Projekt-Schlussbericht

Projekttitel: Unsustainable Alchemy

Autor(en): Michael Günter

Elias Schmidhalter

Datum: 30.05.2015

Inhalt

Teil 1

Management Abstract

Das Projekt „Unsustainable Alchemy“ stellt die Realisierung und Planung einer App für iOS und Android dar. Das Projekt wird im Rahmen von Modul 306 „IT-Kleinprojekt abwickeln“ durchgeführt.

Die App die wir erstellen wollen, wird ein Puzzle-Game sein. Man soll verschiedene Elemente kombinieren und wieder aufteilen können, um neue Elemente zu erschaffen.

Momentan existieren bereits einige solche Applikationen in den App-Stores. Jedoch wollen wir das neue Spielelement, dass man Elemente aufteilen kann, mit einbringen, und so für ein etwas anderes Spielerlebnis sorgen.

Wir haben bereits Vorkenntnisse in JavaScript und AngularJS und können dementsprechend, diese Kenntnisse auf das Projekt anwenden.

Wir haben während des Projekts die Erfahrung gemacht, dass es mithilfe von Webtechnologien sehr einfach ist eine App zu erstellen. Jedoch in dem Bereich Unit-Testing, wird dieses Vorgehen mit Webtechnologien eher schwierig.

Aufgabenstellung

Ausgangslage

Das Projekt ist die Entstehung eines Rätselspiels. Im Spiel wird es darum gehen Elemente zu kombinieren und zu teilen und somit neue Elemente entdecken. Das kombinieren und teilen wird Energie brauchen, diese ist begrenzt. Ziel des Spiels ist es möglichst viele neue Elemente zu entdecken. Das Spiel weist Ähnlichkeiten mit „Little Alchemy“[1] auf.

Das Spiel soll auf iOS und Android laufen. Das Endprodukt wird zu Unterhaltungszwecken veröffentlicht. Wir möchten uns mit diesem Projekt in den Bereichen Mobile- und Spieleentwicklung weiterbilden.

Ziele

In der Initialisierungsphase müssen die genauen Spielregeln und Spielmechaniken ausgearbeitet werden. Auch die Technologie mit welcher das Spiel erstellt wird muss gefunden werden. Ein Zeit und Arbeitsplan muss erstellt werden. Die Risiken der folgenden Phasen müssen erarbeitet und beschrieben werden.

Ausserdem soll die Projektmethode HERMES besser kennengelernt und angewendet werden.

Rahmenbedingungen

Die Initialisierungsphase soll während zwei Vormittagen abgeschlossen werden. Die meisten Bedingungen sind durch den Modulleitfaden gegeben. Ausserdem wird der Zeitplan durch die Präsenzzeiten vorgegeben.

Für die Initialisierungsphase wird weniger Zeit als vorgegeben benötig, da wir bereits eine solide Grund Idee haben. Die Vorgesehenen Dokumente Projektauftrag, Projektplan und Projektführung werden in das Dokument Studie Integriert.

Aufwand

Die Initialisierungsphase soll während zwei Vormittagen abgeschlossen werden. Da wir bereits eine Grundidee haben, werden wir weniger Zeit brauchen. Wir können zudem Zeit bei den Varianten sparen, da wir bereits ähnliche Evaluationen für andere Projekte gemacht haben. Für diese Phase werden wir keine zusätzlichen Materialien und Geldmittel brauchen.

Kosten

Es werden ausschliesslich Personenkosten anfallen. Diese belaufen auf zwei mal sechs Personenstunden. Diese Kosten werden durch die jeweiligen Lehrbetriebe gedeckt. Für dieses Projekt fallen keine Kosten an.

Termine

Die Initialisierung beginnt am 3. Februar 2015 und endet am 10. Februar 2015. Am 17. Februar beginnt die Konzeptionsphase. Diese wird am 17. März enden. Die meisten Termine werden vom Auftraggeber vorgegeben. Weitere Termine werden während der Initialisierungs- und Konzeptionsphase definiert.

Ressourcen

Während der Initialisierungsphase werden vor allem Personalressourcen gebraucht. An der Initialisierungsphase sind die beiden Projektleiter beteiligt.

Ausserdem wird eine Textverarbeitungssoftware (MS Word) verwendet. Diese wird von der GIBB bereitgestellt.

Als Dokumentablage wird GitHub verwendet, diese steht kostenlos zur Verfügung. Die von der GIBB bereitgestellten HERMES vorlagen und Modulunterlagen werden ebenfalls benötigt.

Kommunikation

Die Stakeholder sind die Projektleitung und der Auftraggeber. Die Projektleiter kommunizieren an diversen Meetings. Der Auftraggeber wird anschliessend schriftlich informiert. Weitere Informationen können mündlich ausgetauscht werden.

Risiken

Risiken Beschreibung

1. Es besteht das Risiko, dass die Spielidee nach genauerer Analyse nicht anhand der ursprünglichen Vorstellungen umgesetzt werden kann.
2. Es besteht das Risiko, dass die Spielmechaniken nicht genug genau durchdacht werden und später nicht umsetzbar sind.
3. Ausserdem muss immer damit gerechnet werden, dass ein Projektmitarbeiter verhindert wird. In diesem Fall müsste der Zeitplan angepasst werden.
4. Datenverlust aufgrund von Hardwaredefekt.
5. Der Auftraggeber entscheidet sich das Projekt abzubrechen.
6. Alle Geldmittel sind aufgebraucht.

Risiko Einschätzung

	Unwahrscheinlich	Möglich	Wahrscheinlich
Geringe Auswirkung	6		
Mittlere Auswirkung	5	1	3
Verheerende Auswirkung	2 4		

Massnahmen zur Risikominimierung

1. Dieses Risiko kann nicht minimiert werden.
2. Dieses Risiko kann durch eine gute Planung und Konzeption der Spielmechanismen minimiert werden.
3. Das Risiko selber kann nicht minimiert werden, die Auswirkungen jedoch können durch organisiertes Ablegen der Dokumente und verteilen des Knowhows auf ein Minimum reduziert werden.
4. Dieses Risiko kann durch angemessenen Umgang mit den Geräten und durch Sicherheitskopien minimiert werden.
5. Dieses Risiko kann nicht minimiert werden.
6. Dieses Risiko ist generell gering, da sehr wenige Geldmittel gebraucht erforderlich sind.

Deklaration der Vorkenntnisse

Wir haben beide bereits Erfahrung in der Webentwicklung, und können dieses Wissen auch auf die App Entwicklung übertragen, da wir Webtechnologien verwenden werden.

Michael Günter

- Erfahrung mit Javascript
- Erfahrung mit AngularJS
- Erfahrung mit HTML & CSS
- Erfahrung mit Karmatests

Elias Schmidhalter

- Erfahrung mit Javascript
- Erfahrung mit AngularJS
- Erfahrung mit HTML & CSS
- Erfahrung mit Karmatests
- Vorkenntnisse von Apache Cordova

Deklaration der Vorarbeiten

Für dieses Projekt wurden keine Vorarbeiten geleistet.

Deklaration der verwendeten Firmenstandards

Für dieses Projekt wurden keine Firmenstandards verwendet.

Zeitplan

Für den Zeitplan wurden nachfolgende Arbeitspakete verwendet.

Arbeitspakete

Konzeptionsphase

Definition Spiellogik

In diesem Arbeitspaket wird die Spiellogik genauestens und möglichst detailliert definiert. Es gilt mögliche Schwächen an dem bisherigen Konzept zu identifizieren und auszubauen. Es gilt ein frisches, neues Spielprinzip und Möglichkeiten, diese zu implementieren, zu finden.

UI Design

Alle Benutzerinterfaces müssen designt werden und mögliche Steuerungsmöglichkeiten, die das UI betreffen müssen gefunden werden. Vor allem das Haupt UI, wo sich die Elemente befinden, als auch das Menü, wo die verschiedenen Auflistungen angezeigt werden können, müssen definiert und skizziert werden.

Daten Konzept

Es muss definiert werden, in welcher Form die Daten abgespeichert werden sollen. Soll eine Datenbank verwendet werden, und wenn ja welche Entitäten werden benötigt?

Elemente und Kombinationen definieren

Es muss eine detaillierte hierarchische Struktur von Elementen definiert werden. Ausserdem sollte eine Liste von Elementen angefertigt werden, mit zugehöriger Beschreibung. Die Kombinationen sollten aus der hierarchischen Struktur heraus ersichtlich sein. Möglicherweise gibt es auch eine übersichtlichere Variante der Darstellung. Am Ende sollte man eine Auswahl an ca. 30 Elementen haben, die wir dann auch in die Datenbanktabelle einfügen können.

Technologie Auswahl

Es muss eine Technologie bestimmt werden und genau beschrieben werden, welche Frameworks verwendet werden. Es muss auch definiert werden welches Datenbankmanagementsystem verwendet werden kann.

Game Grundstruktur

Die Grundstruktur bzw. das Projekt sollte angelegt werden und eine Dateistruktur sollte existieren. Ausserdem kann man definieren wo man künftig die Eventbasierte Spiellogik ablegen möchte.

UI Grundstruktur

Hängt fest mit der Game Grundstruktur zusammen. Die grundlegenden Masken und Ansichten der Software sollten stehen, und grundlegende Aktionen sollten von der Gamelogik her steuerbar sein.

Daten Grundstruktur

Die verschiedenen Entitäten sollten in der Datenbank existieren und einige Testdaten, möglicherweise auch schon die Daten für die Elemente und Elementkombinationen enthalten. Der Datenzugriff von der Gamelogik her sollte gewährleistet sein und eine einfache Abfrage sollte auf dem UI ausgegeben werden können. Ein Datenbank-Diagramm wurde erstellt.

Realisierungsphase

Gamelogik: Kombinieren von Elementen

Die Logik sowie die UI Aktionen für das Kombinieren von Elementen müssen umgesetzt werden. Ein Element muss per Drag & Drop verschoben werden können und sich mit einem anderen Element kombinieren, wenn es über dem anderen Element losgelassen wird. Aus den beiden Elementen entstehen zwei weitere, identische Elemente.

Gamelogik: Aufteilen von Elementen

Die Logik sowie die UI Aktionen für das Aufteilen von Elementen müssen umgesetzt werden. Wenn eine lange Berührung auf ein Element stattfindet, muss es sich in ein zufällig ausgewähltes Eltern-Element verwandeln. Damit wird auch das Austauschen der Grundelemente umgesetzt.

Gamelogik: Energieleiste

Die Logik sowie das UI für die Energieleiste müssen umgesetzt werden. Bei bestimmten Aktionen muss die Energieleiste an Energie verlieren. (Welche genau, müssen noch definiert werden). Wenn die Energieleiste leer ist, ist das Spiel vorbei und der Benutzer erhält erneut, die zur Verfügung stehenden Grundelemente. Die zuvor erhaltenen Elemente bleiben jedoch in der Auflistung der Elemente vorhanden.

Gamelogik: Element Liste

Die Logik sowie das UI für die Auflistung der Elemente. Im Hauptmenü des Spiels, sollen alle bisher erhaltenen Elemente aufgelistet werden. Zu jedem Element sollen jeweils das Bild, der Name sowie eine kurze Beschreibung ersichtlich sein.

Gamelogik: Erfolg Liste

Die Logik sowie das UI für die Auflistung der Erfolge. Im Hauptmenü des Spiels, sollen alle bisher freigeschalteten Erfolge, sowie eine bestimmte Anzahl nicht freigeschalteter Erfolge ersichtlich sein. Pro Erfolg sollen der Name und eine kurze Beschreibung ersichtlich sein. Bei nicht abgeschlossenen Erfolgen soll der Name und das Ziel, um den Erfolg freizuschalten, ersichtlich sein.

Grafiken erstellen und einbinden

Allgemeine Grafiken, wie beispielsweise der Titelbildschirm oder der Game-Over Bildschirm, sowie alle Grafiken für alle Elemente müssen erstellt (Pixelart) und später dem entsprechenden Datensatz zugeordnet werden. Ausserdem müssen die Grafiken an den entsprechenden Orten im Spiel angezeigt werden. Die Erstellung der Grafiken erfolgt nebenbei, während der gesamten Realisierungsphase bis der zugehörige Meilenstein erreicht ist.

Tests

Die einzelnen hinzugefügten Funktionen müssen fortlaufend, während der Entwicklung getestet werden. Ausserdem muss am Schluss ein Systemtest folgen, der möglichst alle Situationen und Elementkombinationen abdeckt.

Entwicklungsdokumentation (Realisierungsbericht)

Die Entwicklungsdokumentation wird fortlaufend, während der Entwicklung geführt und aktualisiert. Sie sollte die Architektur sowie wichtige Aspekte der Lösung abdecken.

Einführungsphase

Zusatzinfos für Publishing schreiben

Es müssen Zusatzinformationen, die für das Veröffentlichen auf den Stores benötigt werden, verfasst werden. Dazu gehören, Screenshots, ein App-Icon, eine Beschreibung möglichst auf Englisch und Deutsch.

Einführungsbericht

Der Einführungsbericht muss verfasst werden.

Einführungspräsentation

Die Präsentation muss vorbereitet werden, und ein PowerPoint muss angefertigt werden. Danach muss das Produkt präsentiert werden.

