# Realisierungsbericht

| | |
|---|---|
| **Status** | Abgeschlossen |
| **Projektname** | Unsustainable |
| **Projektleiter** | Michael Günter, Elias Schmidhalter |
| **Auftraggeber** | Andres Scheidegger |
| **Autoren** | Michael Günter, Elias Schmidhalter |
| **Verteiler** | Michael Günter, Elias Schmidhalter, Andres Scheidegger |

**Änderungskontrolle, Prüfung, Genehmigung**

| Version | Datum | Beschreibung, Bemerkung | Name oder Rolle |
|---|---|---|---|
| 0.1 | 27.04.2015 | Erstfassung | Michael Günter |

**Inhaltsverzeichnis**

1 Zusammenfassung....3
2 Technische Detailspezifikation....3
2.1 Systemdesign....3
2.1.1 Struktur....3
2.1.2 Dynamik....10
2.1.3 Promises....11
2.2 Schnittstellendefinitionen....11
2.3 Datenmodell....12
3 Systemdokumentation....13
3.1 Anwendungshandbuch....13
3.1.1 Übersicht....13
3.1.2 Funktionen und Detailbeschreibung....13
3.1.3 Fehlerbehandlung....16
3.2 Integrations- und Installationshandbuch....16
3.3 Betriebshandbuch....16
3.4 Systemtest....17
3.4.1 Testfälle....17
4 Testprotokoll....19
4.1 Systemtest 1....19
4.1.1 Testfall 1: Drag and Drop Elemente können verschoben werden....19
4.1.2 Testfall 2: Zwei Elemente kombinieren....19
4.1.3 Testfall 3: Elemente teilen....19
4.1.4 Testfall 4: Bei Jeder Aktion wird die Energieleiste angepasst....19
4.1.5 Testfall 5: Elementliste....19
4.1.6 Testfall 6: Spiel beenden / Game Over....19
4.1.7 Testfall 7: Elemente Details (Popover)....19
6 Weiterführung der Projektplanung....19
6.1 Abgleich von Planung und tatsächlichem Verlauf der Phase Konzept....19
6.2 Aktualisierung der Risikosituation....19
6.3 Planung der nächsten Phase....19
Anhang A: Quellcode....20
Anhang B: Testcode Unit-Tests....75

# 1 Zusammenfassung

Der Realisierungsbericht dient zur Dokumentation der grundsätzlichen technischen Struktur der Applikation. Auch Informationen zur Benutzerführung und zur Applikationsinstallation, sowie Informationen zum gesamten Testprozess werden in diesem Dokument festgehalten.

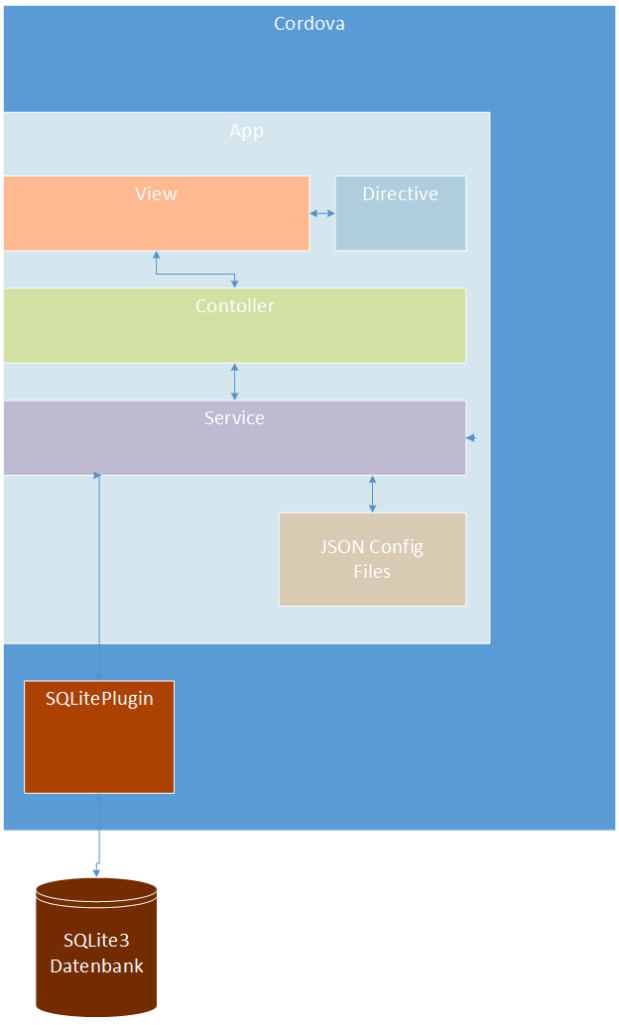
# 2 Technische Detailspezifikation

## 2.1 Systemdesign

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

### 2.1.1 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.

Die Grafik links, ist jedoch nicht mehr dieselbe wie im Konzeptbericht. Es fällt auf, dass zwischen den Services und dem SQLitePlugin kein ngCordova mehr ist. Wir haben uns während der Entwicklung dazu entschlossen ngCordova wegzulassen, weil wir es eigentlich nur für das SQLitePlugin brauchen und wir dies viel einfacher selber von Hand einbinden können, indem wir einen SQLiteService schreiben, der mit dem Plugin interoperiert und ein besseres API zur Verfügung stellt.

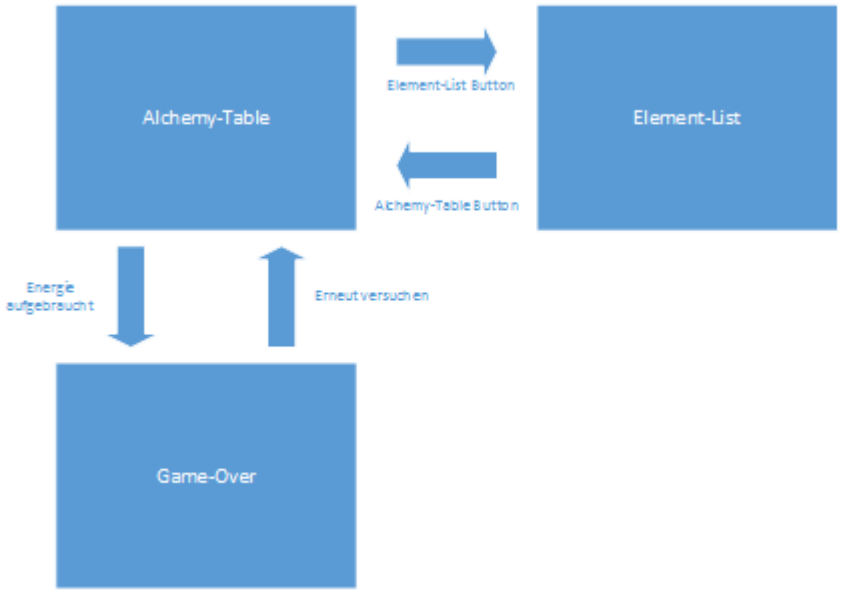
**View**
Eine View ist ein HTML-Dokument, welches verschiedene UI Komponenten anordnet und grafisch aufbereitet darstellt.
Durch die Benutzung von AngularJS haben wir die Möglichkeit direkt in den Views bestimmte Aktionen durchzuführen.

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.

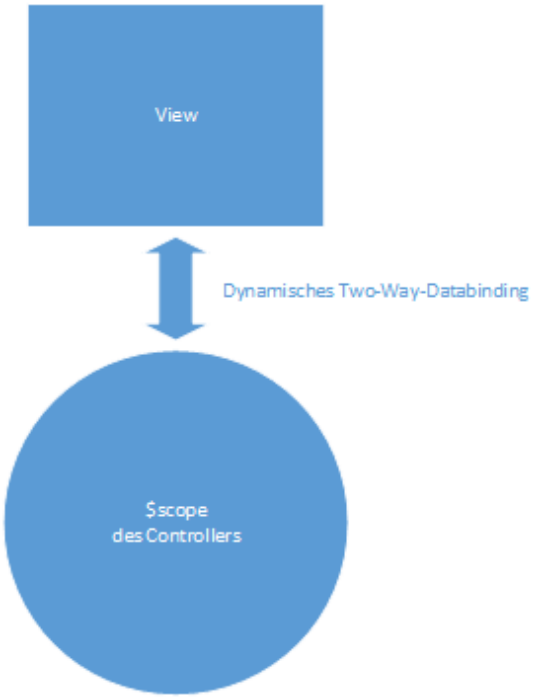
Wir haben genau 3 Views. Einmal der normale Alchemy Table, einmal die Element-Liste und einmal der Game-Over Screen. Zu den Views zählen technisch gesehen auch die Templates für die Directives. Diese werden jedoch später genauer beschrieben.

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.

