhrzeiten, Farben und Geokoordinaten und eine Möglichkeit zur Mehrfachauswahl.

Bei einer Recherche wurden geeignete Komponenten für Web-Inhalte, Daten und Uhrzeiten, Mehrfachauswahl, Farben und eine React-Version für Bootstrap gefunden, welche die Entwicklung etwas erleichterten [6, 7, 8, 9, 10]. Die restlichen benötigten Komponenten mussten komplett selbst entwickelt werden.

5.2.1. Entwicklung der Basis-Komponenten

Jede Komponente wurde konfiguriert, mit weiteren verschachtelt und es wurde sichergestellt, dass sie geladene Daten aus der Cloud annehmen und anzeigen, bei Änderungen ein Event feuern und ihren Status aktualisieren, wie sie sollen.

Die Komponente zur Farbauswahl funktionierte im eigenen Projekt nicht einwandfrei, wie in der Demo [11], sondern musste mit einem gewissen Aufwand angepasst werden.

Zur Umsetzung der Geokoordinaten-Auswahl wurde eine JavaScript Bibliothek namens *leaflet*[10] und eine für diese entwickelte *React*-Komponente[11] verwendet, die eine Weltkarte mit *OpenStreetMap*[12](OSM)-Daten und geeignete Schnittstellen bereitstellt. Mit dieser Komponente kann der Benutzer einfach per Klick auf die Karte einen *POI* bzw. einen Campusbereich auswählen.

Um Bilder und Icons auswählen und hochladen zu können, hat man eine Komponente erstellt, die vier Bilder oder zwölf Icons auf einer Seite anzeigt. Der Benutzer hat die Möglichkeit sich über die Seitennummerierung alle vorhandenen Bilder anzusehen und eins auszuwählen. Des Weiteren wurde eine Fläche hinzugefügt, über die entweder per Drag & Drop oder per Klick Bilder hochgeladen werden können.

Die Komponente zur Eingabe von Lokalisierungstexten wurde aus einer Drop-down-Liste und drei Texteingabefeldern für *Standard*, *Deutsch* und *Englisch* zusammengesetzt. Mit der Drop-down-Liste konnte man aus den vorhandenen Lokalisierungstexten der Datenbank einen auswählen und Änderungen vornehmen.

Um die Komponenten sinnvoll evaluieren zu können, wurden diese auch bereits mit der Cloud-Datenbank verknüpft, sodass Inhalte angezeigt, ausgewählt und teilweise gefiltert werden konnten. Jedoch konnte man noch keine Änderungen mit Auswirkung auf die Datenbank vornehmen (mit Ausnahme des Hochladens von Bildern).

[10] http://leafletjs.com/

[11] https://www.npmjs.com/package/react-leaflet

[12] https://openstreetmap.org

5.2.2. Weiterentwicklung des Prototypen

Im nächsten Schritt wurden die Administrationsseiten aus den Komponenten zusammengestellt und diese mit der Datenbank verknüpft, zuerst wieder ohne Änderungen zu ermöglichen. Je Administrationsseite wurde eine *React*-Komponente entwickelt, die aus einer Navigations- und Seitenleiste, sowie den entwickelten Eingabe- bzw. Auswahlkomponenten zusammengesetzt wurden.

Nach einer weiteren Evaluation wurde das Erstellen, Speichern und Löschen von Datensätzen implementiert. Da einige Funktionalitäten, wie das Speichern von Lokalisierungstexten, Web-Elementen und Arrays, das Löschen von Datensätzen, das Anzeigen von Statusinformationen und Dialogen, etc. von allen Administrationsseiten benötigt wurde, wurde ein sogenanntes *Mixin* programmiert. Ein solches *Mixin* kann in *React*-Komponenten eingebunden werden, um so diese um Funktionalitäten zu erweitern [12]. Dieses wurde entsprechend in die Administrationsseiten eingebunden.

5.2.3. Schwierigkeiten bei der Entwicklung

Bei der Entwicklung gab es einige Schwierigkeiten. Die größte Schwierigkeit war den Lebenszyklus der *React*-Komponenten zu durchschauen. Es war nicht immer klar, wann und warum Komponenten ihren Status aktualisieren. Um dies zu verstehen, wurde zum einen recherchiert, zum anderen musste es aber auch im Debug-Modus nachvollzogen werden. Als Folge musste der Code des Öfteren umstrukturiert werden.

Eine weitere Schwierigkeit war, dass sich *Parse+React* nicht so wie das *Parse SDK* verhalten hat bzw. dass es Bugs hat. Es war z. B. nicht möglich ein Array von Pointer per Query aus einer Datenbank anzufordern [13]. Um das Problem zu lösen, wurde ein Workaround programmiert.

6. FAZIT / AUSBLICK

Es wurde ein neues, auf das Location-Based-Game-Framework zugeschnittenes Administrationswerkzeug entwickelt, welches die Effizienz und Zufriedenheit der Nutzer steigern konnte. Die Entwicklung darauf auszurichten, dass spätere Nutzer daran teilhaben und sie aktiv mitgestalten können, hat dazu beigetragen, die Usability während der Entwicklung stets verbessern zu können. Es ist zu empfehlen die Entwicklung auch weiterhin mit zyklischen Usability-Evaluationen fortzuführen, da man aus diesen immer wieder neue, wertvolle Ideen und Ansätze zur Verbesserung der Usability gewinnen kann.

Es sind noch einige Ideen und Wünsche, die aus den Evaluationen hervorgegangen sind, offengeblieben, die in Zukunft noch umgesetzt werden können und sollten, z. B. das Benutzer- und Rollenkonzept. Auch die anfangs gewünschten Previews und die Seiten für die Nutzungsstatistiken und die Lokalisierung sind noch umzusetzen.

7. REFERENZEN

[1] Florian Sarodnick and Henning Brau, *Methoden der Usability Evaluation*, vol. 2, p. 91, Verlag Hans Huber, 2011.

[2] "Wordpress anforderungen," https://wordpress.org/about/requirements/, abgerufen am 27. September 2015.

[3] "About joomla," https://www.joomla.org/about-joomla.html, abgerufen am 27. September 2015.

[4] Benedikt Kusemann, "Konzeption und Entwicklung einer Androidapp für das Serious Alternate Reality Game *Secret Science Society*," Entwicklungsprojekt, Technische Hochschule Mittelhessen, Februar 2015.

[5] Andrew Imm, "Parse and React - a Shared Chemistry," März 2015, http://blog.parse.com/learn/parse-and-react-shared-chemistry/, abgerufen am 27. September 2015.

[6] "TinyMCE," http://www.tinymce.com/, abgerufen am 30. September 2015.

[7] "React - TinyMCE," https://www.npmjs.com/package/react-tinymce, abgerufen am 30. September 2015.

[8] "React Widgets," http://jquense.github.io/react-widgets/docs/, abgerufen am 30. September 2015.

[9] "React - Color Picker," https://github.com/mapbox/react-colorpickr/, abgerufen am 30. September 2015.

[10] "React - Bootstrap," http://react-bootstrap.github.io/, abgerufen am 30. September 2015.

[11] "React - Color Picker Demo," https://www.mapbox.com/react-colorpickr/example/, abgerufen am 30. September 2015.

[12] "React - Reusable Components: Mixins," https://facebook.github.io/react/docs/reusable-components.html#mixins, abgerufen am 30. September 2015.

[13] "ParseReact - Include Array of Pointer Bug," https://github.com/ParsePlatform/ParseReact/issues/91, abgerufen am 27. September 2015.