Nr.	Arbeitspakete / Tätigkeiten	Anforderungen	Ist in h	Soll in h	Datum	27.01.2015	03.02.2015	10.02.2015	17.02.2015	24.02.2015	03.03.2015	10.03.2015	17.03.2015	24.03.2015	21.04.2015	28.04.2015	05.05.2015	12.05.2015	19.05.2015	26.05.2015
1	**Konzept**		4	4	SOLL															
	Definition Spiellogik				IST															
2	UI Design	Alchemy Table	4	4	SOLL															
					IST															
3	Daten Konzept	Rezepte	6	4	SOLL															
					IST															
7	Elemente und Kombinationen definieren	Rezepte	6	6	SOLL															
					IST															
4	Technologie Auswahl		1	1	SOLL															
					IST															
5	Game Grundstruktur	Neues Spiel, Spielende	4	4	SOLL															
					IST															
6	UI Grundstruktur	Alchemy Table	6	3	SOLL															
					IST															
7	Daten Grundstruktur		8	4	SOLL															
					IST															
8	**Realisierung**																			
	Gamelogik: Kombinieren von Elementen	Kombinieren	8	8	SOLL															
					IST															
7	Gamelogik: Splitten von Elementen	Teilen	4	4	SOLL															
					IST															
8	Gamelogik: Energieleiste	Energie, Energieleiste	1,5	4	SOLL															
					IST															
7	Gamelogik: Element Liste	Element Übersicht	8	4	SOLL															
					IST															
8	Gamelogik: Erfolg Liste	Achievements Übersicht	0	4	SOLL															
					IST															
9	Assets Erstellen und Einbinden	Grafiken, Logos & Splashcreeen, Athmosphäre & Ambiente, Texturen und Hintergründe, Sounds	14	12	SOLL															
					IST															
10	Testing		5	8	SOLL															
					IST															
11	Entwicklungs Doku (Realiserungsbericht)		18	8	SOLL															
					IST															
12	**Einführung**																			
	Zusatzinfos für Publishing schreiben		2	2	SOLL															
					IST															
13	Einführungsbericht		4	6	SOLL															
					IST															
14	Einführungs Präsentation		4	4	SOLL															
					IST															
15					SOLL															
					IST															
16					SOLL															
					IST															
18	**Meilensteine**								① ②				③ ④		⑤	⑥		⑦	⑧ ⑨	⑩
19	① Abschluss Initialisierungsphase																			
20	② Abgabe Studie																			
21	③ Abschluss Konzeptphase, Grundarchitektur																			
22	④ Abgabe Konzeptbericht																			
23	⑤ Erste Funktionierende Version des Spiels																			
24	⑥ Fertigstellung der Grafiken																			
25	⑦ Abschluss Realisierungsphase (Spiel fertig)																			
26	⑧ Abgabe Realisierungsbericht																			
27	⑨ Einführung (Publishing im Store)																			
28	⑩ Abgabe Einführungsbericht																			

Arbeitsprotokoll

Elias Schmidhalter

Arbeitsjournal

Datum: 24. März 2015

Tatsächlicher Zeitbedarf	Geplanter Zeitbedarf	Beschreibung der Arbeit	Bemerkungen, Probleme, genutzte Hilfestellungen
3.5h	2h	Sqlite Plugin Installieren (mit ngCordova)	ngCordova konnte nicht einfach umgesetzt werden (musste weggelassen werden) Abfrage Querys funktionierten nicht direkt.
0.5h	2h	Stammdaten hinzufügen und Querys schreiben	

Reflexion des Arbeitstages

Zu beginn der heutigen Lektion haben Michael und ich gemeinsam versucht das Sqlite Plugin und ngCordova zum Laufen zu kriegen. Nach langem testen und versuchen haben wir entschieden, dass der aufwand ngCordova zu verwenden grösser ist, als das Sqlite Plugin direkt anzusprechen. Wir haben uns entschieden ngCordova wegzulassen.

Anschliessend waren wir damit beschäftigt das Sqlite Plugin zum Funktionieren zu bringen. Dies war etwas kompliziert, da wir für jede Änderung am Code die Applikation neu Kompilieren und auf ein iPhone Deployen mussten. Nach einigen Fehlgeschlagenen Versuchen konnten wir Daten aus der Datenbank auslesen und einfache schreib Querys absetzen.

Wir mussten nun noch die Daten die wir vom SQLite Service bekamen so umformen, dass sie zu den JSON Objekten die wir in der Applikation verwenden passten.

Bis das dies Fertig war, war die Lektion schon fast zu ende und es verblieb wenig zeit für den Aufbau der Stammdaten.

Zum Schluss konnten wir noch beginnen die Stammdaten und Datenstruktur aufzubauen.

Pendenzen für den nächsten Arbeitstag

Wir müssen noch die verbleibende Datenbankstruktur in Sqlite abbilden und eine Funktion zum erstmaligen einfügen und zurücksetzen der Basisdaten schreiben.

Michael Günter

Arbeitsjournal

Datum: 24. März 2015

Tatsächlicher Zeitbedarf	Geplanter Zeitbedarf	Beschreibung der Arbeit	Bemerkungen, Probleme, genutzte Hilfestellungen
3.5h	2h	SQLite-Plugin einbinden und Datenzugriff sicherstellen	Wir hatten vor ngCordova zu verwenden, jedoch konnten wir es damit nicht umsetzen und mussten das Plugin manuell einbinden.
3h	2h	DB-Schema und Stammdaten-Generator realisieren, sowie beispielhafte Stammdaten erfassen	Ich bin auf das Problem gestossen, wie ich mehrere asynchrone Queries aneinanderhängen kann und in der korrekten Reihenfolge ausführen kann.

Reflexion des Arbeitstages

Zu Beginn der Lektion habe ich noch einige kleinere Bugs gefixt. Danach war ich jedoch fast die ganze Zeit mit Elias daran das SQLite-Plugin, das wir für den Datenzugriff verwenden wollen, einzubinden. Im Konzept hatten wir geplant, dass wir das Plugin mit ngCordova einbinden möchten. Jedoch wollte dies nicht recht funktionieren und wir sind dabei auf massive Probleme gestossen. Nachdem wir lange damit rumprobiert haben, entschieden wir uns das Plugin von Hand, ohne ngCordova, einzubinden, was dann nach einigen Versuchen (diese waren etwas kompliziert, weil wir es immer auf einem nativen Device testen mussten) auch klappte. Der Aufwand mit ngCordova wäre schlicht zu gross gewesen. Wir konnten dann sicherstellen, dass bei der Ausführung auf einem mobilen Gerät das SQLite Plugin verwendet wird und ansonsten WebSQL. Wir waren nun in der Lage ein erstes Query auf die Datenbank abzusetzen.

Als dies funktionierte haben wir einen SQLite Service geschrieben, der ein einfaches Query annimmt und die Resultate in JSON formatierte Objekte umformt und asynchron zurückliefert.

Diese Arbeit dauerte, aufgrund der erwähnten Probleme, deutlich länger als erwartet weshalb dann der Morgen auch schon fast zu Ende war und wir nicht mehr, wie geplant, die Dinge mit den Stammdaten machen konnten. Wir haben lediglich begonnen einige Queries für das DB-Schema zu schreiben.

Nun war die Arbeit in der Schule zu Ende und als ich am Abend nachhause kam, arbeitete ich weiter.

Ich habe den DB-Populator bereitgestellt. Dazu musste ich zuerst die Queries für das DB-Schema und für die Beispiel-Stammdaten finalisieren. Danach habe ich den DB-Populator Service geschrieben, welcher grundsätzlich beim ersten Applikationsstart das DB-Schema anhand des Queries und die Stammdaten anhand der Queries generiert und somit die Datenbank betriebsbereit macht.

Leider bin ich dabei auch wieder auf Probleme gestossen. Die SQL Queries die wir alle schön mit Semikolons abgetrennt hatten funktionierten so leider nicht. Das WebSQL API (SQLite-Plugin verwendet auch dasselbe API) akzeptiert leider nicht mehrere Queries, die mit Semikolons getrennt sind. Also musste ich die Queries auftrennen und irgendwie die Methode vom SQLite Service einzeln pro Query aufrufen. Dies erwies sich als sehr schwer weil man die Queries (vorallem die Queries vom DB-Schema) in der richtigen Reihenfolge ausführen muss, damit sie funktionieren. Dass Queries asynchron ausgeführt werden erschwerte zudem die Sache. Deshalb musste ich eine Chain-Funktion schreiben, die mehrere Queries annimmt und diese in der korrekten Reihenfolge ausführt. Ich sass die meiste Zeit an dieser Funktion und musste lange probieren bis sie richtig funktionierte.

Doch dann funktionierte es endlich und die Datenbank konnte erfolgreich betriebsbereit gemacht werden.

Pendenzen für den nächsten Arbeitstag

Der nächste Schritt ist nun das Laden der Daten, sowie das Implementieren der Queries beim Kombinieren und Aufteilen. Danach folgt das automatische Speichern der Position der Elemente.

Teil 2

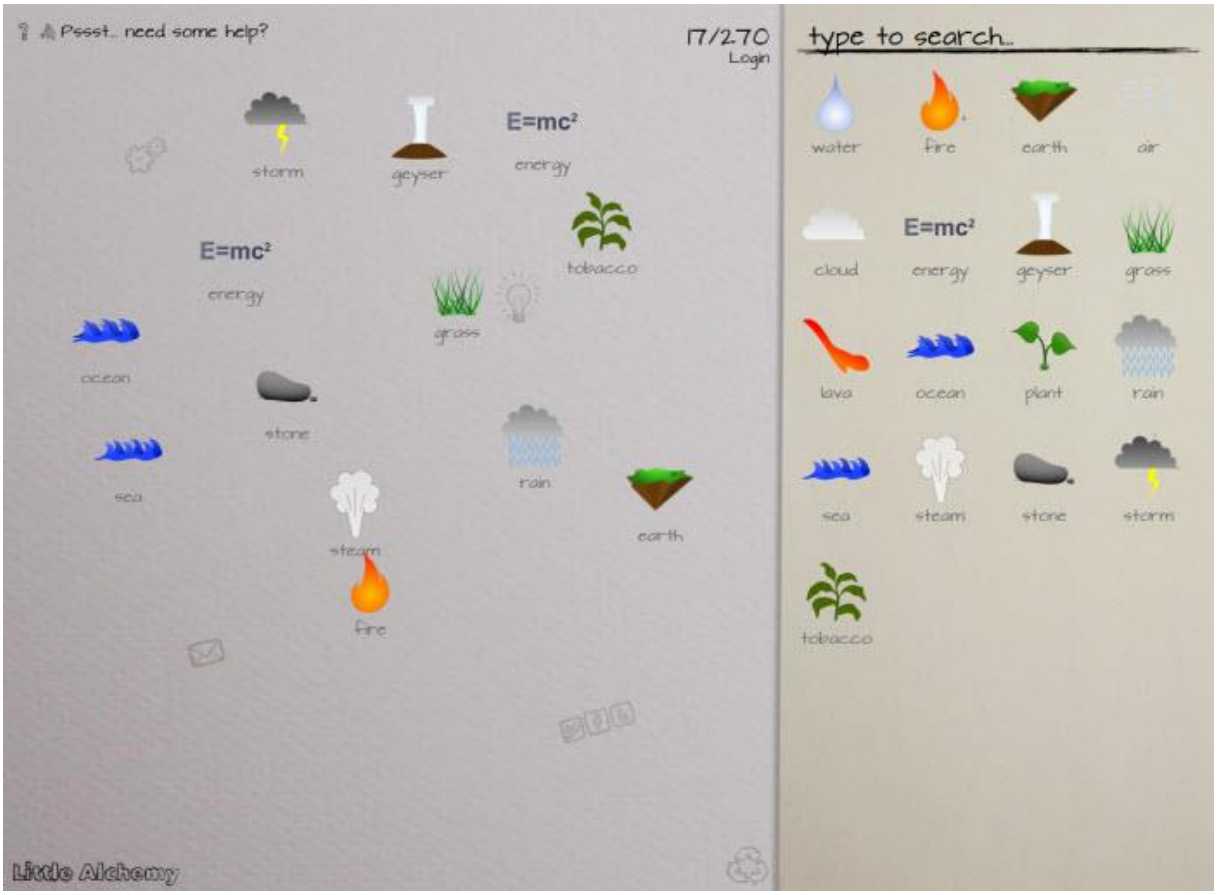
Situationsanalyse

Ausgangslage

Es ist vorgesehen, dass wir, wie in dem Initialisierungsauftrag geschildert, ein Spiel entwickeln, in welchem man verschiedene Elemente oder Gegenstände kombinieren und splitten kann, um neue Elemente zu schaffen. Mit den neuen Elementen lassen sich wiederum durch dieselben Prinzipien neue, andere Elemente erschaffen. Das grundsätzliche Ziel eines solchen Spiels, ist es den Benutzer zu unterhalten.

Momentan gibt es schon einige Spiele, welche dieses Prinzip nutzen. Sowohl im Browser als auch im mobilen Bereich.

- Little Alchemy, für Google Chrome, Android und iOS.
- Alchemie, für Android und iOS



Diese Spiele haben über die Zeit sehr viele neue Elemente erhalten. Bei Little Alchemy sind es beispielsweise, momentan ca. 550 verschiedene Elemente. Dies wird nicht unser Ziel sein. Wir werden uns auf ein erweitertes Spielprinzip konzentrieren, welches auch vorerst mit ca. 30 – 50 Elementen auskommt. Unser Spiel soll später auch in die App-Stores kommen, um möglichst viele Benutzer anzusprechen.

Stärken

Die Stärken der aktuell existierenden Spiele sind vor allem die unglaubliche Vielfalt an Elementen. Ausserdem macht es Spass, herauszufinden, welche Elemente kombiniert werden können. Auch verschiedene Lustige Aspekte wurden beachtetet, indem zum Beispiel bei der Kombination bestimmter Elemente, etwas Unerwartetes, Lustiges herauskam.

Da wir uns um ein etwas abgeändertes Konzept bemühen werden, ist es nicht von Nöten, dass wir eine immense Vielfalt an Elementen implementieren. Viel wichtiger ist jedoch, dass es dem Spieler nicht langweilig wird, und wir auch ähnlich wie in den bestehenden Spielen, einige lustige Elemente einbringen.

Zusammenfassung:

- Vielfalt der Elemente
- Spass beim Kombinieren
- Lustige Aspekte

Schwächen

Die grösste Schwäche an der heutigen Situation ist eigentlich nur eine einzige. Die heute existenten Spiele berufen sich allesamt auf dasselbe Spielprinzip. Somit gibt es viele Spiele, die zwar auf den ersten Blick einzigartig erscheinen, jedoch sich nachher als Kopie eines anderen Spiels, lediglich mit etwas abgeänderten Elementen, entpuppt.