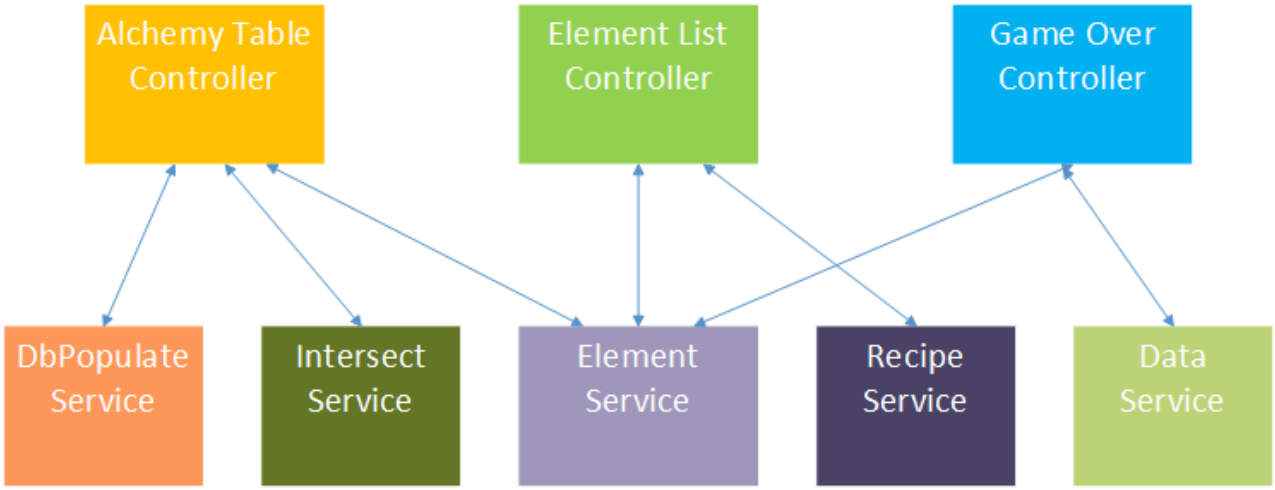



Aber das Two-Way-Databinding wirkt keineswegs nur zwischen View und Controller, sondern auch innerhalb des Controllers und innerhalb der View, wenn zum Beispiel an zwei Stellen auf dieselbe Variable zugegriffen wird. Auch die Argumente für Directives sind nach Wahl ein Databinding und ändern so direkt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

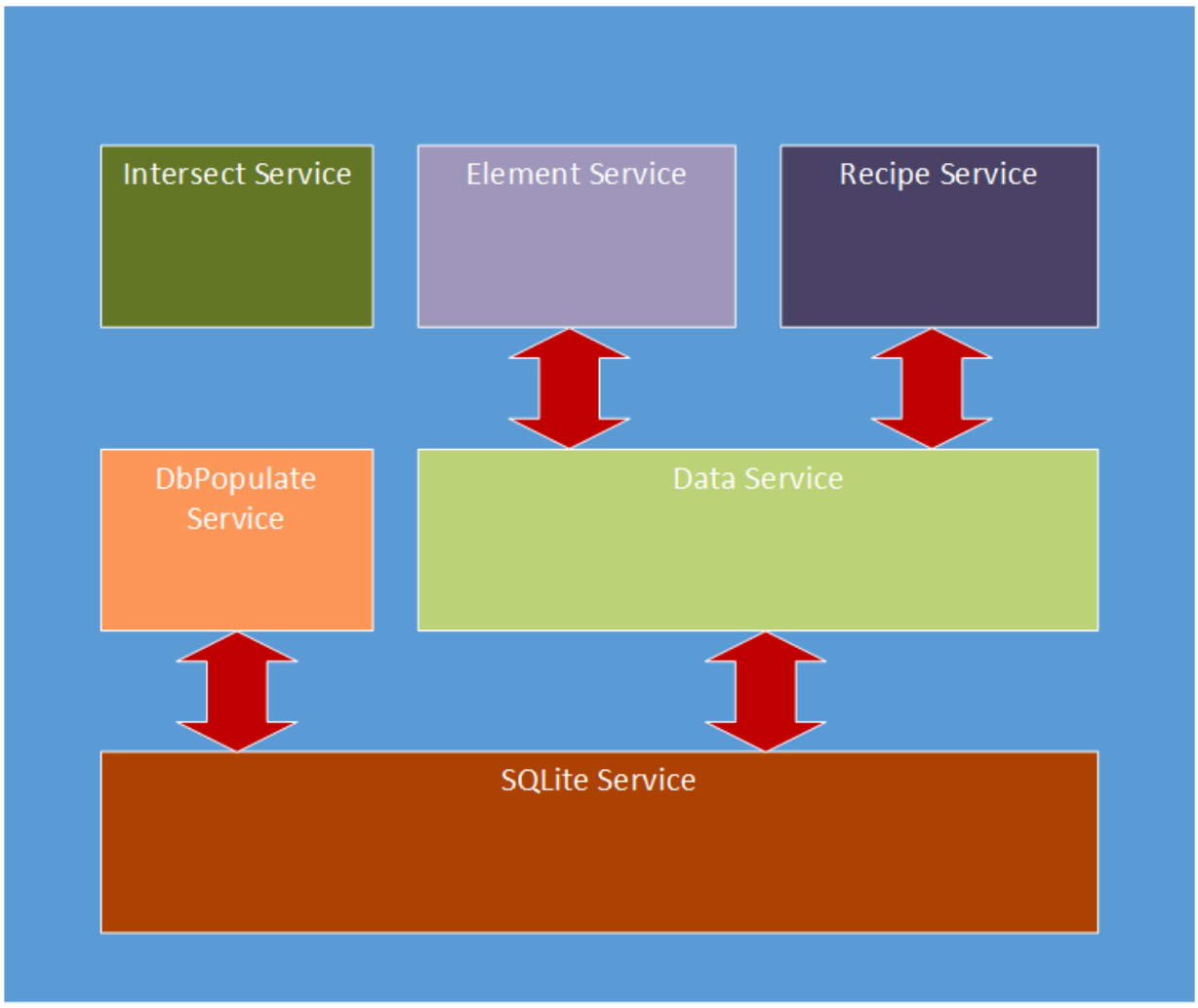
Der Dataservice erstellt anhand von SQL Abfragen Business Entitäten (JSON Objects).
In diesem Service sind die SQL-Statements abgelegt. Der Dataservice verwendet den SQLite Service um die abgelegten Queries auf der Datenbank auszuführen.

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.

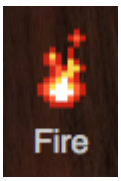


**Directive**
Direktiven sind zusammengefasste UI-Komponenten. Für dieses Projekt haben wir insgesamt drei verschiedene Direktiven erstellt.

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.

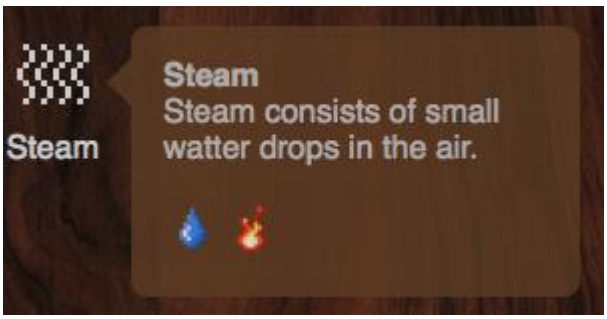
*Element Directive*




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.

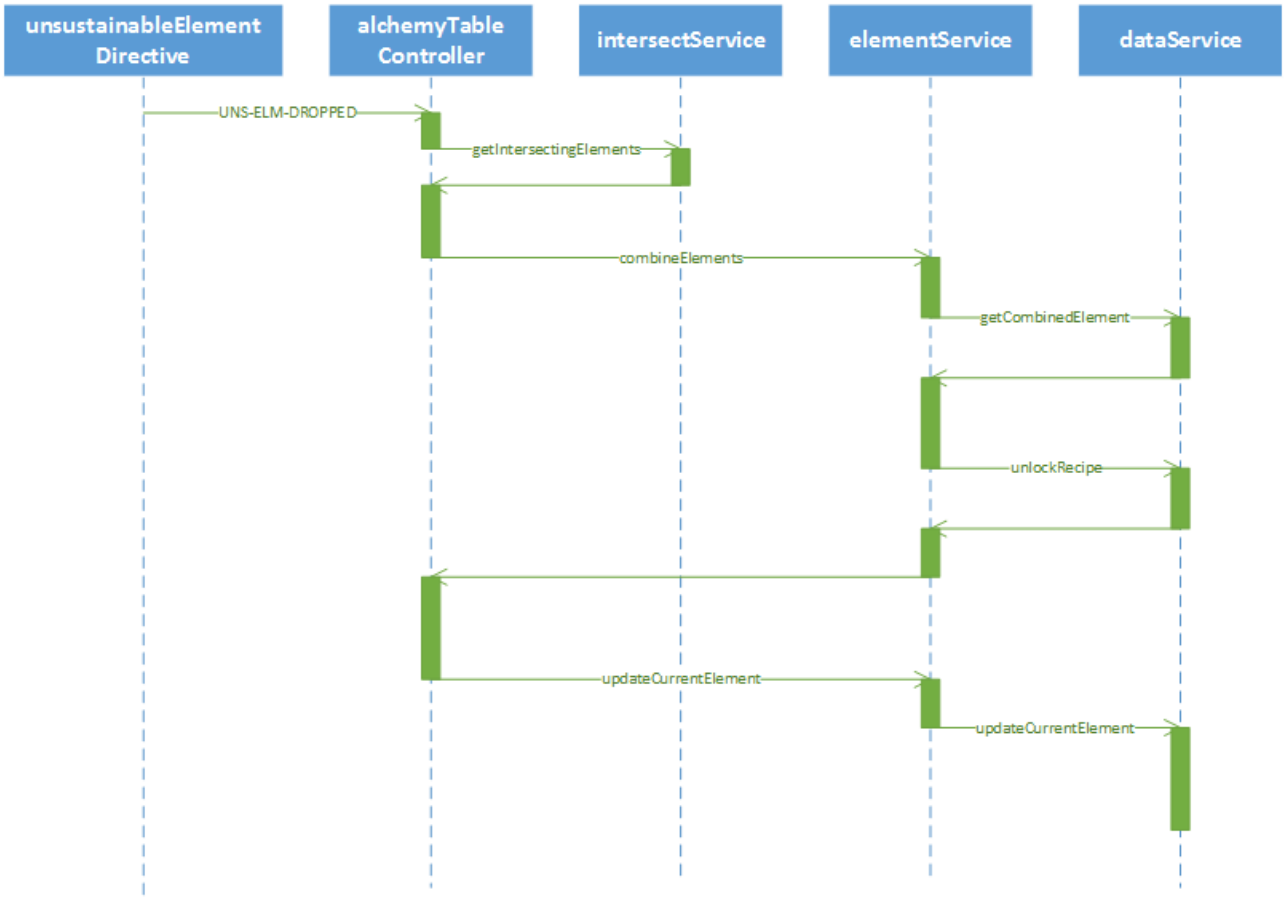
*Routen*
Die Routen befinden sich im routes.js und definieren für jede View einen Controller und unter welchem State diese erreichbar sind.

*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.

### 2.1.2 Dynamik

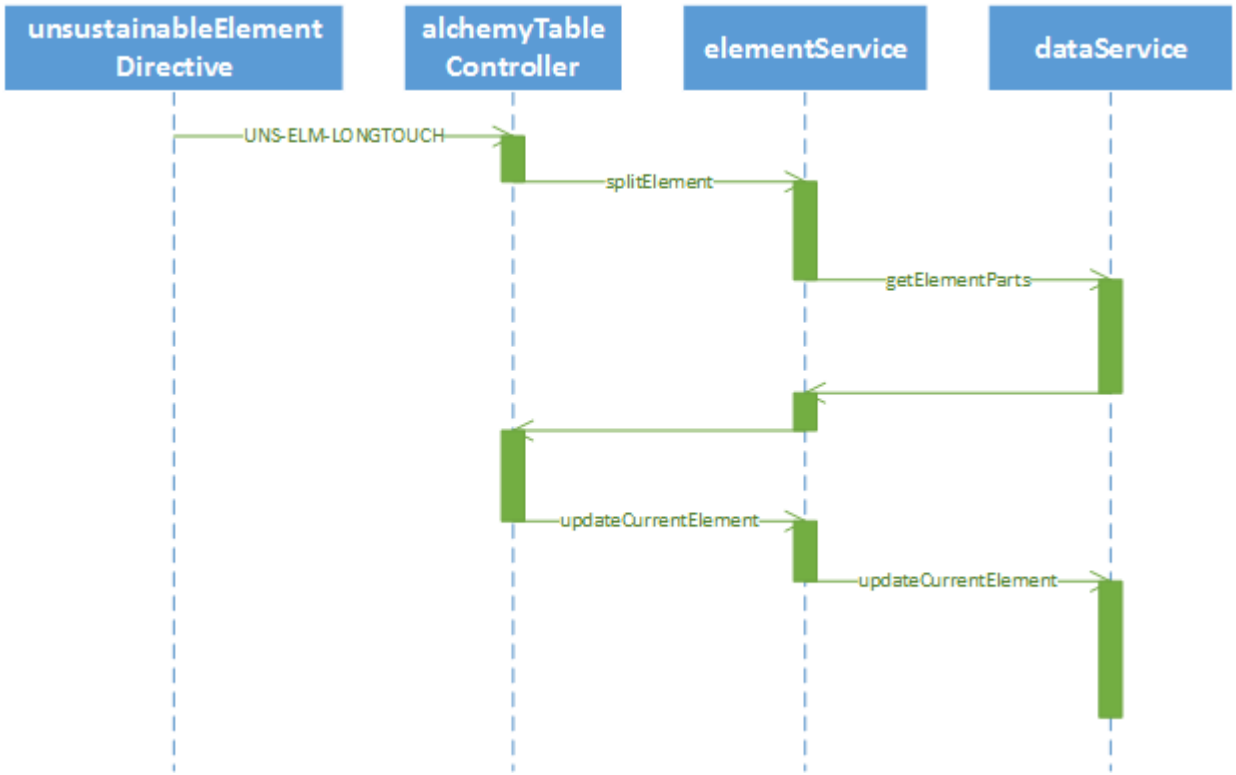
In diesem Abschnitt geht es darum die Dynamik der Applikation zu verständlichen. Dazu werden zwei wichtige Prozesse der Dynamik beschrieben. Einerseits das Kombinieren und andererseits das Aufteilen von Elementen.




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.

### 2.1.3 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.

## 2.2 Schnittstellendefinitionen

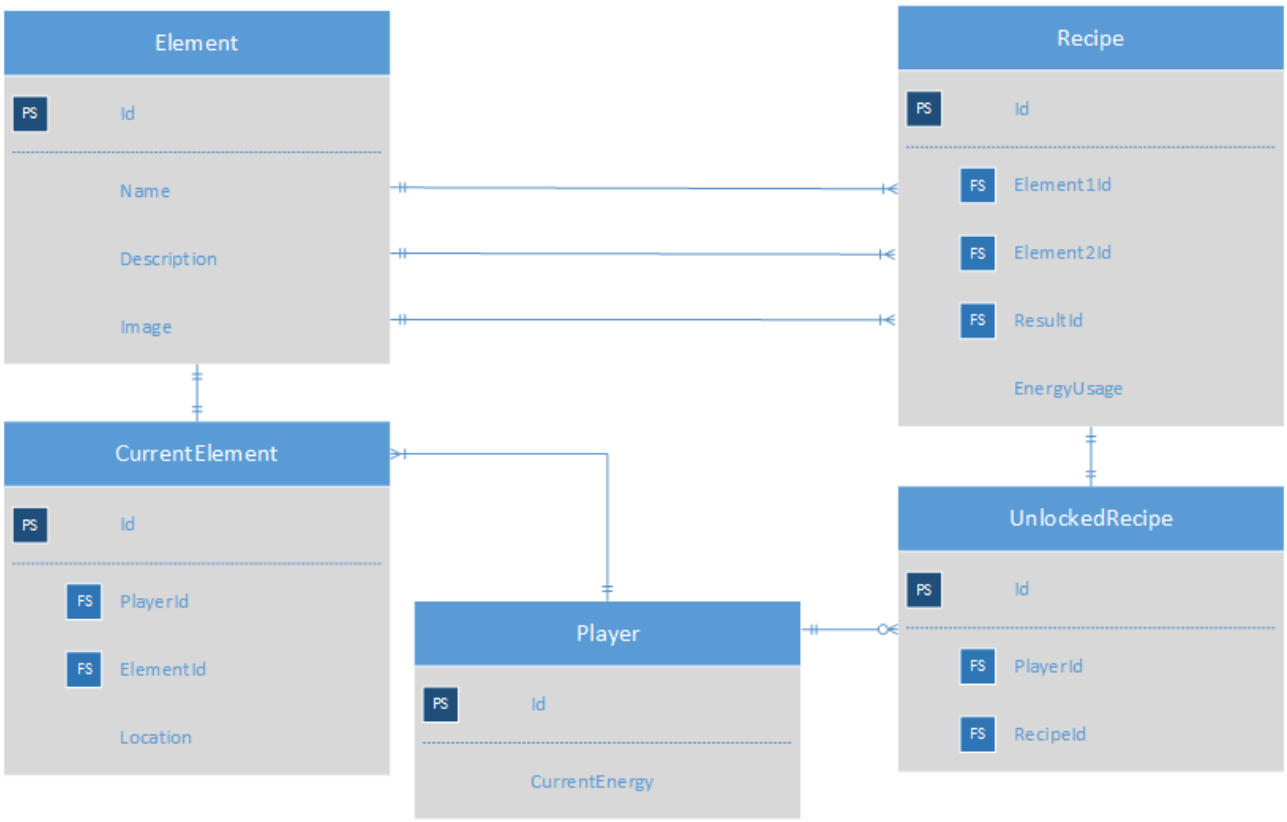
Wir haben eine Schnittstelle für die Persistenzschicht. Diese verwenden wir um Daten zu holen und zu speichern.

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.

## 2.3 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.

# 3 Systemdokumentation

Die Systemdokumentation beschreibt das Anwendungshandbuch, die Integrations- und Installationsanleitung sowie das Betriebshandbuch der Unsustainable-App.

## 3.1 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“.

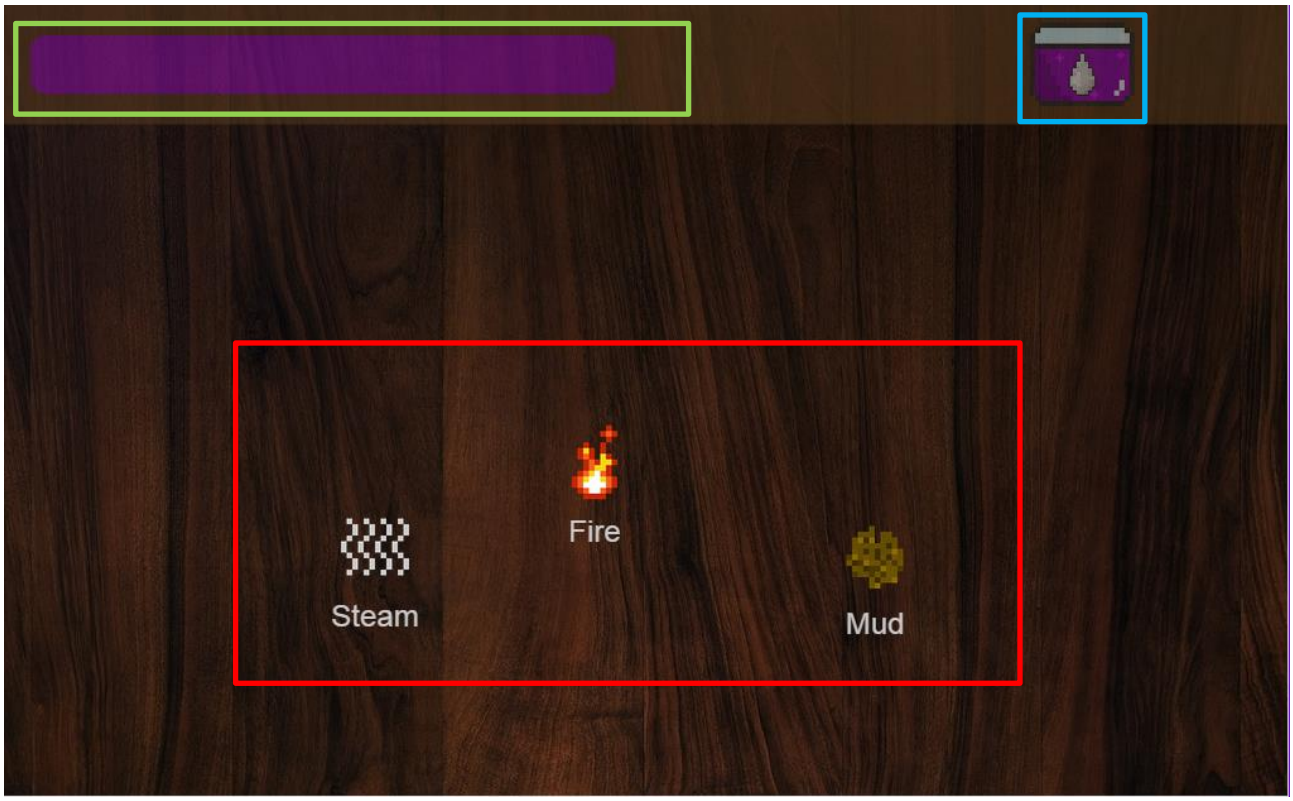
### 3.1.1 Übersicht

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.