In unserem Projekt werden wir genau diesen Schwachpunkt verstärkt behandeln. Wir wollen ein neues, überarbeitetes Konzept einbringen. In den aktuellen Spielen ist es nämlich so, dass man einen Elemente-Vorrat hat, welcher unbegrenzt ist. Sobald man ein Element entdeckt hat, ist es in unbegrenzter Masse verfügbar.

Dies wollen wir mit Unsustainable Alchemy ändern. Wir wollen, dass der Spieler immer gleich viele Elemente auf dem Spielfeld hat, und nur diese zum Kombinieren zur Verfügung hat. Ausserdem soll auch das Splitten von Elementen möglich sein, damit dem Spieler ermöglicht wird, in der Element-Hierarchie hoch und runter zu navigieren.

Ausserdem versuchen wir verschiedene Erfolge einzubauen, um auch zwischendurch, beim Erlangen bestimmter Elemente, dem Benutzer das Gefühl zu übermitteln, erfolgreich zu sein.

Zusammenfassung:

- Alle Spiele basieren auf demselben Konzept
- In manchen Spielen, ist eine gute Bedienbarkeit nicht gewährleistet.
- Es gibt keine Möglichkeit, zu verlieren.
- Man hat erst Erfolg, wenn man alle Elemente gefunden hat.

Systemziele

Der folgende Abschnitt stellt eine Übersicht über alle Ziele dar, die wir während des gesamten Projekts erreichen möchten.

Schwächen bezogene Ziele

Ziel	Schwäche	Beschreibung	Erreichung	Kategorie
1	1, 3, 4	Wir wollen, während der Konzeptionsphase, ein neues, überarbeitetes Spielprinzip, welches sich von den bisherigen Prinzipien unterscheidet, erarbeiten, und später in der Realisationsphase implementieren.	Das Ziel ist erreicht, sobald wir das Spielprinzip implementiert haben.	Muss
2	2	Wir wollen eine übersichtliche Bedienbarkeit gewährleisten, welche besser als bei manchen anderen Spielen ist. Diese wollen wir während der Konzeptionsphase erarbeiten und während der Realisation implementieren.	Das Ziel ist erreicht, sobald wir die Benutzeroberfläche benutzerfreundlich implementiert haben.	Kann

Andere Ziele

Ziel	Beschreibung	Erreichung	Kategorie
3	Wir wollen dem Spieler primär ein unterhaltsames Erlebnis liefern, während er unser Spiel spielt.	Das Ziel ist erreicht, wenn das Spiel unterhaltsam ist.	Kann
4	Wir wollen während der Realisierung unser Wissen im Bereich App- und Spielentwicklung erweitern.	Das Ziel ist nach der Realisierungsphase erreicht.	Muss
5	Wir wollen, während wir die die Grafiken für das Spiel erstellen, erlernen, wie man Pixelgrafiken schön aussehen lässt. Dabei sollen schöne Pixelgrafiken erstellt werden.	Das Ziel ist erreicht, wenn wir schöne Pixelgrafiken erstellen können.	Kann
6	Das Spiel soll am Ende der Realisierung auf den gängigsten Mobilplattformen (Android und iOS) lauffähig sein.	Das Ziel ist erreicht, sobald die App auf beiden Plattformen läuft.	Muss
7	Das Spiel soll am Ende der Realisierung flüssig laufen und nicht ruckeln.	Das Ziel ist erreicht, sobald das Spiel flüssig läuft.	Kann
8	Wir wollen am 17.02.2015 mit der Konzeptionsphase beginnen, welche am 3.3.2015 abgeschlossen werden soll, damit wir danach mit der Realisationsphase beginnen können.	Das Ziel ist erreicht, wenn der Zeitrahmen eingehalten werden kann.	Muss
9	Das Spiel wird am Ende in den Stores veröffentlicht.	Das Spiel ist veröffentlicht.	Kann

Rahmenbedingungen

Wir werden am 17. Februar mit der Konzeptionsphase des Projekts beginnen. Diese wollen wir, während ca. 3 Wochen beenden. Sobald diese Phase beendet ist, gilt es die Software zu realisieren. Für die Konzeptionsphase werden wir keine zusätzlichen Aufwände ausserhalb der Schulzeit verbuchen müssen, da die Zeit, die wir zur Verfügung gestellt bekommen vollkommen ausreicht. Die meiste Zeit werden wir uns also in der Gibb mit dem Projekt beschäftigen. Während der Realisierungsphase wird es von Nöten sein, ausserschulische Aufwände zu betreiben, welche meist Zuhause oder Unterwegs vollführt werden. Die Kommunikation und Organisation erfolgt dabei auf den heute gängigen Kommunikationsmitteln. Source Code und Dokumente werden auf GitHub verwaltet und zur Verfügung gestellt.

Abgrenzung

Wir wollen unser Projekt vorerst nicht zu Umfangreich gestalten. Das bedeutet, dass wir uns vorerst auf ca. 30 verfügbare Elemente beschränken werden. Wir wollen mit dem Projekt nicht unsere Grafikkünste ins Rampenlicht stellen, sondern uns geht es eher um das ausgefeilte Spielprinzip. Unser Vorhaben soll vorerst eine Basis für ein gutes Spiel bieten, welches man später mit zusätzlichen Elementen erweitern kann.

Lösungsvorschläge

Variantenübersicht

- Variante A: Mobile Lösung mit normalen Elementen
- Variante B: Mobile Lösung mit chemischen Elementen
- Variante C: Native mobile Lösung

Variante A: Mobile Lösung mit normalen Elementen

Die Variante A soll mit Apache Cordova umgesetzt werden, und Webtechnologien für die Darstellung verwenden. Durch die Webtechnologien ist man etwas eingeschränkter, als mit einer nativen Lösung, die Entwicklung geht jedoch schneller und effizienter voran.

Die Lösung soll ca. 30 Elemente beinhalten, welche bei bestimmten Elementkombinationen auch etwas Lustiges ergeben. Es werden keine chemischen Elemente, sondern eher Elemente aus dem Alltag verwendet.

Der Benutzer hat die Möglichkeit verschiedene Erfolge zu erreichen, und diese in einem separaten User-Interface anzuzeigen.

Vorteile

- Schnelle Entwicklung durch Webtechnologien
- Gleicher Code für alle Plattformen
- Lustige Aspekte, durch Kombinationen vorhanden

Nachteile

- Eingeschränkte Möglichkeiten durch Webtechnologien

Variante B: Mobile Lösung mit chemischen Elementen

Die Variante B soll ähnlich wie die Variante A aufgebaut werden.

Jedoch sollen anstelle der normalen Elemente, chemische Elemente verwendet werden. Dadurch gehen jedoch möglicherweise die lustigen Aspekte der Kombinationen verloren. Man könnte die Elemente statt in einer klassischen Liste, in einem Periodensystem auflisten.

Vorteile

- Schnelle Entwicklung durch Webtechnologien
- Gleicher Code für alle Plattformen

Nachteile

- Lustige Aspekte sind nicht vorhanden
- Eingeschränkte Möglichkeiten durch Webtechnologien

Variante C: Native mobile Lösung

Eine Native Applikation ist eine, welche direkt für ein Gerät geschrieben wurde. Dies wird normalerweise in einer vom Hersteller bereitgestellten Programmiersprache gemacht.

Eine Native Applikation ist sehr vorteilhaft, wenn man oft auf Systemfunktionen wie zum Beispiel die Kamera zugreift. Ausserdem ist die native Programmierung sehr performant und gut integriert. Die Hersteller legen viel Wert darauf, dass es gute und detaillierte Guidelines und vorgefertigte Bausteine gibt. Der grosse Nachteil dieser Variante ist, dass für jede Plattform, also in diesem Fall iOS und Android, die ganze App neu entwickelt werden muss.

Vorteile

- Alle Möglichkeiten durch native Entwicklung
- Performant

Nachteile

- Plattformabhängige Entwicklung in verschiedenen Sprachen
- Erhöhter Zeitaufwand

Bewertung der Varianten

Anhand folgender Tabelle werden die verschiedenen Lösungsvarianten miteinander verglichen. Sie dient als Entscheidungsgrundlage für die Wahl der Variante.

Variante	Zeitaufwand (2x)	Unterhaltung (1x)	Realisierbarkeit (2x)	Score
Variante A	8	9	8	**41**
Variante B	7	4	8	34
Variante C	4	9	4	25

Die Tabelle bewertet die 3 Lösungsvarianten nach ihrem Zeitaufwand, der Unterhaltung des Endprodukts und der Realisierbarkeit des Projekts.

Je höher der Wert für den Zeitaufwand desto weniger Zeit nimmt die Realisierung in Anspruch. Je höher der Wert für die Unterhaltung desto mehr wird der Benutzer des Spiels unterhalten. Je höher der Wert für die Realisierbarkeit desto einfacher oder besser ist es realisierbar.

Wir treffen anhand des Gesamt-Scores unsere Entscheidung auf die Variante A, weil sie den niedrigsten Zeitaufwand bietet, den grössten Unterhaltungsfaktor sowie, durch die Webtechnologien einfach und schnell realisierbar ist. Der Unterhaltungsfaktor ist bei Variante A grösser als bei Variante B, weil es mehr Spass macht, Elemente die man kennt zu kombinieren, anstatt völlig unbekannte chemische Elemente zu kombinieren.

Lösungsbeschreibung

Unsere Lösung wird mit Apache Cordova, das heisst Javascript, HTML und CSS, umgesetzt. Apache Cordova ist ein Framework, welches die Entwicklung von mobilen Anwendungen mit Webtechnologien ermöglicht und Schnittstellen zu Anbindungen (bsp. Kamera) bereitstellt. Wir haben uns dazu entschieden, weil es eine schnelle und effiziente Möglichkeit bietet, die Lösung zu realisieren.

Spielprinzip

Wir berufen uns auf ein bereits vorhandenes Spielprinzip und ändern dieses ab, um eine gewisse Frische des Spiels zu gewährleisten.

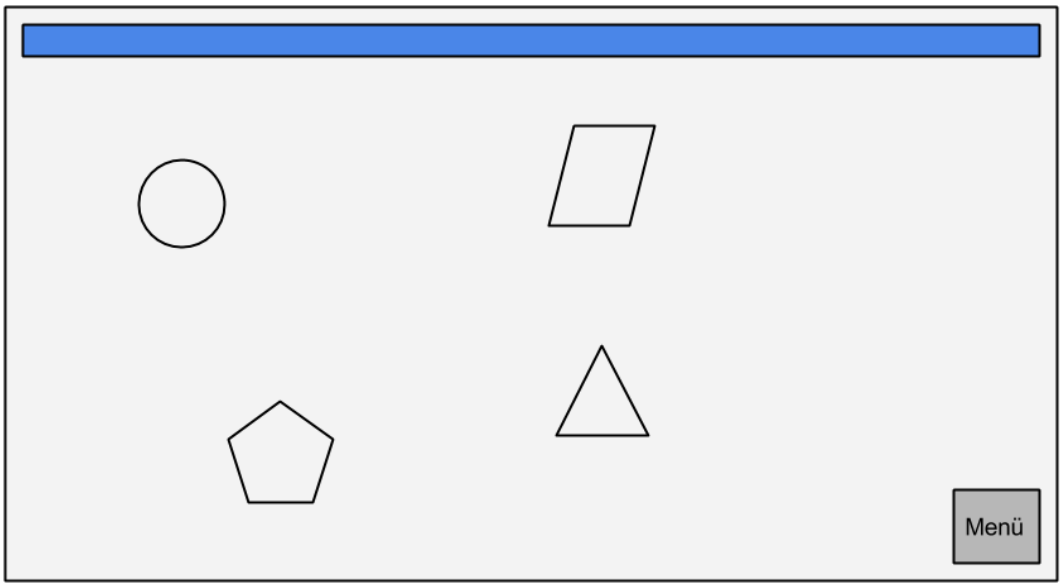
Grundsätzlich stehen dem Spieler am Anfang eine bestimmte Anzahl (ca. 5 oder 6) Elemente zur Verfügung. Diese werden als Grundelemente bezeichnet. Die Grundelemente lassen sich durch andere Grundelemente austauschen. So kann man durch die verschiedenen Grundelemente durchwechseln.

Alle Elemente lassen sich kombinieren. Aus zwei Elementen entstehen 2 andere, jedoch gleiche Elemente. Das Resultat der Kombination kann manchmal logisch, manchmal aber auch lustig sein. Diese beiden Elemente lassen sich nicht mehr auswechseln. Es lassen sich lediglich die Grundelemente auswechseln. Sie lassen sich jedoch stattdessen aufteilen. Durch die Aufteilung wird möglicherweise zufällig gewählt welches der Eltern-Elemente durch die Aufteilung entsteht. Man kann durch das Aufteilen jedoch nur die Eltern-Elemente erhalten.

Es existiert eine zusätzliche Energiebar. Diese verliert bei verschiedenen Aktionen Energie. (Welche genau, müssen wir noch in der Konzeptionsphase festlegen). Wenn die Energie aufgebraucht ist, ist das Spiel zu Ende und man muss wieder mit den Grundelementen anfangen.

Das Ziel des Spiels ist es durch diese Möglichkeiten, so viele Elemente wie möglich zu finden. Jedes gefundene Element wird in einer Liste eingetragen und detaillierte Informationen über das Element stehen zur Verfügung. Beim Erhalt bestimmter Elemente werden Erfolge freigeschaltet.

Benutzeroberfläche



Grundsätzlich kombiniert der Spieler die verschiedenen Elemente durch ziehen und verschieben, der auf der Fläche herumliegenden Elemente. Werden zwei Elemente übereinander losgelassen, so kombinieren sie sich. Werden sie lange berührt, so splitten sie sich. Oben soll die Energiebar eingeblendet werden.

Durch den Menü Button soll der Benutzer eine Übersicht über die erhaltenen Elemente erhalten und die Erfolge einsehen können.