### 3.1.2 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.

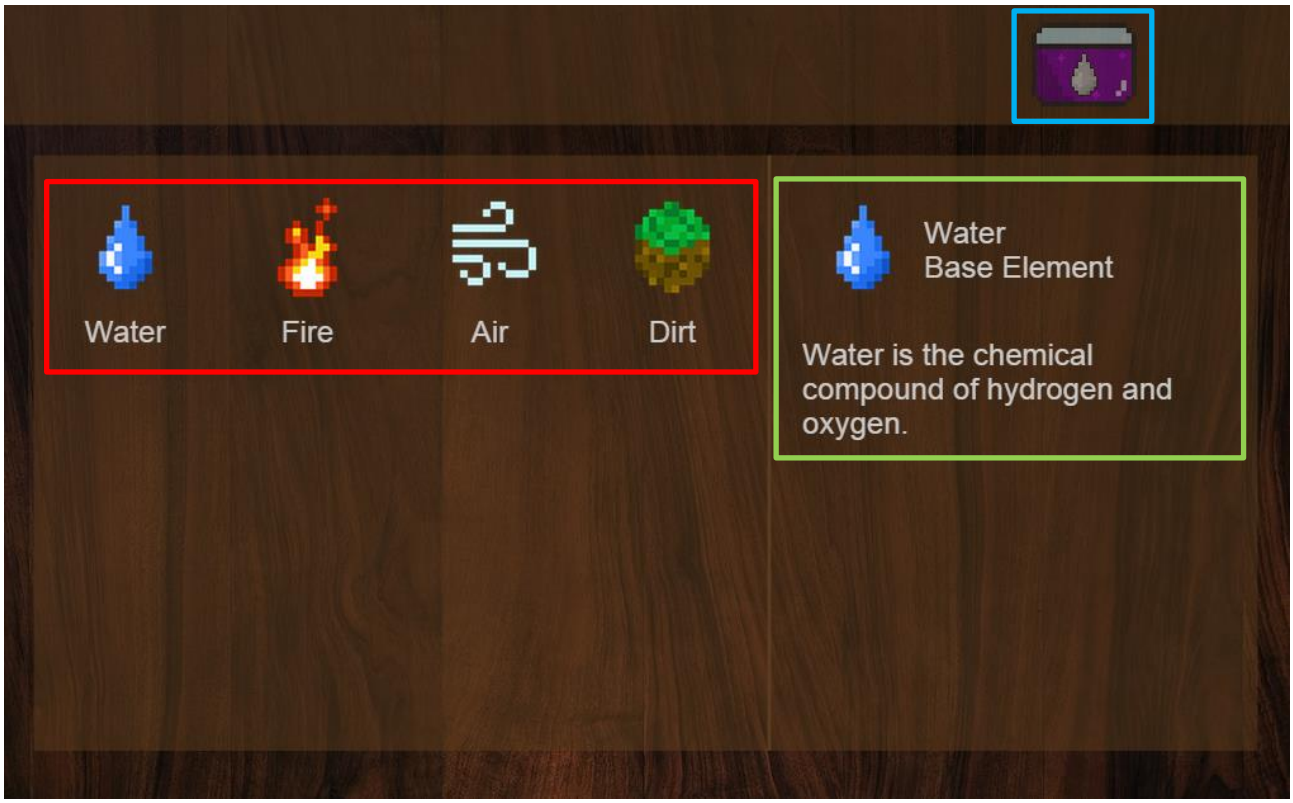



Sie sehen, dass sich auf dem Alchemy Table verschiedene Elemente befinden (rot markiert). Diese Elemente können mittels Berühren und Ziehen des Fingers über den Touchscreen bewegt werden.

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.

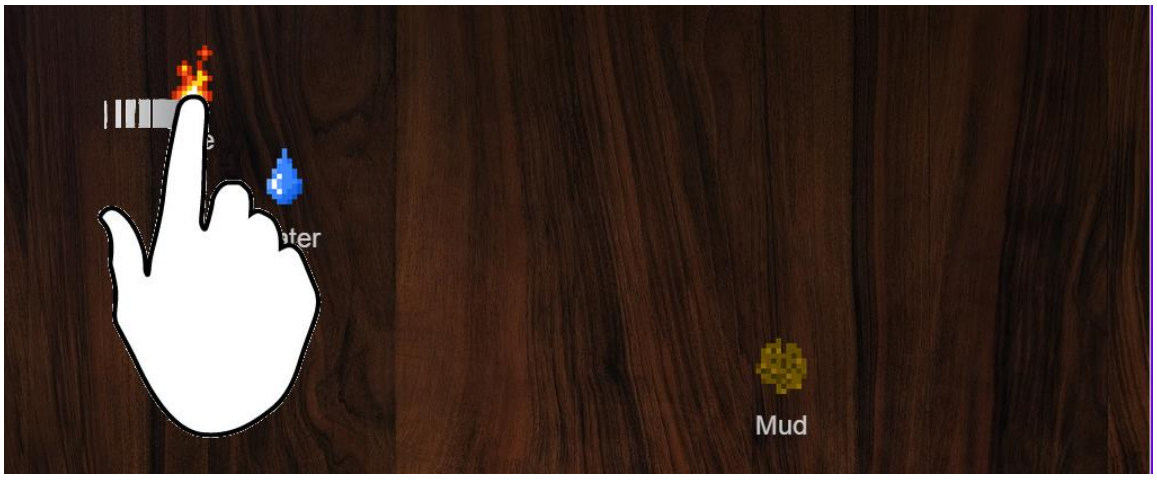



Auf der linken Seite sehen Sie alle Elemente (rot markiert), die Sie bisher gefunden haben. Mittels einer Berührung auf ein Element erhalten Sie auf der rechten Seite (grün markiert) detaillierte Informationen über das ausgewählte Element. Es ist auch ersichtlich wieviel Energie das Element benötigt 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.

3.1.3 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.

## 3.2 Integrations- und Installationshandbuch

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.

## 3.3 Betriebshandbuch

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

## 3.4 Systemtest

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

### 3.4.1 Testfälle

| **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<br>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.<br>Die Energieleiste wird verkleinert und die vorhandene Energie sinkt. |

| **Beschreibung** | Ein Element teilen | |
|---|---|---|
| **Abgedeckte Anwendungsfälle** | 3. Teilen<br>4. Energie | |
| **Ausgangssituation** | | |
| **Vorbereitungsschritte** | 1. App Starten.<br>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.<br>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<br>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. |

# 4 Testprotokoll

## 4.1 Systemtest 1

Getestete Version: 1.0
Tester: Michael Günter
Datum, Zeit: 10.05.2015, 10:00 bis 11:00 Uhr

4.1.1 Testfall 1: Drag and Drop Elemente können verschoben werden

| Testschritt | Erfüllt | Bemerkung |
|---|---|---|
| 1. | ☑ | |
| 2. | ☑ | |

4.1.2 Testfall 2: Zwei Elemente kombinieren

| Testschritt | Erfüllt | Bemerkung |
|---|---|---|
| 1. | ☑ | |

4.1.3 Testfall 3: Elemente teilen

| Testschritt | Erfüllt | Bemerkung |
|---|---|---|
| 1. | ☑ | |

4.1.4 Testfall 4: Bei Jeder Aktion wird die Energieleiste angepasst

| Testschritt | Erfüllt | Bemerkung |
|---|---|---|
| 1. | ☑ | |

4.1.5 Testfall 5: Elementliste

| Testschritt | Erfüllt | Bemerkung |
|---|---|---|
| 1. | ☑ | |
| 2. | ☑ | |

4.1.6 Testfall 6: Spiel beenden / Game Over

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

4.1.7 Testfall 7: Elemente Details (Popover)

| Testschritt | Erfüllt | Bemerkung |
|---|---|---|
| 1. | ☑ | |

# 6 Weiterführung der Projektplanung

## 6.1 Abgleich von Planung und tatsächlichem Verlauf der Phase Konzept

Der Soll-Ist Vergleich befindet sich im Anhang im Zeitplan (Gantt-Diagramm)

## 6.2 Aktualisierung der Risikosituation

An der Risiko-Situation hat sich soweit nichts verändert.

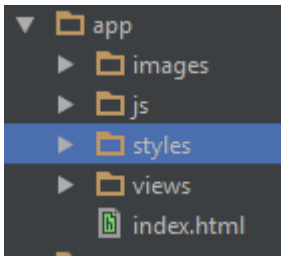
## 6.3 Planung der nächsten Phase

Die Planung für die nächste Phase befindet sich im Zeitplan. (Gantt-Diagramm).

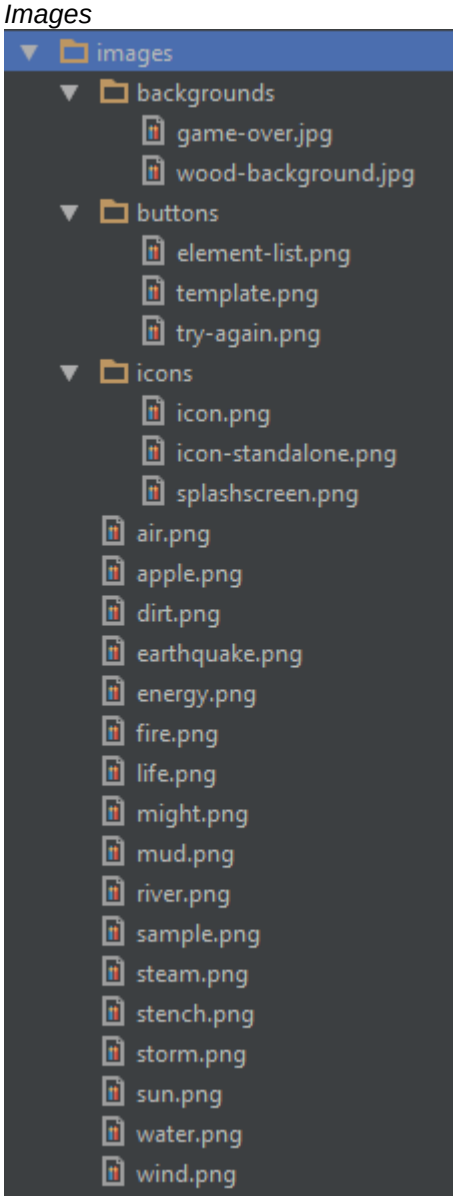
## Anhang A: Quellcode

Im Folgenden wird der Quellcode der App festgehalten. Hinweis: Die Achievements wurden nicht realisiert.

**Übersicht**

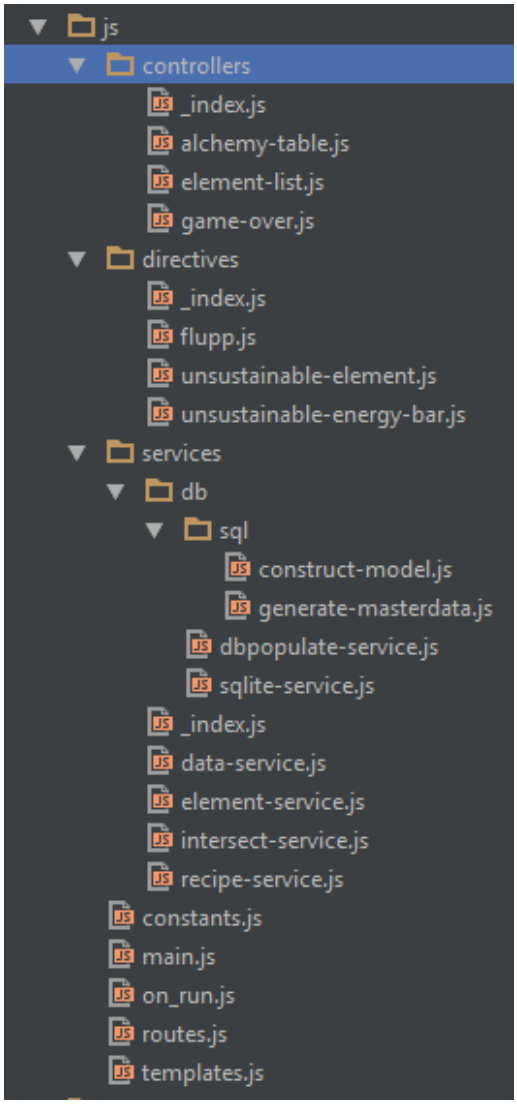



*Images*

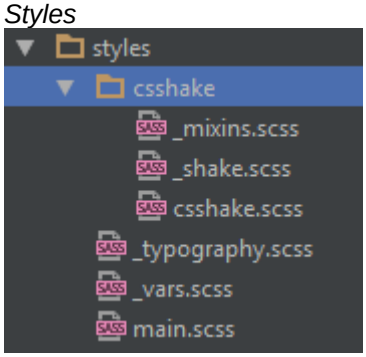



Hinweis: Im Moment sind noch nicht alle Bilder vorhanden. Deswegen existiert das Sample Bild. Später werden wir die Bilder noch vervollständigen.

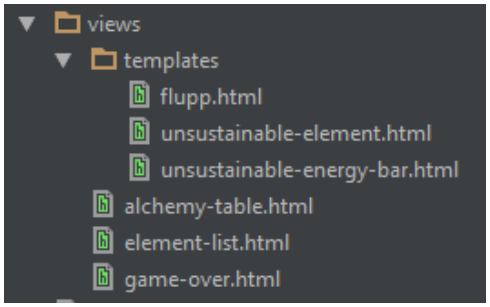
*JS*




*Styles*




*Views*



**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;
```

**js/controllers/_index.js**

```
'use strict';

var angular = require('angular');

module.exports = angular.module('app.controllers', []);

// Define the list of controllers here
require('./alchemy-table');
require('./game-over');
require('./element-list');
```

**js/controllers/alchemy-table.js**

```
'use strict';
var controllersModule = require('./_index');

/**
 * @ngInject
 */
function alchemyTableCtrl($scope, intersectService, elementService, dbPopulateService,$log) {
    // ViewModel
    var vm = this;

    dbPopulateService.constructModel().then(function () {
        dbPopulateService.generateMasterData().then(function () {
            elementService.getCurrentElements().then(function (data) {
                $log.log("alcemyTableCtrl",data);
                vm.elements = data;
                vm.energy = data[0].CurrentEnergy;

                angular.forEach(vm.elements, function (element) {
                    element.Location = JSON.parse(element.Location);
                });
            });
        });
    });

    //Combine Elements
    $scope.$on("UNS-ELM-DROPPED", function (event, element) {
        intersectService.getIntersectingElements(element, vm.elements).then(function
(intersecting) {
            console.log(element.Name + " intersects with " + intersecting[0].Name);

            elementService.combineElements(element, intersecting[0]).then(function
(combinedElements) {
                combinedElements[0].Location = element.Location;
                combinedElements[1].Location = intersecting[0].Location;
                combinedElements[0].CeId = element.CeId;
                combinedElements[1].CeId = intersecting[0].CeId;

                vm.elements.splice(vm.elements.indexOf(element), 1);
                vm.elements.splice(vm.elements.indexOf(intersecting[0]), 1);

                vm.elements.push(combinedElements[0]);
                vm.elements.push(combinedElements[1]);

                elementService.updateCurrentElement(combinedElements[0]);
                elementService.updateCurrentElement(combinedElements[1]);

                vm.energy -= combinedElements[0].EnergyUsage;
            }, function (err) {
                console.log("Well shit! That's not a valid combination.");
            })
        });
```

```
    });

    //Split elements
    $scope.$on("UNS-ELM-LONGTOUCH", function (event, element) {
        console.log("splitting: " + element.Name);
        elementService.splitElement(element).then(function (data) {
            data.Location = element.Location;
            data.CeId = element.CeId;

            vm.elements.splice(vm.elements.indexOf(element), 1);
            vm.elements.push(data);

            elementService.updateCurrentElement(data);

            vm.energy -= data.EnergyUsage / 2;
        }, function (err) {
            console.log("Well shit! That's not a splittable element.");
        });
    });
}

controllersModule.controller('alchemyTableCtrl', alchemyTableCtrl);
```

**js/controllers/element-list.js**

```
'use strict';
var controllersModule = require('./_index');

/**
 * @ngInject
 */
function elementListCtrl($scope, recipeService, elementService) {
    // ViewModel
    var vm = this;

    vm.unlockedElements = [];

    elementService.getBaseElements().then(function (elements) {
        vm.unlockedElements = vm.unlockedElements.concat(elements);
        vm.selected = vm.unlockedElements[0];
    });

    recipeService.getUnlockedRecipes().then(function (elements) {
        vm.unlockedElements = vm.unlockedElements.concat(elements);
    });

    $scope.$watchCollection("vm.selected", function (newCollection) {
        vm.selectedChildren = [];

        if (newCollection && newCollection.isBaseElement != 1 && newCollection.Id) {
            elementService.getElementParts(newCollection).then(function (elements) {
                console.log("tews",elements,newCollection.Id)
                vm.selectedChildren = elements;
            });
        }
    });
}

controllersModule.controller('elementListCtrl', elementListCtrl);
```

**js/controllers/game-over.js**

```
'use strict';
var controllersModule = require('./_index');

/**
 * @ngInject
 */
function gameOverCtrl($scope, intersectService, elementService, dataService) {
    // ViewModel
    var vm = this;
    vm.show = false;

    elementService.restoreBaseElements().then(function () {
        return dataService.updateCurrentEnergy(1000);
    }).then(function () {
        vm.show = true;
    });
}

controllersModule.controller('gameOverCtrl', gameOverCtrl);
```

**js/directives/_index.js**

```
'use strict';

var angular = require('angular');

module.exports = angular.module('app.directives', []);

// Define the list of directives here
require('./unsustainable-element.js');
require('./unsustainable-energy-bar.js');
require('./flupp.js');
```

**js/directives/flupp.js**

```
'use strict';

var directivesModule = require('./_index.js');