Anforderungen

Anforderung	Ziel	Beschreibung
1 (Neues Konzept)	1	Als Spieler will ich mit einem neuen Konzept überrascht werden, um neue Dinge auszuprobieren.
2 (Kombinieren)	1	Als Spieler will ich Elemente kombinieren und teilen, um neue Elemente zu erschaffen.
3 (Glücksfaktor)	1	Als Spieler erwarte ich einen gewissen Glücksfaktor, damit nicht alles von Strategie und Nachdenken abhängt.
4 (Misserfolg)	1	Als Spieler will ich, dass ich verliere wenn die Energiebar keine Energie mehr enthält. Damit ich ein Gefühl des Misserfolgs erhalte.
5 (Erfolge)	1	Als Spieler will ich gelegentlich mit Erfolgen belohnt werden, um Motivation, Spass und ein Erfolgsgefühl zu erlangen.
6 (UI)	2	Als Spieler will ich ein übersichtliches, gut aussehendes, Benutzerinterface bedienen, damit ich einen professionellen Eindruck des Spiels erhalte.
7 (Elemente Übersicht)	2	Als Spieler will ich eine Übersicht über die erhaltenen Elemente. Die Übersicht soll Informationen über die einzelnen Elemente enthalten, damit der Spieler immer informiert über den Fortschritt ist.
8 (Erfolgs Übersicht)	2	Als Spieler will ich eine Übersicht über die erzielten Erfolge. Die Übersicht soll Informationen zu erhaltenen Erfolgen und Anweisungen zu freischaltbaren Erfolgen enthalten, damit der Spieler weiss, was es als nächstes zu holen gibt.
9 (Unterhaltung)	3	Als Spieler will ich von dem Spiel unterhalten werden, um die Langeweile zu eliminieren.
10 (Motivation)	3	Als Spieler will ich lustige Aspekte in dem Spiel vorfinden, damit ich motiviert werde, noch mehr lustige Elemente zu finden.
11 (Weiterbildung)	4	Die Entwickler entwickeln ein mobiles Spiel, um Wissen im Bereich App- und Spielentwicklung zu erweitern.
12 (Grafiken)	5	Die Entwickler geben sich Mühe, schöne Pixelgrafiken für die Elemente zu erstellen, damit ein angemessenes Spielerlebnis für den Spieler ermöglicht wird.
13 (Android/iOS)	6	Als Spieler will ich das Spiel auf Android und iOS spielen können, um eine optimale Kompatibilität auf allen Plattformen abzudecken.
14 (Performance)	7	Als Spieler will ich, dass das Spiel flüssig und performant läuft, damit es nicht zu unerwünschten Unterbrechungen kommt.
15 (Zeitplan)	8	Als Projektleiter wollen wir, dass wir die geplante Zeit für die Konzeptionsphase einhalten können, damit wir pünktlich mit der Realisation beginnen können.

Diese Anforderungen werden alle durch unsere Lösung abgedeckt.

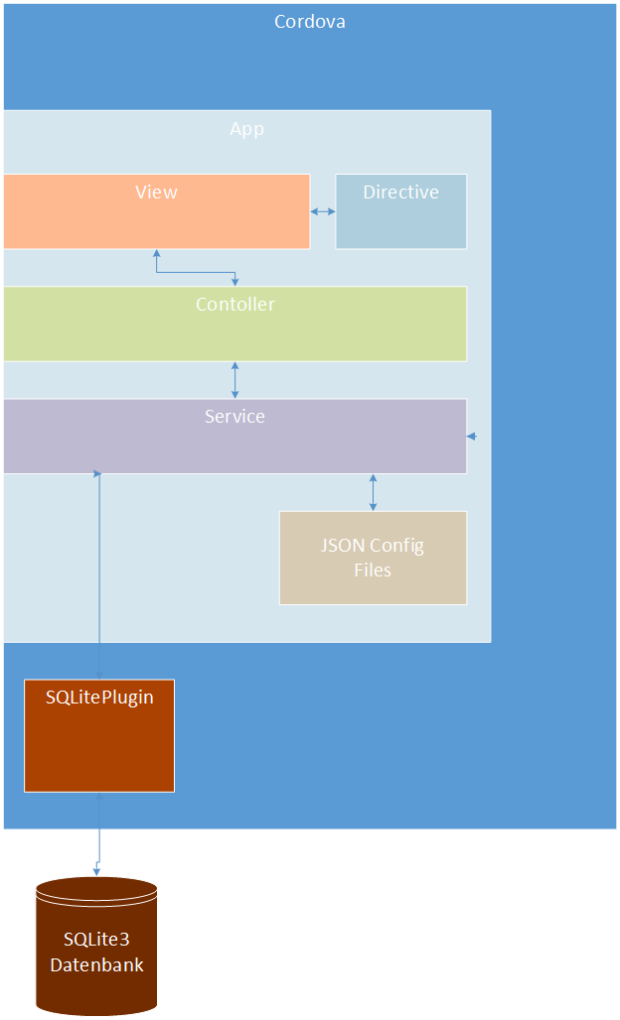
Systemarchitektur

Systemdesign

Das Systemdesign beschreibt die statische Struktur der Applikation sowie die grundlegende Dynamik. Es stellt dar, wie die verschiedenen Module miteinander interagieren.

Struktur

Hervorgehend aus dem Konzeptbericht haben wir eine grundlegende Struktur definiert.



Auf diesem Konzept haben wir aufgebaut, und wir haben unsere Lösung in diese grundlegenden Module unterteilt.

Wie gehabt umschliesst Apache Cordova unsere gesamte Applikation und stellt die Ausführung auf den mobilen Betriebssystemen sicher. Danach ist die Lösung unterteilt in die Module App, View, Directive, Controller und Services. Ausserdem existiert eine SQLite Datenbank für die Speicherung des Spielfortschritts und für die Bereitstellung von Stammdaten.

View

Eine View ist ein HTML-Dokument, welches verschiedene UI Komponenten anordnet und grafisch aufbereitet darstellt.

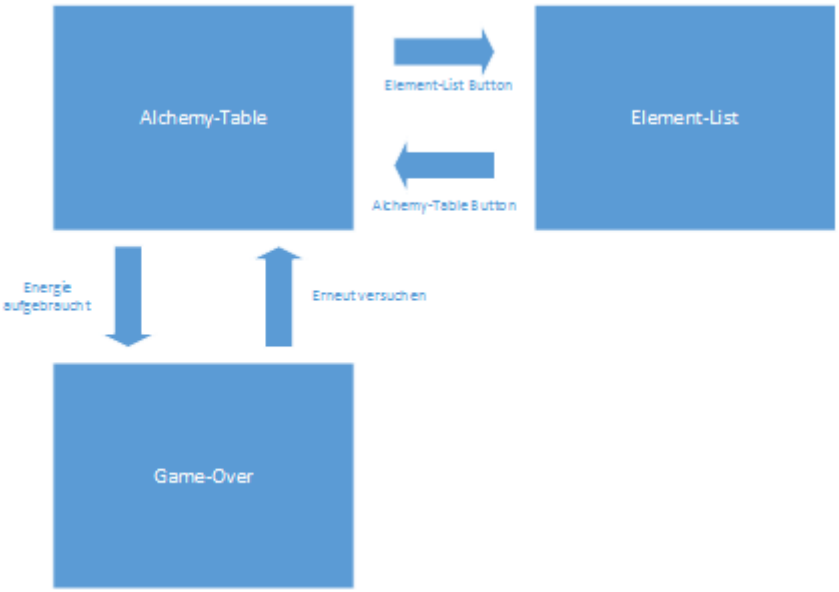
Durch die Benutzung von AngularJS haben wir die Möglichkeit direkt in den Views bestimmte Aktionen durchzuführen.

Wir können zum Beispiel verschiedene Directives in eine View einbinden. Eine Directive ist hat meist ein eigenes Layout und eigene Logik. Sie stellt grundsätzlich eine zusammengefasste UI-Komponente dar.

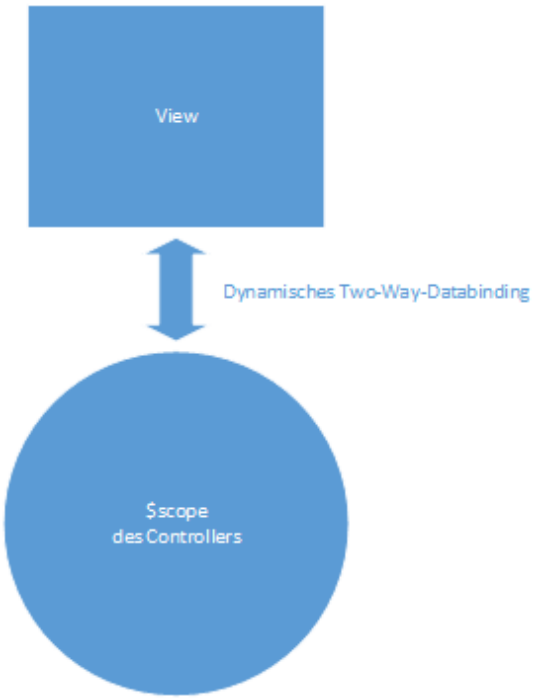
Wir haben genau 3 Views. Einmal der normale Alchemy Table, einmal die Element-Liste und einmal der Game-Over Screen. Zu den Views zählen technisch gesehen auch die Templates für die Directives. Diese werden jedoch später genauer beschrieben.

Im Hintergrund jeder View arbeitet ein korrespondierender Controller, der die JavaScript-Logik implementiert.

Die 3 Views hängen folgendermassen zusammen:



Im Hintergrund aller Views arbeitet ein Controller an der Anwendung der Logik. Der Controller besitzt Variablen auf dem $scope, die direkt auf der View mittels eines dynamischen Two-Way-Databindings verknüpft sind. Heisst, wenn auf dem UI ein Wert verändert wird, kann der neue Wert im Controller unmittelbar verwendet werden. Das gleiche gilt umgekehrt: Wenn im Controller der Wert verändert wird, wurde die Änderung auch direkt im UI übernommen. Dieses Databinding gilt für alle Variablen auf dem $scope des Controllers.



Aber das Two-Way-Databinding wirkt keineswegs nur zwischen View und Controller, sondern auch innerhalb des Controllers und innerhalb der View, wenn zum Beispiel an zwei Stellen auf dieselbe Variable zugegriffen wird. Auch die Argumente für Directives sind nach Wahl ein Databinding und ändern so direkt die Variable im Directive und auch umgekehrt, Änderungen im Directive werden über das Argument zurück in die View reflektiert.

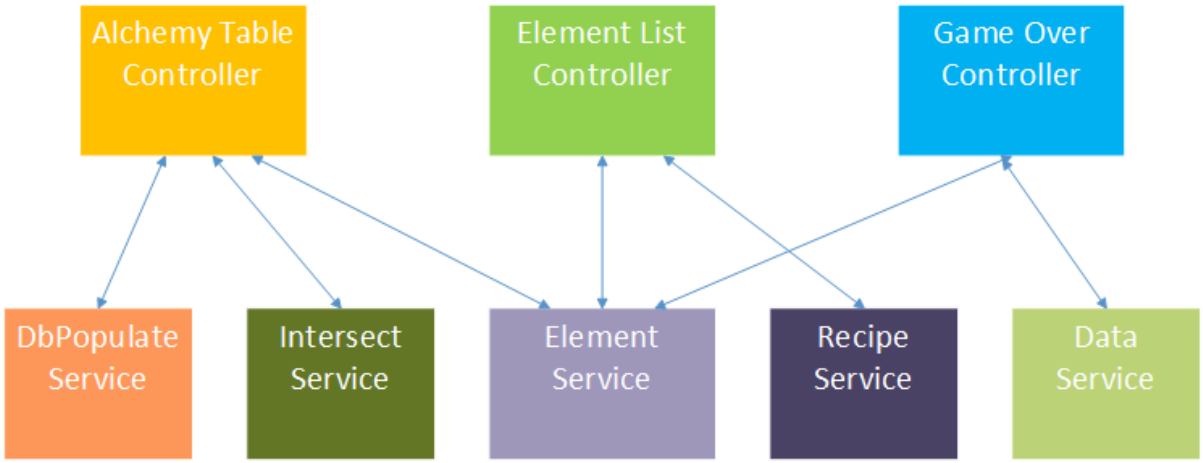
Wir haben uns entschlossen unser ViewModel (Die Variablen auf der View) nicht direkt auf das $scope zu legen, weil es dann leicht versehentlich die globalen Variablen auf dem $rootScope oder die Variablen von hierarchisch höher gelegenen $scopes, überschreiben könnte. Deshalb haben wir ein ViewModel Objekt eingeführt, auf welchem wir unser ViewModel ablegen. Wir schreiben also anstelle von $scope.test -> $scope.vm.test, wobei vm für ViewModel steht.

Controller

Der Controller bereitet die Daten aus den Services für die Anzeige auf der View auf. Der Controller schreibt die Daten auf das ViewModel. Das ViewModel ist via Data-Binding an die View gebunden.

Der Controller kann auf verschiedene Services zugreifen. Die Services werden via Dependency Injection eingebunden. Für jede View wurde auch ein Controller erstellt.

Für jede View existiert ein korrespondierender Controller. In folgender Abbildung ist ersichtlich, welche Services von welchen Controllern verwendet werden.



Die spezifischen Aufgaben der Services werden später in diesem Dokument beschrieben.

Service

Der Service beinhaltet die Business Logik des Programmes. Hier wird auf die Persistenz Schicht zugegriffen, Daten aufbereitet und eingaben von den Controllern bearbeitet.

Wir haben grundsätzlich den SQLite Service welcher den zugriff auf die Datenbank sicherstellt. Der SQLite Service baut eine Datenbankverbindung zum Datenbankprovider auf. Dieser ist auf den mobilen Endgeräten der lokale SQLite3 Dienst. Wenn die Applikation im Browser gestartet wird, wird, falls vorhanden WebSQL verwendet.

Der SQLite Service bietet eine Query Methode an. Diese kann ein prepared SQL-Query auf der Datenbank ausführen. Zusätzlich gibt es eine Chain Methode, die mehrere Queries nacheinander in der richtigen Reihenfolge ausführt.