/**
 * @ngInject
 */
function flupp(dataService) {
    var directive = {};
    directive.templateUrl = "templates/flupp.html";
    directive.replace = true;
    directive.restrict = 'E';
    directive.scope = {
        elementData: '='
    };

    directive.link = function (scope, element, attributes) {
        scope.show = false;
        scope.elementParts = [];

        var shortTabData = null;

        scope.$root.$on("UNS-ELM-SHORT-TAB", function (event, data) {
            if (scope.show || data.elementData.$$hashKey != scope.elementData.$$hashKey){
                return;
            }

            scope.show = true;
            shortTabData = data;
        });

        scope.getPositionX = function () {
            if (shortTabData == null) {
                return;
            }

            return scope.elementData.Location.x + (shortTabData.width / 2) + 15;
        };

        scope.getPositionY = function () {
            if (shortTabData == null) {
                return;
            }

            return scope.elementData.Location.y - (shortTabData.width / 2);
        };

        if (!!window.cordova) {
            angular.element(document.body).bind("touchstart", onTouchStart);
        } else {
            angular.element(document.body).bind("mousedown", onTouchStart);
```

```
        }

        function onTouchStart() {
            scope.$apply(function () {
                if (scope.show) {
                    scope.show = false;
                }
            });
        }

        scope.$watch('elementData', function (curr) {
            if (curr == null) {
                return;
            }

            console.log("Query new elementParts");
            dataService.getElementParts(curr).then(function (data) {
                scope.elementParts = data;
            });
        });
    };

    return directive;
}

directivesModule.directive('flupp', flupp);
```

**js/directives/unsustainable-element.js**

```
'use strict';

var directivesModule = require('./_index.js');

/**
 * @ngInject
 */
function unsustainableElement(elementService, intersectService) {
    var directive = {};
    directive.templateUrl = "templates/unsustainable-element.html";
    directive.replace = true;
    directive.restrict = 'E';
    directive.scope = {
        elementData: '=',
        touchDuration: '=',
        touchDurationUntilShake: '='
    };

    directive.link = function (scope, element, attributes) {
        scope.elementData.Location = scope.elementData.Location || {'x': 100, 'y': 100};
        scope.elementClass = "";

        var mouseDown = false;
        var isLongTouch = false;
        var isMouseMoved = false;
        var timerUntilShake;
        var timer;
        var touchDuration = scope.touchDuration || 1200;
        var touchDurationUntilShake = scope.touchDurationUntilShake || 300;

        var bounds = document.getElementById("uns-alchemy-table");

        if (!!window.cordova) {
            element.bind("touchstart", onTouchStart);
            angular.element(bounds).bind("touchmove", onTouchMove);
            angular.element(bounds).bind("touchend", onTouchEnd);
        }
        else {
            element.bind("mousedown", onTouchStart);
            angular.element(bounds).bind("mousemove", onMouseMove);
            angular.element(bounds).bind("mouseup", onTouchEnd);
        }

        scope.getPositionX = function () {
            return scope.elementData.Location.x - element[0].clientWidth / 2;
        };

        scope.getPositionY = function () {
            return scope.elementData.Location.y - element[0].clientHeight / 2;
        };
```

```
var startPosition = angular.copy(scope.elementData.Location);
function onTouchStart(e) {
    if (mouseDown) {
        return;
    }

    console.log("Touchstart");
    mouseDown = true;
    startPosition = angular.copy(scope.elementData.Location);
    isMouseMoved = false;
    timerUntilShake = setTimeout(startLongTouch,touchDurationUntilShake);
}

function startLongTouch() {
    // If mouse was already moved before long touch starts
    if (isMouseMoved) {
        return;
    }

    console.log("Start long touch");
    scope.$apply(function () {
        scope.elementClass = "shake shake-horizontal shake-constant";
    });

    timer = setTimeout(onLongTouch, touchDuration);
    isLongTouch = true;
}


function cancelLongTouch() {
    console.log("Cancel long touch");
    scope.$apply(function () {
        scope.elementClass = "";
    });

    isLongTouch = false;

    if (timer) {
        clearTimeout(timer);
    }

    if (timerUntilShake) {
        clearTimeout(timerUntilShake);
    }
}

function onLongTouch() {
    console.log("Finish long touch");
    if (!mouseDown) {
        return;
    }
```

```
            cancelLongTouch();
            mouseDown = false;
            scope.$emit("UNS-ELM-LONGTOUCH", scope.elementData);
        }

        function onTouchEnd(e) {
            if (!mouseDown) {
                return;
            }

            console.log("Touchend");
            mouseDown = false;

            // If element is at start position
            if (!isLongTouch && !isMouseMoved &&
intersectService.checkIntersection(scope.elementData.Location, startPosition, 20)) {

                console.log("Short tab");
                scope.$root.$emit("UNS-ELM-SHORT-TAB", {elementData: scope.elementData, width:
element[0].clientWidth});

            }

            cancelLongTouch();

            elementService.updateCurrentElement(scope.elementData).then(function () {
                scope.$emit("UNS-ELM-DROPPED", scope.elementData);
            });
        }

        function onMouseMove(e) {
            if (!mouseDown) return;

            // Long touch tolerance
            var clientPosition = { x: e.clientX, y: e.clientY };
            if (isLongTouch && intersectService.checkIntersection(scope.elementData.Location,
clientPosition, 20)) {

                return;
            }

            scope.$apply(function () {
                scope.elementData.Location.x = clientPosition.x;
                scope.elementData.Location.y = clientPosition.y;
                resetPositionBounds();
            });

            // Do not continue if element wasn't moved from its start position
            if (intersectService.checkIntersection(scope.elementData.Location, startPosition,
20)) {

                return;
```

```
            }

            isMouseMoved = true;
            if (isLongTouch) {
                cancelLongTouch();
            }
        }

        function onTouchMove(e) {
            if (!mouseDown) return;
            if (e.touches.length > 1) return;

            // Long touch tolerance
            var clientPosition = { x: e.touches[0].clientX, y: e.touches[0].clientY };
            if (isLongTouch && intersectService.checkIntersection(scope.elementData.Location,
clientPosition, 20)) {

                return;
            }

            scope.$apply(function () {
                scope.elementData.Location.x = clientPosition.x;
                scope.elementData.Location.y = clientPosition.y;
                resetPositionBounds();
            });

            // Do not continue if element wasn't moved from its start position
            if (intersectService.checkIntersection(scope.elementData.Location, startPosition, 0))
{

                return;
            }

            isMouseMoved = true;
            if (isLongTouch) {
                cancelLongTouch();
            }
        }

        function resetPositionBounds() {
            // Top
            var topBound = document.body.clientHeight - bounds.clientHeight;
            if (scope.getPositionY() < topBound) {
                scope.elementData.Location.y = topBound + element[0].clientHeight / 2;
            }

            // Bottom
            var bottomBound = document.body.clientHeight;
            if (scope.getPositionY() + element[0].clientHeight > bottomBound) {
                scope.elementData.Location.y = bottomBound - element[0].clientHeight / 2;
            }
```

```
            // Left
            var leftBound = document.body.clientWidth - bounds.clientWidth;
            if (scope.getPositionX() < 0) {
                scope.elementData.Location.x = leftBound + element[0].clientWidth / 2;
            }

            // Right
            var rightBound = document.body.clientWidth;
            if (scope.getPositionX() + element[0].clientWidth > rightBound) {
                scope.elementData.Location.x = rightBound - element[0].clientWidth / 2;
            }
        }
    };

    return directive;
}

directivesModule.directive('unsustainableElement', unsustainableElement);
```

**js/directives/unsustainable-energy-bar.js**

```
'use strict';

var directivesModule = require('./_index.js');

/**
 * @ngInject
 */
function unsustainableEnergyBar($state, dataService) {
    var directive = {};
    directive.templateUrl = "templates/unsustainable-energy-bar.html";
    directive.replace = true;
    directive.restrict = 'E';
    directive.scope = {
        energy: '='
    };

    directive.link = function (scope, element, attributes) {
        scope.energyClass = "";

        scope.$watch('energy', function (energy) {
            if (energy == null) {
                return;
            }

            dataService.updateCurrentEnergy(energy).then(function () {
                if (energy <= 0) {
                    $state.go('gameOver');
                }

                scope.energyClass = "shake shake-constant";
                setTimeout(function () {
                    scope.$apply(function () {
                        scope.energyClass = "";
                    });
                }, 200);
            });
        });
    };

    return directive;
}

directivesModule.directive('unsustainableEnergyBar', unsustainableEnergyBar);
```

**js/services/_index.js**

```
'use strict';

var angular = require('angular');


module.exports = angular.module('app.services', [])
    .constant("AppSettings",require('../constants'));

// Define the list of services here
require('./intersect-service');
require('./element-service');
require('./db/dbpopulate-service');
require('./db/sqlite-service');
require('./data-service');
require('./recipe-service');
```

**js/services/db/sql/construct-model.js**

```
exports.query = function () {
    return "\
        CREATE TABLE IF NOT EXISTS Element (\
            Id INTEGER PRIMARY KEY,\
            Name TEXT NOT NULL,\
            Description TEXT NOT NULL,\
            Image TEXT NOT NULL\
        );\
        \
        CREATE TABLE IF NOT EXISTS Recipe (\
            Id INTEGER PRIMARY KEY,\
            Element1Id INTEGER NOT NULL,\
            Element2Id INTEGER NOT NULL,\
            ResultId INTEGER NOT NULL,\
            EnergyUsage INTEGER NOT NULL,\
            CONSTRAINT fk_Recipe_Element\
                FOREIGN KEY (Element1Id)\
                REFERENCES Element (Id)\
            CONSTRAINT fk_Recipe_Element1\
                FOREIGN KEY (Element2Id)\
                REFERENCES Element (Id)\
            CONSTRAINT fk_Recipe_Element2\
                FOREIGN KEY (ResultId)\
                REFERENCES Element (Id)\
        );\
        \
        CREATE TABLE IF NOT EXISTS Achievement (\
            Id INTEGER PRIMARY KEY,\
            Name TEXT NOT NULL,\
            Description TEXT NOT NULL,\
            Image TEXT NOT NULL,\
            Query TEXT NOT NULL\
        );\
        \
        CREATE TABLE IF NOT EXISTS Player (\
            Id INTEGER PRIMARY KEY,\
            CurrentEnergy INTEGER NOT NULL\
        );\
        \
        CREATE TABLE IF NOT EXISTS CurrentElement (\
            Id INTEGER PRIMARY KEY,\
            PlayerId INTEGER NOT NULL,\
            ElementId INTEGER NOT NULL,\
            Location TEXT NOT NULL,\
            CONSTRAINT fk_CurrentElement_Player1\
                FOREIGN KEY (PlayerId)\
                REFERENCES Player (Id),\
            CONSTRAINT fk_CurrentElement_Element1\
                FOREIGN KEY (ElementId)\
                REFERENCES Element (Id)\
        );\
```

```
        \
        CREATE TABLE IF NOT EXISTS UnlockedAchievement (\
            Id INTEGER PRIMARY KEY,\
            PlayerId INTEGER NOT NULL,\
            AchievementId INTEGER NOT NULL,\
            CONSTRAINT fk_UnlockedAchievement_Player1\
                FOREIGN KEY (PlayerId)\
                REFERENCES Player (Id),\
            CONSTRAINT fk_UnlockedAchievement_Achievement1\
                FOREIGN KEY (AchievementId)\
                REFERENCES Achievement (Id)\
        );\
        \
        CREATE TABLE IF NOT EXISTS UnlockedRecipe (\
            Id INTEGER PRIMARY KEY,\
            PlayerId INTEGER NOT NULL,\
            RecipeId INTEGER NOT NULL,\
            CONSTRAINT fk_UnlockedRecipe_Player1\
                FOREIGN KEY (PlayerId)\
                REFERENCES Player (Id),\
            CONSTRAINT fk_UnlockedRecipe_Recipe1\
                FOREIGN KEY (RecipeId)\
                REFERENCES Recipe (Id)\
        );\
    ";
};
```

**js/services/db/sql/generate-masterdata.js**

```
exports.query = function () {
    return "\
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (1, 'Water', 'Water is the chemical compound of hydrogen and oxygen.', 'water.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (2, 'Fire', 'Fire describes the formation of flames during burning.', 'fire.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (3, 'Steam', 'Steam consists of small watter drops in the air.', 'steam.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (4, 'Air', 'Air is the gas mix of earths atmosphere.', 'air.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (5, 'Dirt', 'Dirt is the death substance which is on the ground.', 'dirt.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (6, 'Mud', 'Mud contains 99% commercially available dirt.', 'mud.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (7, 'Energy', 'The energy on earth is unsustainable.', 'energy.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (8, 'Intelligence', 'Intelligence describes the comprehention and thinking of
something.', 'sample.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (9, 'River', 'A river is a natural flowing water resource.', 'river.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
```

```
        (10, 'Wind', 'Wind is air in a hurry.', 'wind.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (11, 'Earthquake', 'Destructive eruption of earth.', 'earthquake.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (12, 'Life', 'Life is 42.', 'life.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (13, 'Storm', 'The storm is a invention of ancient meteorologists.', 'storm.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (14, 'Might', 'Might is the power to change the thinking of other people.',
'might.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (15, 'Pressure', 'Pressure is the force applied perpendicular to the surface of an
object.', 'sample.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (16, 'Sun', 'The sun is a star which is orbiting the earth.', 'sun.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (17, 'Fish', 'Fishs are animals which can be fished.', 'sample.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (18, 'Fish Stick', 'A baked or fried snack similar to french fries but made of fish.',
'sample.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (19, 'Weapon', 'A weapon is a device which kills people.', 'sample.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
```

```
        VALUES\
        (20, 'Moscito', 'Small, flying, bloodsuckers.', 'sample.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (21, 'Death', 'The death is coming in the future for anybody.', 'sample.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (22, 'Stench', 'It smells... Take a deep breath.', 'stench.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (23, 'Human', 'You are a human, arent you?', 'sample.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (24, 'Plant', 'A plant is living in a flowerpot and has dignity and rights.',
'sample.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (25, 'King', 'The king is a normal human. But he has a crown on his head.',
'sample.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (26, 'War', 'The state similarlly to peace, in which humans are killing them mutually.',
'sample.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (27, 'Humans', 'Multiple humans.', 'sample.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (28, 'Wheat', 'Wheat is a plant which grows on the ground... Like literally any other
plant.', 'sample.png');\
        \
        INSERT INTO Element\
        (Id, Name, Description, Image)\
        VALUES\
        (29, 'Tree', 'Tree are made of paper and are available in any paper-store.',
'sample.png');\
```

```
\
INSERT INTO Element\
(Id, Name, Description, Image)\
VALUES\
(30, 'Zombie', 'Zombies are people who rose from death.', 'sample.png');\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(1, 1, 2, 3, 50);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(2, 5, 1, 6, 30);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(3, 7, 1, 9, 40);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(4, 7, 4, 10, 30);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(5, 7, 5, 11, 80);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(6, 7, 6, 12, 60);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(7, 7, 10, 13, 80);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(8, 7, 8, 14, 20);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(9, 4, 5, 15, 40);\
\
INSERT INTO Recipe\
```

```
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(10, 7, 2, 16, 120);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(11, 1, 12, 17, 30);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(12, 2, 17, 18, 50);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(13, 2, 8, 19, 40);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(14, 4, 12, 20, 20);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(15, 2, 12, 21, 200);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(16, 17, 21, 22, 40);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(17, 12, 8, 23, 100);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(18, 12, 5, 24, 50);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(19, 14, 23, 25, 80);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
```

```
(20, 14, 19, 26, 100);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(21, 23, 23, 27, 30);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(22, 16, 24, 28, 50);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(23, 7, 24, 29, 80);\
\
INSERT INTO Recipe\
(Id, Element1Id, Element2Id, ResultId, EnergyUsage)\
VALUES\
(24, 12, 21, 30, 130);\
\
INSERT INTO Player\
(Id, CurrentEnergy)\
VALUES\
(1, 1000);\
\
INSERT INTO CurrentElement\
(Id, PlayerId, ElementId, Location)\
VALUES\
(1, 1, 1, '{\"x\": 100, \"y\": 200}');\
\
INSERT INTO CurrentElement\
(Id, PlayerId, ElementId, Location)\
VALUES\
(2, 1, 2, '{\"x\": 200, \"y\": 200}');\
\
INSERT INTO CurrentElement\
(Id, PlayerId, ElementId, Location)\
VALUES\
(3, 1, 4, '{\"x\": 300, \"y\": 200}');\
\
INSERT INTO CurrentElement\
(Id, PlayerId, ElementId, Location)\
VALUES\
(4, 1, 5, '{\"x\": 400, \"y\": 200}');\
\
INSERT INTO CurrentElement\
(Id, PlayerId, ElementId, Location)\
VALUES\
(5, 1, 7, '{\"x\": 500, \"y\": 200}');\
\
```

```
        INSERT INTO CurrentElement\
        (Id, PlayerId, ElementId, Location)\
        VALUES\
        (6, 1, 8, '{\"x\": 600, \"y\": 200}');\
    ";
};
```

**js/services/db/sqlite-service.js**

```
'use strict';

var servicesModule = require('./../_index.js');

/**
 * @ngInject
 */
servicesModule.service('sqliteService', function ($q, $timeout, $window, AppSettings, $log) {
    var service = {};
    var db;
    if ($window.sqlitePlugin) {
        db = $window.sqlitePlugin.openDatabase(
            {
                name: AppSettings.dbName,
                createFromLocation: 1,
                location: 1,
                androidLockWorkaround: 1
            });
    }
    else {
        db = $window.openDatabase(AppSettings.dbName, '1.0', 'Unsustainable Database', 2 * 1024 *
1024);
    }

    service.query = function (sql, preparedValues) {
        preparedValues = preparedValues || [];
        var deferred = $q.defer();
        db.transaction(function (tx) {
            tx.executeSql(sql, preparedValues, function (db, res) {
                    $log.log("SQLData",res);
                    var result = [];
                    for(var i = 0; i<res.rows.length;i++){
                        result.push(angular.copy(res.rows.item(i)));
                    }
                    deferred.resolve(result);
                }, function (tx, err) {
                    $log.error("SQLError ",err);
                    deferred.reject(err);
                }
            );
        });

        return deferred.promise;
    };

    service.chain = function (queries) {
        return queries.reduce(function (previous, query) {
            return previous.then(function () {
                return service.query(query, []);
            });
        }, $q(function (resolve) { resolve()}));
```

```
    };

    return service;
});
```

**js/services/db/db-populate-service.js**

```
'use strict';

var servicesModule = require('./../_index.js');

/**
 * @ngInject
 */
servicesModule.service('dbPopulateService', function ($http, $q, sqliteService) {
    var service = {};

    service.constructModel = function () {
        var sql = require('./sql/construct-model.js').query();
        var queries = sqlToQueries(sql);

        return sqliteService.chain(queries);
    };

    service.generateMasterData = function () {
        var deferred = $q.defer();

        // TODO: Is it enough to check only for elements containing data?
        sqliteService.query("SELECT COUNT(*) AS count FROM Element", []).then(function (data) {
            if (!(data[0].count > 0)) {
                var sql = require('./sql/generate-masterdata.js').query();
                var queries = sqlToQueries(sql);

                sqliteService.chain(queries).then(function () {
                    deferred.resolve();
                });
            } else {
                deferred.resolve();
            }
        });

        return deferred.promise;
    };

    function sqlToQueries(sql) {
        var queries = sql.split(";");
        queries.splice(queries.length - 1, 1);

        return queries;
    }

    return service;
});
```

**js/services/data-service.js**

```
'use strict';

var servicesModule = require('./_index.js');