Darauf bauen einige Services wie zum Beispiel der DbPopulate Service dieser befüllt die Datenbank mit Stammdaten. Er ist aufgeteilt in die Erstellung des Datenbankschemas und andererseits in das befüllen der Datenbank mit Stammdaten. Der DbPopulate Service wird nur aufgerufen, wenn die App das erste Mal gestartet wird.

Der Dataservice erstellt anhand von SQL Abfragen Business Entitäten (JSON Objects).

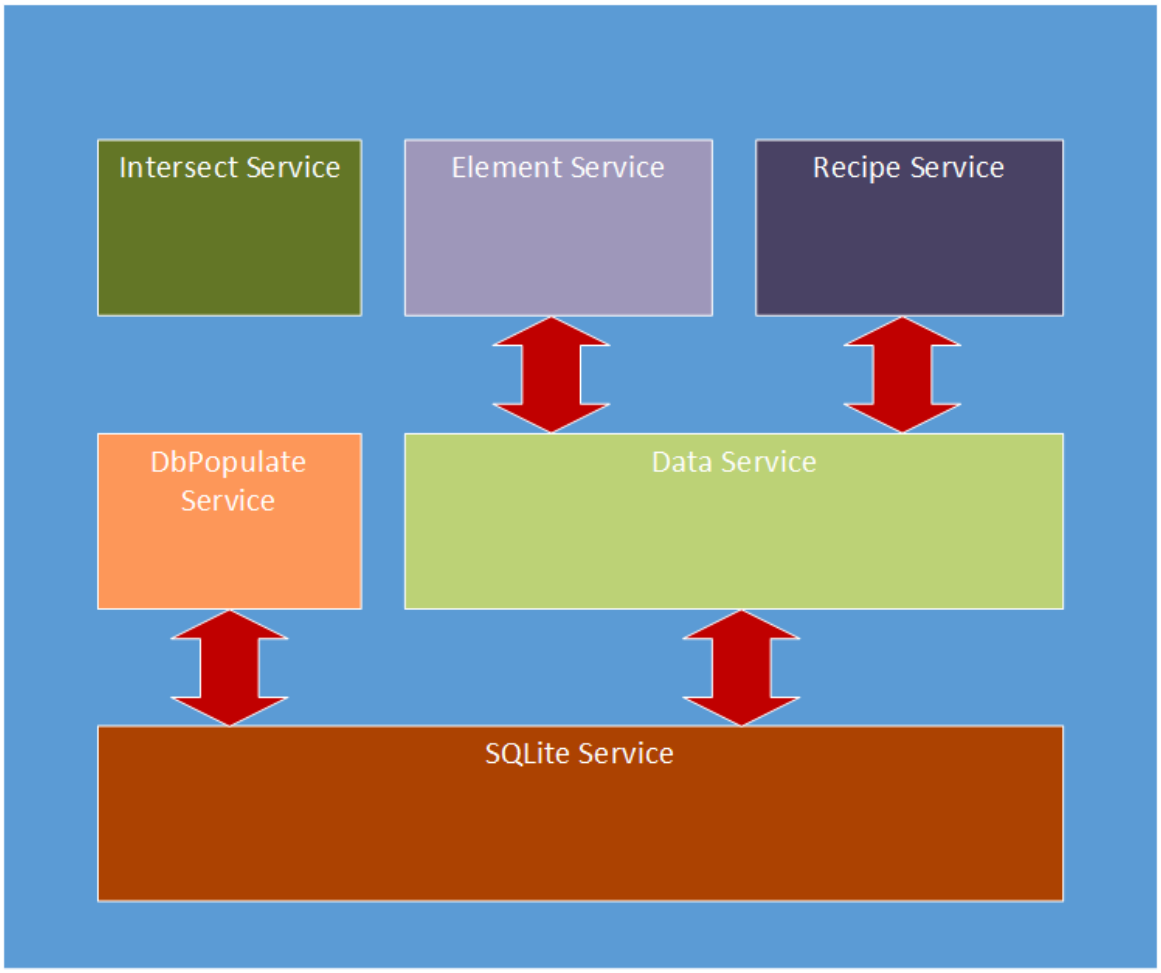
In diesem Service sind die SQL-Statements abgelegt. Der Dataservice verwendet den SQLite Service um die abgelegten Queries auf der Datenbank auszuführen.

Zu den Methoden des Dataservice gehören folgende:

- getCurrentElements, holt alle Elemente, die sich aktuell auf dem Alchemytable befinden
- getAllElements, holt alle Elemente
- getCombinedElement, holt anhand von zwei Elementen das kombinierte Element
- getElementParts, holt anhand des kombinierten Elements die beiden Element Teile
- getBaseElements, holt alle Elemente ohne Rezept
- getBaseElementsExcept, holt alle Elemente ohne Rezept, ausser ein definiertes Element
- isBaseElement, gibt true zurück wenn das Element ein Basis-Element ist
- restoreBaseElements, setzt die aktuellen Elemente auf dem Alchemy Table zurück auf die Basis-Elemente
- updateCurrentElement, setzt die Position des übergebenen Elements
- updateCurrentEnergy, setzt die übrige Energie des Spielers
- getUnlockedRecipes, holt alle freigeschaltenen Rezepte und die zugehörigen Elemente
- unlockRecipe, schaltet ein Rezept frei

Dieser Dataservice wird vom Element und Recipe Service verwendet. Diese bieten eine Schnittstelle zwischen dem Datenservice und den Controllern. Im Element Service befindet sich die Logik zum Kombinieren, Teilen und Zurücksetzen von Elementen. Ansonsten ist er eine pure Schnittstelle zum Dataservice.

Ausserdem besteht noch der Intersect Service dieser macht keine Zugriffe auf die Datenbank, er überprüft lediglich ob zwei Elemente sich überschneiden. Dabei existieren die Überprüfung zur Überschneidung von zwei Elementen, sowie eine Überprüfung für zwei Positionen.



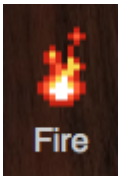
Directive

Direktiven sind zusammengefasste UI-Komponenten. Für dieses Projekt haben wir insgesamt drei verschiedene Direktiven erstellt.

Die Logik der Directives befindet sich in den einzelnen JavaScript Dateien. Die HTML Datei nennen wir Template. Das Template wird vom Directive referenziert. Einem Directive können verschiedene Parameter übergeben werden. Dies erfolgt direkt auf der View, wo das Directive eingebunden wurde. Diese sind meist dynamisch an das ViewModel des Controllers gebunden, und Änderungen des Wertes in der Directive werden unmittelbar auf das ViewModel des Controllers zurückreflektiert.

Alle Directives werden auf dem Alchemy Table verwendet.

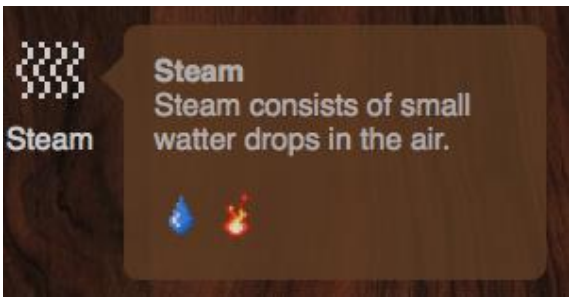
Element Directive



Für die Darstellung und die Interaktion mit Elementen haben wir eine Element Direktive erstellt. Die Logik für das Verschieben von Elementen, sowie der Aufruf für das Kombinieren und Splitten von Elementen wird in dieser Direktive implementiert.

Die Abbildung zeigt das Aussehen der Direktive auf dem Alchemy Table.

Flupp Directive



Für die Darstellung detaillierter Informationen haben wir ein Flupp Directive erstellt. Das Flupp Directive ist eine Art Popout oder Popover, welches erscheint wenn kurz auf das Element getippt wird. In dem Directive ist Logik für das Öffnen und Schliessen des Flupps, aber auch für die Darstellung der Daten vorhanden.

Die Abbildung zeigt das Aussehen der Direktive nachdem das Element angetippt wurde.

Energybar Directive

Für die Darstellung der Energie Leiste haben wir eine Direktive erstellt. Das Directive dient nur dazu, die Bar richtig darzustellen und konstant zu aktualisieren. Die Abbildung zeigt die Energie Leiste, die auf dem Alchemy Table dargestellt wird.

App

Im App-Modul werden globale Konfigurationen vorgenommen. Darunter fallen die Konfiguration für die Datenbank, die Routen und der App Start.

Routen

Die Routen befinden sich im routes.js und definieren für jede View einen Controller und unter welchem State diese erreichbar sind.

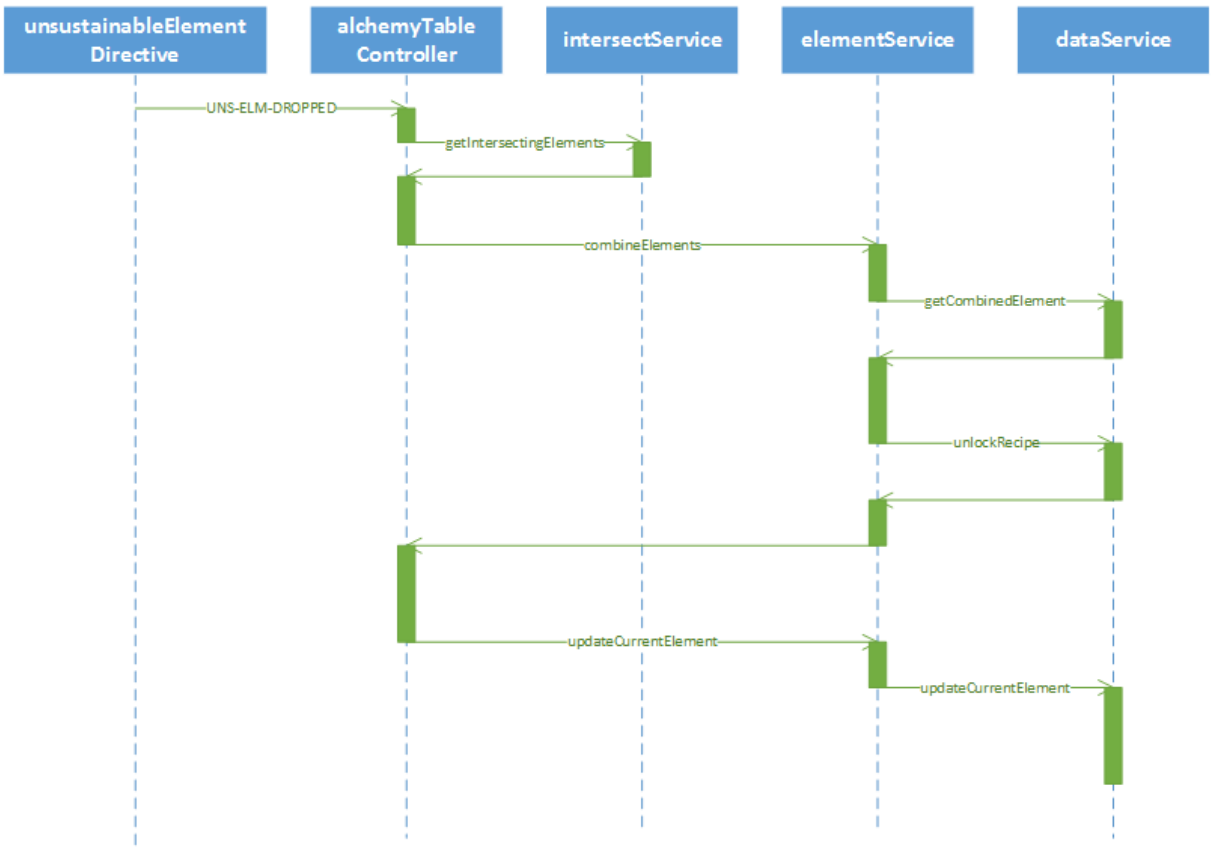
App Start

Beim Appstart wird das Device Ready von Cordova abgewartet, falls die App auf einem mobilen Endgerät ausgeführt wird. Dies ist notwendig, da ansonsten der Datenbankzugriff über das Plugin nicht gewährleistet ist. Anschliessend wird die App gestartet.

Das Index-HTML ist das erste Dokument, das von Cordova aufgerufen wird. Hier wird grundsätzlich der Bereich definiert, in dem später die Views eingebunden werden. Ausserdem werden die Styles (CSS) eingebunden. Wichtige Einstellungen für den Viewport werden auch hier vorgenommen. Die Scripts werden eingebunden. Dabei ist zu beachten, dass wir lediglich das main.js einbinden, welches vorher gebundelt wurde. Zusätzlich muss cordova.js eingebunden werden, um die Funktionalität von Cordova sicherzustellen. Diese Datei existiert nur auf dem mobilen Endgerät und ist im Source-Code nicht ersichtlich.

Dynamik

In diesem Abschnitt geht es darum die Dynamik der Applikation zu verständlichen. Dazu werden zwei wichtige Prozesse der Dynamik beschrieben. Einerseits das Kombinieren und andererseits das Aufteilen von Elementen.



In der obigen Abbildung ist das Sequenzdiagramm fürs Kombinieren ersichtlich.

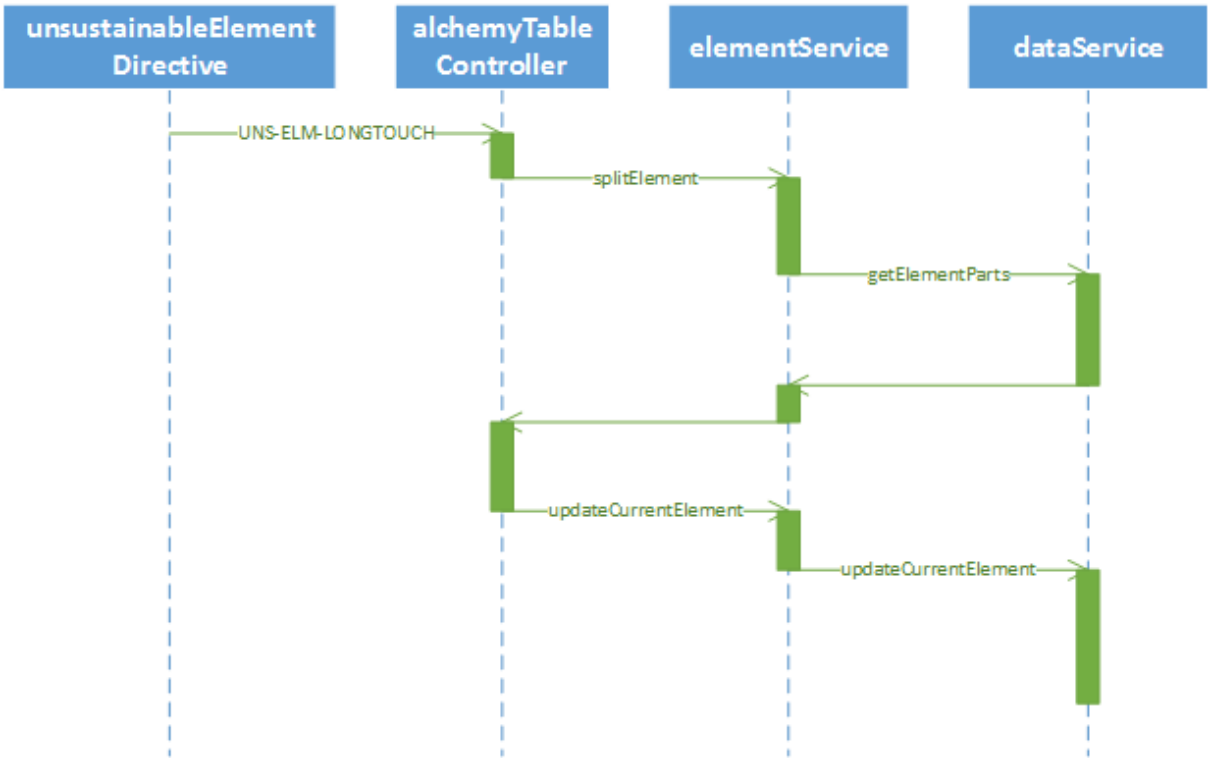
Zu allererst feuert das Element-Directive ein Event wenn es vom Spieler abgelegt wird. Dieses Event nennen wir intern das UNS-ELM-DROPPED Event. Alle Elemente auf dem Alchemy Table feuern jeweils dieses Event wenn sie nach dem verschieben gedroppt werden.

Auf dem Alchemy Table Controller hören wir dann auf dieses Event und warten darauf, bis es eintritt. Wenn es eingetreten ist konsultiert der Alchemy Table Controller den Intersect Service um herauszufinden, ob das Element sich mit einem anderen Element überschneidet. Dazu holt er alle überschneidenden Elemente. Wenn es ein überschneidendes Element gibt, wird dieses mit dem Element aus dem Event kombiniert. Dies geschieht mithilfe des Element Service. Dieser wiederum benutzt den Data Service um ein Query mittels des SQLiteService (welcher oben nicht abgebildet ist, da es sich nur um ein einfaches Absetzen eines Queries handelt) auszuführen.

Der Data Service holt das Resultat aus der Kombination. Ausserdem wird danach noch das Rezept freigeschaltet, falls das kombinierte Element neu ist.

Das kombinierte Element wird an den Alchemy Table Controller zurückgeliefert, welcher dann wiederum, mithilfe des Element Service, die aktuell auf dem Spielbrett befindlichen Elemente, die zur Kombination benötigt wurden, aktualisiert und durch zwei Instanzen des kombinierten Elements ersetzt.

Alle Aufrufe für diesen Prozess erfolgen asynchron.



Die obige Abbildung dient zur Veranschaulichung des Prozesses des Aufteilens von Elementen.

Als erstes sieht man, dass vom Element Directive ein Event gefeuert wird, wenn das Element lange berührt wird. Wir nennen dieses Event intern UNS-ELM-LONGTOUCH. Der Alchemy Table Controller wartet bis dieses Event eintritt und leitet danach das Aufteilen in die Wege. Dazu konsultiert er den Element Service welcher wieder den Data Service verwendet um die Element Einzelteile zu erhalten. Ein zufälliges Einzelteil wird ausgewählt und an den Alchemy Table Controller zurückgeliefert. Dieser aktualisiert danach mithilfe des Element Service das Element auf dem Alchemy Table und ersetzt es mit dem Einzelteil des aufgeteilten Elements.

Alle Aufrufe für diesen Prozess erfolgen asynchron.

Promises

Im gesamten Projekt haben wir das Konzept der Promises angewandt ($q service von Angular, basierend auf JQuery Q). Eine Promise ist ein Objekt welches dem Aufrufer verspricht irgendwann in der Zukunft die angeforderte Aktion auszuführen und ein Resultat zurückzuliefern. Dieses Resultat lässt sich mit dem .then() call auf einer Promise abwarten. Alles was in der Callback Function des .then() ausgeführt wird, wird zu einem späteren Zeitpunkt, wenn die Promise finished ist, ausgeführt. Als Parameter der Callback Function im .then() wird das zurückgelieferte Resultat mitgegeben, welches in der Function verwendet werden kann. Nahezu jeder Aufruf, welcher oben in den Diagrammen ersichtlich ist, erfolg per Promise.

Schnittstellendefinition

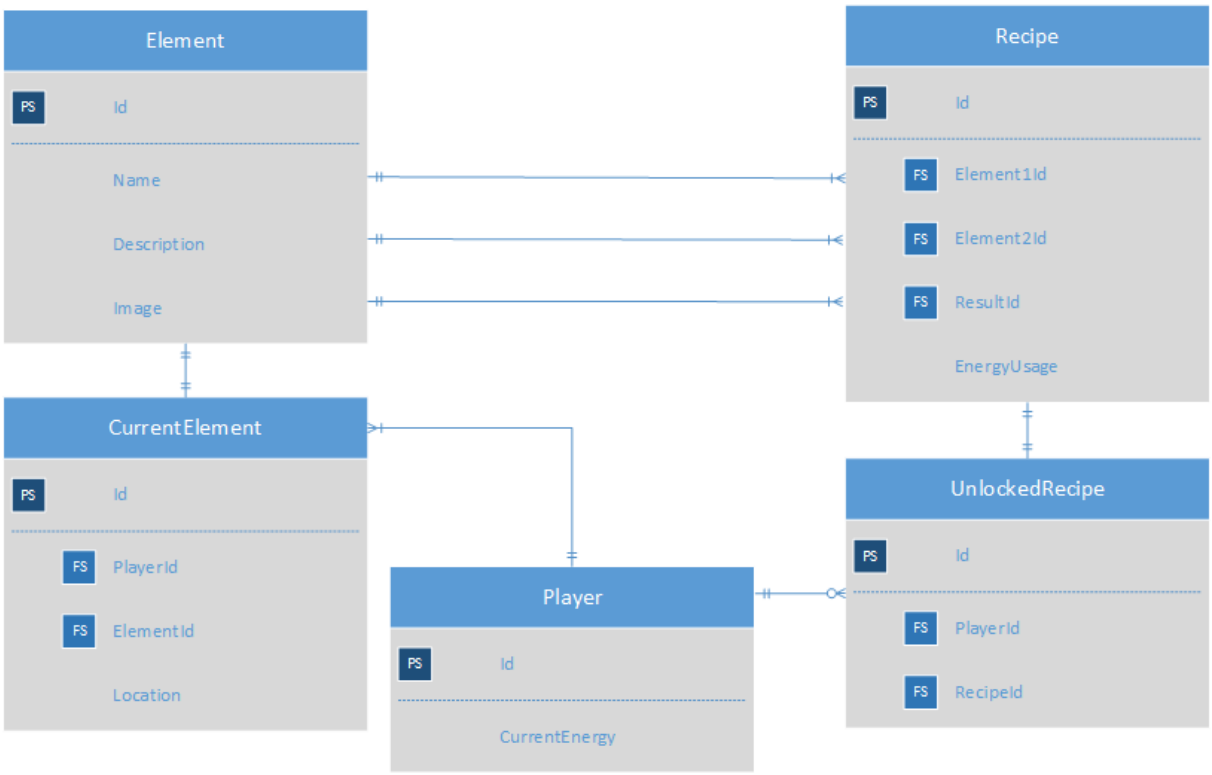
Wir haben eine Schnittstelle für die Persistenzschicht. Diese verwenden wir um Daten zu holen und zu speichern.

Dazu existiert der SQLiteService welcher alle Zugriffe auf die Datenbank regelgt. Dazu wird das Cordova SQLite Plugin verwendet. Der DataService ist eine Abstraktion des SQLiteService, der die einzelnen SQL-Queries definiert. Diese SQL Queries werden dann vom SQLite Service auf der Datenbank ausgeführt.

Im SQLiteService werden die SQL-Queries dem SQLite Plugin übergeben. Das Plugin führt diese anschliessend aus. Als Resultat erhalten wir Result-Sets, die anschliessend vom SQLiteService zu JSON Objekten gemappt werden.Auf dem Result-Set ist eine Liste von Rows, die die einzelnen Zeilen des Resultats repräsentieren. Diese Zeilen haben wie von SQL gewohnt als Attribute die im Query selektierten Tabellenspalten.

Datenmodell

Dieses Datenmodell ist bereits aus dem Konzeptbericht bekannt. Bitte betrachten Sie folgende Abbildung.



Es hat sich seit dem Konzept einiges geändert. Was auffällt ist, dass die Achievements und die UnlockedAchievements fehlen. Grund dafür ist, dass wir die Achievements aus zeitlichen Gründen nicht implementiert haben. Ausserdem wurden die Fehler aus dem Diagramm aus dem Konzept ausgebügelt.

Testkonzept und Testspezifikation

Testkonzept

Während der Entwicklung wenden wir das Test-Driven-Development an. Das heisst, dass wir erst Testfälle (Unit Tests) schreiben und erst danach die Funktionalität umsetzen. Die Testfälle sind identisch zu den User Stories und sind dann erfüllt wenn alle User Stories erfolgreich und ohne Fehlermeldung durchgespielt werden können.

Testspezifikation

Allgemein haben wir bei unseren Tests die Tests für die Achievements weggelassen, da die Achievements auch nicht realisiert wurden.

Beschreibung	Drag and Drop Elemente können verschoben werden
Abgedeckte Anwendungsfälle	1. Drag and Drop
Ausgangssituation	Die App wurde gestartet.
Vorbereitungsschritte	1. Die App Installieren und Starten

Testschritte	**Erwartetes Resultat**
1. Element mit Finger an einen anderen Ort ziehen.	Das Element folgt dem Finger.
2. Element loslassen	Das Element bleibt an dieser Stelle.

Beschreibung	Zwei Elemente kombinieren.
Abgedeckte Anwendungsfälle	2. Kombinieren 4. Energie
Ausgangssituation	Es ist Energie vorhanden und mehrere kombinierbare Elemente sind vorhanden.
Vorbereitungsschritte	1. App Starten.

Testschritte	**Erwartetes Resultat**
1. Mithilfe von Drag and Drop wird ein Element auf ein anderes Element verschoben	Falls diese Elemente kombinierbar sind entstehen zwei Einheiten des neuen Elementes. Die Energieleiste wird verkleinert und die vorhandene Energie sinkt.

Beschreibung	Ein Element teilen	
Abgedeckte Anwendungsfälle	3. Teilen 4. Energie	
Ausgangssituation		
Vorbereitungsschritte	1. App Starten. 2. Ein Kombiniertes Element ist vorhanden.	
Testschritte		**Erwartetes Resultat**
1. Lange auf ein kombiniertes Element tippen		Das Element wird in ein zufälliges Teilelement geteilt. Das Teilelement erscheint an derselben Position an der das vorherige Element war. Die Energieleiste wird verkleinert und die vorhandene Energie sinkt.

Beschreibung	Bei Jeder Aktion wird die Energieleiste angepasst	
Abgedeckte Anwendungsfälle	5. Energieleiste	
Ausgangssituation		
Vorbereitungsschritte	App Starten.	
Testschritte		**Erwartetes Resultat**
1. Ein Element teilen oder mit einem anderen kombinieren.		Die Energieleiste wird angepasst und zeigt einen kleineren Wert an.

Beschreibung	Elementliste	
Abgedeckte Anwendungsfälle	6. Rezepte 13. Elemente Übersicht	
Ausgangssituation	Es wurden bereits Elemente entdeckt.	
Vorbereitungsschritte	App Starten.	
Testschritte		**Erwartetes Resultat**
1. Elementliste Aufrufen.		Es erscheint eine Elementliste. In dieser sind alle Basis Elemente und alle bereits gefundenen kombinierten Elemente ersichtlich. Es sollen keine Doppeleinträge vorhanden sein.
2. Ein Element antippen		Es erscheint rechts die vollständige Beschreibung sowie mögliche Teilelemente des Elements.

Beschreibung	Spiel beenden / Game Over	
Abgedeckte Anwendungsfälle	7. Spielende	
Ausgangssituation		
Vorbereitungsschritte	App Starten.	
Testschritte		**Erwartetes Resultat**
1. Elemente kombinieren und teilen bis keine Energie mehr vorhanden ist.		Sobald keine Energie vorhanden ist, wird der Game Over Screen angezeigt.
2. „Try Again“ antippen		Das Spiel wird auf die Ausgangseinstellung zurückgesetzt. Es sind wieder die Basiselemente ersichtlich und die Energieleiste ist wieder voll.
3. Elementliste aufrufen		Die bereits entdeckten Elemente in der Elementliste sind immer noch vorhanden.

Beschreibung	Elemente Details (Popover)	
Abgedeckte Anwendungsfälle	12. Element Details	
Ausgangssituation		
Vorbereitungsschritte	App Starten.	
Testschritte		**Erwartetes Resultat**
1. Ein Element kurz antippen		Es erscheint ein Popover auf dem eine kurze, nicht immer ernstzunehmende, Beschreibung des Elements ersichtlich ist. Unter der Beschreibung sieht man ob es ein Basiselement ist. Falls es das nicht ist werden die bisher gefundenen Kombinationsmöglichkeiten angezeigt.

Einführungskonzept

Einführungsplan

Die App Unsustainable ist ein neues Produkt, deshalb gibt es keine Daten zu migrieren. Da die App auch ohne Backend funktioniert, müssen wir auch keine Serverkomponenten bereitstellen.

Die App wird lediglich im Apple Appstore und im Google Playstore veröffentlicht.

Um die App zu veröffentlichen müssen folgende Schritte gemacht werden.

1. Entwickler Account erstellen
2. Provisionierungs Profile beantragen
3. App kompilieren und signieren
4. Beschreibung und Screenshots anfertigen
5. App hochladen
6. Auf Freigabe warten
7. Eventuelle Beanstandungen der Store-Mitarbeiter korrigieren und erneut hochladen

Die App wird nach der Freigabe kostenlos zum Download bereit stehen. Zukünftige Updates werden ebenfalls kostenlos zur Verfügung gestellt.

Das Hauptrisiko bei der Einführung ist, dass die App von den Store Betreibern abgelehnt wird und noch überarbeitet werden muss.

Migrationsplan

Die Applikation ist ein neues Produkt und erfordert keine Migration von alten Daten.

Ausbildungsplan

Die Benutzer sollen die Steuerung und die Benutzung des Spiels selber erkennen und ausprobieren. Im Zweifelsfall kann ein Benutzerhandbuch in Form einer Wiki Seite auf Github publiziert werden.

Die Wiki Seite beschreibt wie man Elemente kombinieren kann und wie man diese aufteilen kann. Es wird ebenfalls eine versteckte Liste mit allen Kombinationen geben.

Testprotokoll

Getestete Version: 1.0

Tester: Elias Schmidhalter, Michael Günter

Datum, Zeit: 19.05.2015, 09:20 bis 10:00 Uhr

Testfall 1: Drag and Drop Elemente können verschoben werden

Testschritt	Erfüllt	Bemerkung
1.	☑	
2.	☑	

Testfall 2: Zwei Elemente kombinieren

Testschritt	Erfüllt	Bemerkung
1.	☑	

Testfall 3: Elemente teilen

Testschritt	Erfüllt	Bemerkung
1.	☑	

Testfall 4: Bei Jeder Aktion wird die Energieleiste angepasst

Testschritt	Erfüllt	Bemerkung
1.	☑	

Testfall 5: Elementliste

Testschritt	Erfüllt	Bemerkung
1.	☑	
2.	☑	

Testfall 6: Spiel beenden / Game Over

Testschritt	Erfüllt	Bemerkung
1.	☑	
2.	☑	
3.	☑	

Testfall 7: Elemente Details (Popover)

Testschritt	Erfüllt	Bemerkung
1.	☑	

Benutzerdokumentation

Anwendungshandbuch

Das Anwendungshandbuch bietet Ihnen eine Übersicht über die wichtigsten Funktionen der Unsustainable-App. Wenn Sie Informationen zur Installation der Applikation benötigen, befolgen Sie bitte die Schritte in dem Abschnitt „Itegrations- und Installationshandbuch“.

Übersicht

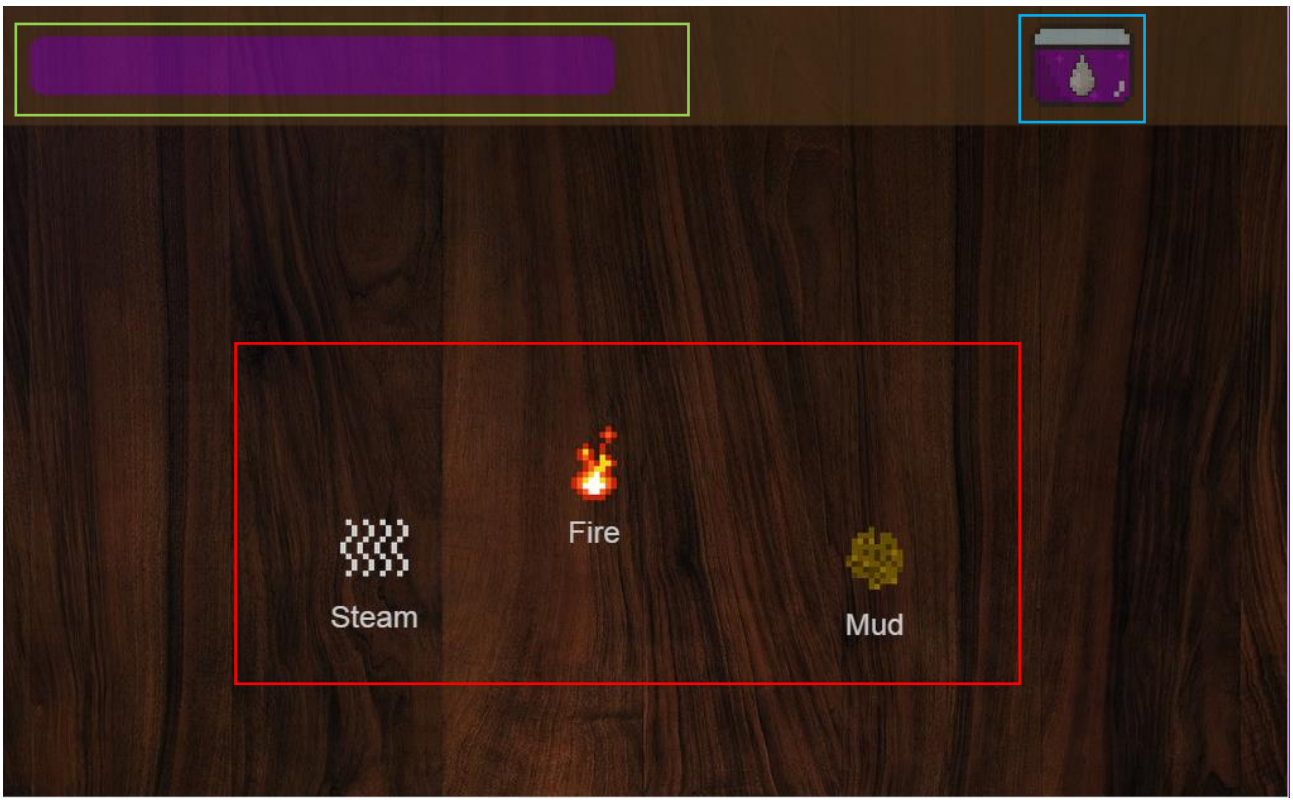
Die Unsustainable-App ist ein Puzzle Game, bei dem es darum geht, verschiedene Elemente zu kombinieren um neue Elemente zu erschaffen, die wiederum kombiniert werden können. Das Ziel des Spiels ist es, so viele Elemente wie möglich zu finden und durch austesten verschiedener Kombinationen freizuschalten. Elemente können durch simples Verschieben kombiniert werden.

In den folgenden Beschreibungen werden wir Ihnen einen groben Überblick über den Aufbau der Applikation und über die groben Funktionalitäten verschaffen.

Funktionen und Detailbeschreibung

Zuerst sollten Sie sich einen Überblick über den Aufbau der Applikation verschaffen.

Betrachten Sie folgende Abbildung, die den sogenannten „Alchemy Table“ darstellt.

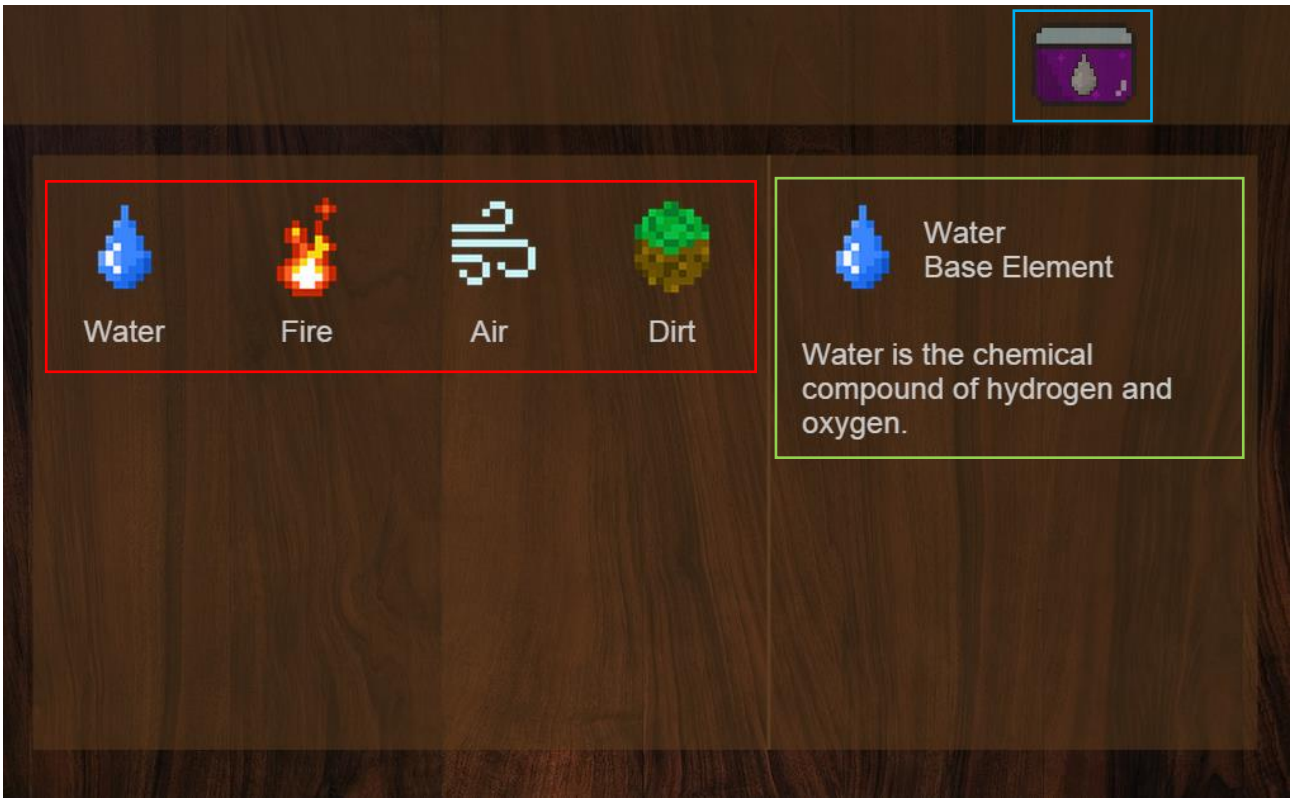


Sie sehen, dass sich auf dem Alchemy Table verschiedene Elemente befinden (rot markiert). Diese Elemente können mittels Berühren und Ziehen des Fingers über den Touchscreen bewegt werden.

Am oberen Bildschirmrand (grün markiert) sehen Sie die Energiebar. Sie haben nicht unbegrenzt viel Energie zur Verfügung. Jede Interaktion mit dem Elemente (Kombinieren, Teilen, etc.) kostet Sie Energie. Wenn die Energie zur Neige geht, ist das Spiel zu Ende, und wird neu gestartet. Sie erhalten, dann wiederum die Ausgangselemente.

Ausserdem finden Sie oben rechts die Schaltfläche um auf die Element-Liste zu gelangen. Bei Berührung der Schaltfläche wechselt die Applikation in die Elementlisten-Ansicht.

Die folgende Abbildung zeigt die Elementliste.



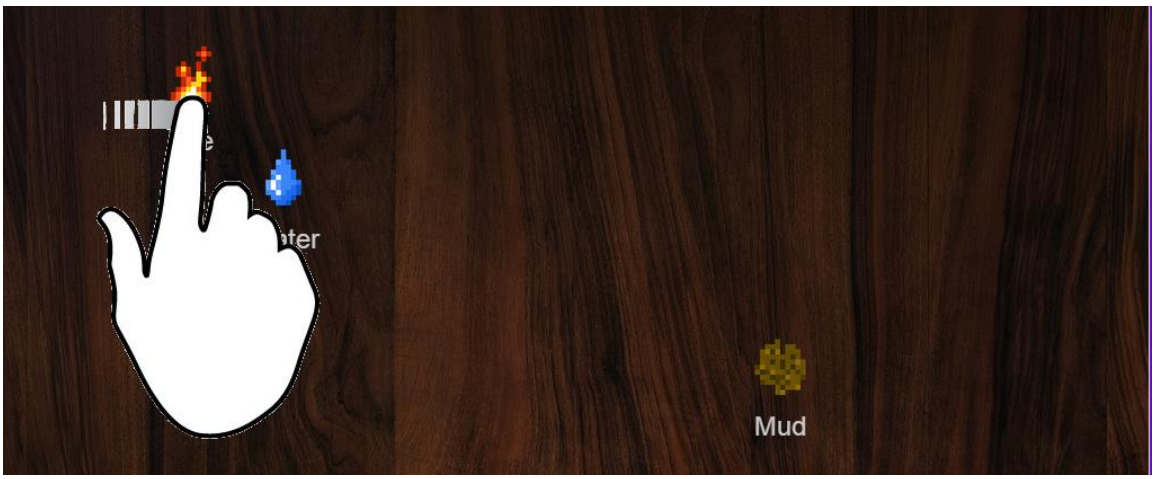
Auf der linken Seite sehen Sie alle Elemente (rot markiert), die Sie bisher gefunden haben. Mittels einer Berührung auf ein Element erhalten Sie auf der rechten Seite (grün markiert) detaillierte Informationen über das ausgewählte Element. Es ist auch ersichtlich wieviel Energie das Element benötigt hat, damit es hergestellt werden konnte und mit welchen Elementen es hergestellt wurde.

Mithilfe des blau markierten Buttons gelangen Sie wieder zurück zum Alchemy Table und können dort weitermachen, wo sie aufgehört haben.

Das Spiel speichert an jeder Stelle automatisch, es sind keine weiteren Aktionen für das Speichern notwendig.

Kombinieren von Elementen

Um Elemente zu kombinieren stellen Sie zunächst sicher, dass Sie sich auf dem Alchemy Table befinden. Berühren Sie anschliessend mit ihrem Finger ein Element und ziehen Sie es auf ein anderes Element. Wenn für die beiden Elemente eine Kombination existiert, verwandeln sich die beiden Elemente in zwei neue Elemente.



Aufteilen von Elementen

Da Sie beim Kombinieren von Elementen wiederum zwei Elemente erhalten, haben Sie immer gleichviel Elemente auf dem Alchemy Table. Daher braucht es eine Möglichkeit, die Elemente die Sie für die Kombination gebraucht haben wieder zurückzuerlangen. Sie können das jeweilige kombinierte Element wieder aufteilen. Beim Aufteilen erhalten Sie wiederum eines der beiden Elemente, die Sie beim Kombinieren gebraucht haben. Welches Element Sie erhalten, hängt vom Zufall ab.

Um ein Element aufzuteilen, berühren Sie es lange und lassen Sie das Element nicht los. Bewegen Sie es nicht. Das Element beginnt sich zu schütteln. Danach teilt es sich und Sie erhalten eines der Kombinationselemente.

Anzeigen von Informationen

Einige Informationen zu dem Element sind nicht nur in der bereits beschriebenen Elementliste ersichtlich, sondern Sie können die Informationen auch während des Spiels auf dem Alchemy Table abrufen.

Tippen Sie dazu das gewünschte Element kurz an, um ein kleines Informationsfenster zu öffnen.

Fehlerbehandlung

Elemente lassen sich nicht aufteilen.

Um Elemente aufzuteilen, müssen Sie beachten, dass Sie das Element lange berühren und nicht loslassen. Bitte nehmen Sie zur Kenntnis, dass nicht alle Elemente geteilt werden können. Die Grundelemente können nicht geteilt werden und stattdessen können aus Grundelementen andere Grundelemente erschaffen werden.

Die Applikation lässt sich auf meinem Android Telefon nicht spielen / installieren.

Für die Ausführung der App wird mindestens Android 5.0 vorausgesetzt. Bitte stellen Sie sicher, dass die Software auf Ihrem Telefon mindestens die Version 5.0 hat. Sie müssen gegebenenfalls ein Software-Update ausführen. Versuchen Sie anschliessend die App erneut zu installieren und zu starten.

Elemente lassen sich nicht bewegen

Wenn sich die Elemente nicht bewegen lassen, haben Sie es möglicherweise zustande gebracht, die Applikation auf einem veralteten Android Telefon zu installieren. Bitte überprüfen Sie, ob Sie mindestens Android 5.0 installiert haben und führen Sie gegebenenfalls ein Software-Update aus. Installieren Sie danach die App erneut und versuchen Sie es danach nochmal.

Die App startet nicht mehr / Der Alchemy Table ist leer

Ihre Spieldaten sind möglicherweise korrupt. Leider existiert kein Repair-Tool. Bitte deinstallieren Sie die Applikation und stellen Sie sicher, dass alle Daten gelöscht sind. Installieren Sie danach die App nochmal und versuchen Sie erneut.

Integrations- und Betriebshandbuch

Um die App auf Android oder iOS zu installieren, rufen Sie bitte den entsprechenden App-Store auf und suchen Sie nach „Unsustainable“. Sie sollten anschliessend unsere Applikation finden und diese mittels des „Installieren“ Buttons installieren können.

Bitte beachten Sie, dass für Android mindestens die Version 5.0 vorausgesetzt wird.

Betriebshandbuch

Es sind keine weiteren Vorkehrungen für die Sicherstellung des Betriebs notwendig.

Projekterfahrung

Das Projekt ist relativ gut verlaufen. Wir hatten jedoch für die Umsetzung einiger Features zu wenig Zeit. Dies waren alle Features die mit den Achievements zusammenhängen. Wir haben dadurch zwar nicht alle Anforderungen abgedeckt, jedoch haben wir durchaus alle unsere definierten Ziele erreicht.

Wir haben vor allem Erfahrungen mit Apache Cordova gesammelt, da unsere gesamte Applikation damit umgesetzt wurde. Was nicht so gut gelaufen ist, ist die Einbindung der Datenzugriffe, da wir damit lange herumexperimentieren mussten. Ausserdem konnten wir nicht testdriven vorgehen, wie das ursprünglich geplant war. Viele der Code Abschnitte konnten nicht mit Unit-Tests getestet werden, da diese einen Datenzugriff voraussetzen und dieser vom Testrunner nicht gewährleistet wird. Was jedoch gut gelaufen ist, ist die Tatsache, dass wir sehr schnell und einfach eine App entwickeln konnten. Dies verdanken wir der Nutzung von Apache Cordova.

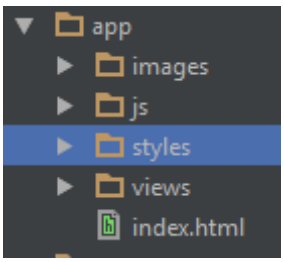
In Zukunft kann man durchaus wieder mit Cordova Arbeiten, aber einfach nicht testdriven und auch nicht unbedingt mit Datenzugriffen.

Teil 3

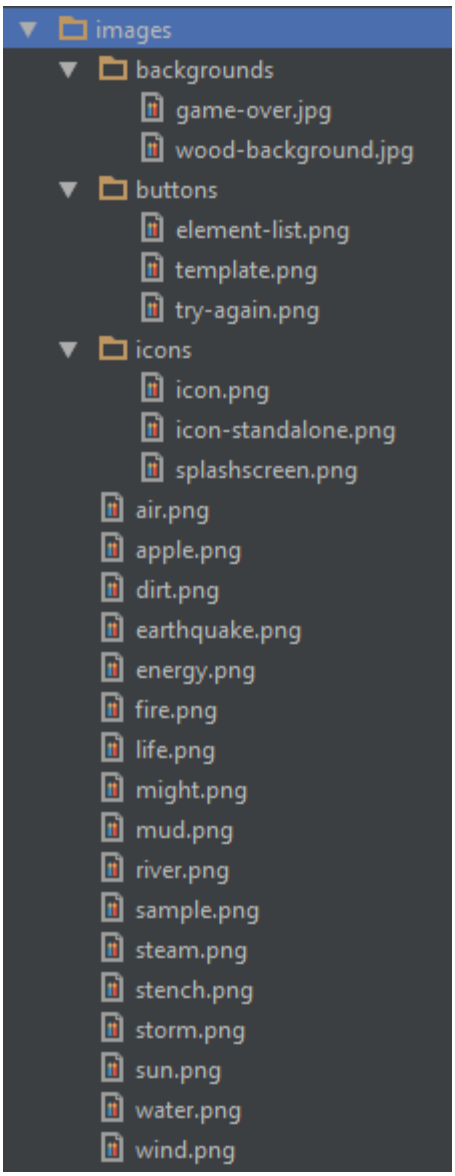
Selber erstellte Listings und Skripte

Im Folgenden werden die ersten 5 Seiten des Quellcodes der App festgehalten. Hinweis: Die Achievements wurden nicht realisiert.

Übersicht

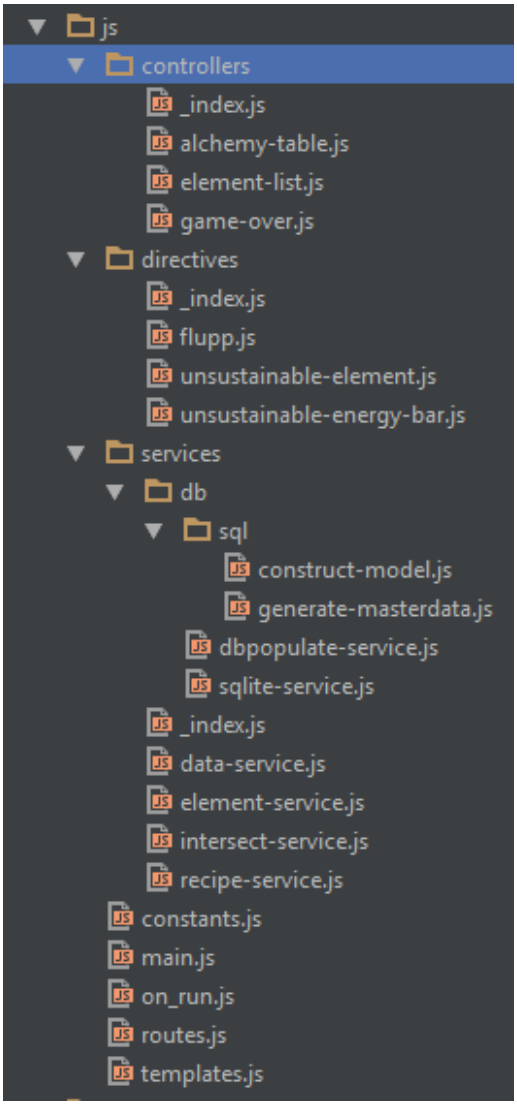


Images

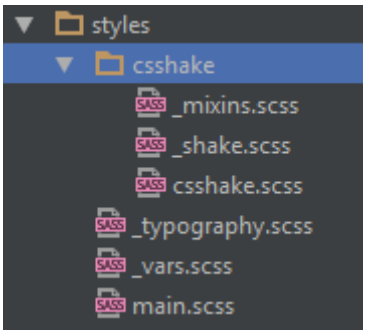


Hinweis: Im Moment sind noch nicht alle Bilder vorhanden. Deswegen existiert das Sample Bild. Später werden wir die Bilder noch vervollständigen.

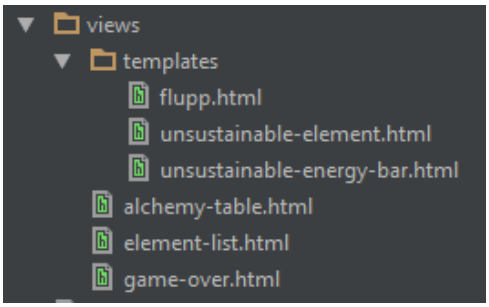
JS



Styles



Views


views
templates
flupp.html
unsustainable-element.html
unsustainable-energy-bar.html
alchemy-table.html
element-list.html
game-over.html

index.html

```
<!doctype html>

<html class="no-js">

  <head>

      <base href=".">

      <meta charset="utf-8">

      <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

      <title ng-bind="pageTitle"></title>

       <meta name="format-detection" content="telephone=no" />

      <meta name="msapplication-tap-highlight" content="no" />

      <meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1,
minimum-scale=1, width=device-width" />

      <link rel="stylesheet" href="css/csshake/csshake.css">

      <link rel="stylesheet" href="css/main.css">

  </head>

  <body>

      <div class="uns-main-view" ui-view=""></div>

      <script src="cordova.js"></script>

      <script src="js/main.js"></script>

  </body>

</html>
```

js/main.js

```
'use strict';

var angular = require('angular');

// angular modules
require('angular-ui-router');
require('angular-touch');
require('./templates');
require('./controllers/_index');
require('./services/_index');
require('./directives/_index');

// create and bootstrap application
angular.element(document).ready(function () {

    var requires = [
        'ui.router',
        'ngTouch',
        'templates',
        'app.controllers',
        'app.services',
        'app.directives'
    ];

    // mount on window for testing
    window.app = angular.module('app', requires);

    angular.module('app').constant('AppSettings', require('./constants'));


    angular.module('app').config(require('./routes'));
```

```
    angular.module('app').run(require('./on_run'));


    if (!!window.cordova) {
        document.addEventListener('deviceready', bootstrap, false);
    } else {
        bootstrap();
    }


    function bootstrap() {
        angular.bootstrap(document, ['app']);
    }
});
```

js/on_run.js

```
'use strict';


/**
 * @ngInject
 */
function OnRun($rootScope, AppSettings) {


  // change page title based on state
  $rootScope.$on('$stateChangeSuccess', function(event, toState) {
    $rootScope.pageTitle = '';


    if ( toState.title ) {
      $rootScope.pageTitle += toState.title;
      $rootScope.pageTitle += ' \u2014 ';
    }


    $rootScope.pageTitle += AppSettings.appTitle;
  });


}


module.exports = OnRun;
```

js/constants.js

```
'use strict';


var AppSettings = {
    appTitle: 'Example Application',
    apiUrl: '/api/v1',
    dbName: "unsustainable.db"
};


module.exports = AppSettings;
```

js/routes.js

```
'use strict';


/**
 * @ngInject
 */
function Routes($stateProvider, $locationProvider, $urlRouterProvider) {
    $locationProvider.html5Mode(false);


    $stateProvider
        .state('alchemyTable', {
            url: '/',
            controller: 'alchemyTableCtrl as vm',
            templateUrl: 'alchemy-table.html'
        })
        .state('elementList',{
          url:'/element-list',
            controller:'elementListCtrl as vm',
            templateUrl:'element-list.html'
        })
        .state('gameOver', {
            url: '/game-over',
            controller: 'gameOverCtrl as vm',
            templateUrl: 'game-over.html'
        });


    $urlRouterProvider.otherwise('/');
}


module.exports = Routes;
```

Literaturverzeichnis

- AngularJS: http://angularjs.org
- Browserify Gulp Angularjs Boilerplate: https://github.com/jakemmarsh/angularjs-gulp-browserify-boilerplate
- Apache Cordova : http://cordova.apache.org/
- Little Alchemy : http://littlealchemy.com/

Glossar

- Kombinieren: Das kombinieren zweier Elemente um ein neues Element zu erschaffen.
- Aufteilen: Das lange Berühren eines Elements, um ein zufälliges Element zu erhalten, das beim Kombinieren benötigt wurde.
- Promise: Ein Objekt, das verspricht, in der nahen Zukunft eine Antwort zurückzuliefern.
- Test-Driven: Test-Driven Development ist, wenn man zuerst die Anforderung in einem Test umsetzt und danach die Funktion implementiert und verbessert bis der Tests als erfolgreich markiert wird.
- Achievement: Ein Erfolg, der freigeschaltet wird, wenn ein bestimmtes Ereignis eintritt.