/**
 * @ngInject
 */
servicesModule.service('dataService', function ($q, $timeout, sqliteService, $log) {
    var service = {};

    service.getCurrentElements = function () {
        var query = "SELECT e.Id, e.Name, e.Description, e.Image, p.CurrentEnergy, ce.Location,
ce.Id AS CeId FROM CurrentElement AS ce " +
            "JOIN Player AS p ON p.Id = ce.PlayerId " +
            "JOIN Element AS e ON e.Id = ce.ElementId " +
            "WHERE p.Id = 1" +
            ";";

        return sqliteService.query(query, []);
    };

    service.getAllElements = function () {
        var query = "SELECT * FROM Element ";
        return sqliteService.query(query);
    };

    service.getCombinedElement = function (element1, element2) {
        var parameters = [element1.Id, element2.Id, element1.Id, element2.Id];
        var query = "SELECT e.Id, e.Name, e.Description, e.Image, r.EnergyUsage,r.Id as RecipeId
FROM Element AS e " +
            "JOIN Recipe AS r ON r.ResultId = e.Id " +
            "WHERE (r.Element1Id = ? AND r.Element2Id = ?) OR (r.Element2Id = ? AND r.Element1Id
= ?)";

        return sqliteService.query(query, parameters);
    };

    service.getElementParts = function (element) {
        var parameters = [element.Id];
        var query = "SELECT e.Id, e.Name, e.Description, e.Image, r.EnergyUsage FROM Element AS e
" +
            "JOIN Recipe AS r ON r.Element1Id = e.Id OR r.Element2Id = e.Id " +
            "WHERE r.ResultId = ?";

        return sqliteService.query(query, parameters);
    };

    service.getBaseElements = function () {
        var parameters = [];
        var query = "SELECT e.Id, e.Name, e.Description, e.Image, r.EnergyUsage FROM Element AS e
" +
```

```
            "LEFT JOIN Recipe AS r ON r.ResultId = e.Id " +
            "WHERE r.ResultId IS NULL";

        return sqliteService.query(query, parameters);
    };

    service.getBaseElementsExcept = function (elementId) {
        var parameters = [elementId];
        var query = "SELECT e.Id, e.Name, e.Description, e.Image, r.EnergyUsage FROM Element AS e
" +
            "LEFT JOIN Recipe AS r ON r.ResultId = e.Id " +
            "WHERE r.ResultId IS NULL AND e.Id != ?";

        return sqliteService.query(query, parameters);
    };

    service.isBaseElement = function (element) {
        var deferred = $q.defer();

        var parameters = [element.Id];
        var query = "SELECT COUNT(*) AS count FROM Element AS e \
            LEFT JOIN Recipe AS r ON r.ResultId = e.Id\
            WHERE r.ResultId IS NULL AND e.Id = ?\
        ";

        sqliteService.query(query, parameters).then(function (result) {
            deferred.resolve(result[0].count == 1);
        });

        return deferred.promise;
    };

    service.getBaseElements = function () {
        var parameters = [];
        var query = "SELECT e.Id,e.Name,e.Image,e.Description,1 as isBaseElement FROM Element AS
e \
            LEFT JOIN Recipe AS r ON r.ResultId = e.Id\
            WHERE r.ResultId IS NULL\
        ";

        return sqliteService.query(query, parameters);
    };

    service.restoreBaseElements = function () {
        var deferred = $q.defer();

        service.getBaseElements().then(function (baseElements) {
            var queries = [];
            queries.push("DELETE FROM CurrentElement");


            var location = {x: 100, y: 200};
```

```
            angular.forEach(baseElements, function (element) {
                queries.push("INSERT INTO CurrentElement\
                    (PlayerId, ElementId, Location)\
                    VALUES\
                    (1, " + element.Id + ", '" + JSON.stringify(location) + "')\
                ");

                location.x = location.x + 100;
            });

            sqliteService.chain(queries).then(function () {
                deferred.resolve();
            });
        });

        return deferred.promise;
    };

    service.updateCurrentElement = function (element) {
        var parameters = [element.Id, JSON.stringify(element.Location), element.CeId];
        var query = "UPDATE CurrentElement SET \
            ElementId = ?, Location = ?\
            WHERE Id = ?\
        ";

        return sqliteService.query(query, parameters);
    };

    service.updateCurrentEnergy = function (energy) {
        var parameters = [energy];
        var query = "UPDATE Player SET CurrentEnergy = ? WHERE Id = 1";

        return sqliteService.query(query, parameters);
    };

    service.getUnlockedRecipes = function () {
        var parameters = [];
        var query = "SELECT r.Id as RecipeId,r.Element1Id,r.Element2Id,r.ResultId as
Id,r.EnergyUsage,e.Description,e.Name,e.Image FROM Recipe AS r \
        JOIN Element AS e ON e.Id = r.ResultId\
        JOIN UnlockedRecipe AS ur ON ur.RecipeId = r.Id\
        JOIN Player AS p ON p.Id = ur.PlayerId";

        return sqliteService.query(query, parameters);
    };

    service.unlockRecipe = function (recipeId) {
        var deferred = $q.defer();
        if (recipeId != null) {
            sqliteService.query("SELECT count(*) AS count FROM UnlockedRecipe WHERE RecipeId =
?", [recipeId]).then(function (data) {
                if (data[0].count >= 1) {
```

```
                    deferred.reject();
                    return;
                }

                var query = "INSERT INTO UnlockedRecipe(RecipeId,PlayerId) VALUES (?,?) ";
                console.log(recipeId);
                sqliteService.query(query, [recipeId, 1]).then(deferred.resolve
                    , deferred.reject);

            }, deferred.reject);
        }
        else
        {
            deferred.reject("Id cannot be null!");
        }

        return deferred.promise;
    };

    return service;
});
```

**js/services/element-service.js**

```
'use strict';

var servicesModule = require('./_index.js');

/**
 * @ngInject
 */
servicesModule.service('elementService', function ($q, $timeout, dataService, $log) {
    var service = {};

    service.combineElements = function (element1, element2) {
        var deferred = $q.defer();
        dataService.getCombinedElement(element1, element2).then(function (data) {
            if (data.length != 1) {
                deferred.reject();
            }
            dataService.unlockRecipe(data[0].RecipeId);
            deferred.resolve([data[0], angular.copy(data[0])]);
        }, deferred.reject);

        return deferred.promise;
    };

    service.splitElement = function (element) {
        var deferred = $q.defer();

        dataService.isBaseElement(element).then(function (isBaseElement) {
            var promise = null;
            if (isBaseElement) {
                promise = dataService.getBaseElementsExcept(element.Id);
            } else {
                promise = dataService.getElementParts(element);
            }

            promise.then(function (data) {
                if (data.length == 0) {
                    deferred.reject();
                }

                var min = 0;
                var max = data.length - 1;

                // Random number between min and max
                var random = Math.floor(Math.random()*(max-min+1)+min);

                deferred.resolve(data[random]);
                $log.log(data);
            }, deferred.reject);
        });
```

```
        return deferred.promise;
    };

    service.getCurrentElements = function () {
        return dataService.getCurrentElements();
    };

    service.getBaseElements= function () {
        return dataService.getBaseElements();
    };

    service.updateCurrentElement = function (element) {
        return dataService.updateCurrentElement(element);
    };

    service.restoreBaseElements = function () {
        return dataService.restoreBaseElements();
    };

    service.getElementParts = function (element) {
        return dataService.getElementParts(element);
    };

    return service;
});
```

**js/services/recipe-service.js**

```
'use strict';

var servicesModule = require('./_index.js');

/**
 * @ngInject
 */
servicesModule.service('recipeService', function ($q, $timeout, dataService, $log) {
    var service = {};

    service.getUnlockedRecipes = function () {
       return dataService.getUnlockedRecipes();
    };

    return service;
});
```

**js/services/intersect-service.js**

```
'use strict';

var servicesModule = require('./_index.js');

/**
 * @ngInject
 */
servicesModule.service('intersectService',function ($q, $timeout) {
    var service = {};

    service.getIntersectingElements = function (element, elements) {
        var defer = $q.defer();
        $timeout(function () {
            var intersectingElements = [];
            for (var i = 0; i < elements.length; i++) {
                if (element.$$hashKey != elements[i].$$hashKey) {
                    if (service.checkIntersection(element.Location, elements[i].Location, 20)) {
                        intersectingElements.push(elements[i]);
                    }
                }
            }

            if (intersectingElements.length > 0) {
                defer.resolve(intersectingElements);
            } else {
                defer.reject();
            }
        });

        return defer.promise;
    };

    service.checkIntersection = function (position1, position2, tolerance) {
        if ((position1.x > (position2.x + tolerance)) || ((position1.x + tolerance) <
position2.x)) {
            return false;
        }
        if ((position1.y > (position2.y + tolerance)) || ((position1.y + tolerance) <
position2.y)) {
            return false;
        }
        return true;
    };

    return service;
});
```

**styles/_typography.scss**

```
p {
  margin-bottom: 1em;
}

.heading {
  margin-bottom: 0.618em;

  &.-large {
    font-size: $font-size--lg;
    font-weight: bold;
    line-height: $half-space * 3 / 2;
  }

  &.-medium {
    font-size: $font-size--md;
    font-weight: normal;
    line-height: $half-space;
  }

  &.-small {
    font-size: $font-size--sm;
    font-weight: bold;
    line-height: $half-space * 2 / 3;
  }

  &.-smallest {
    font-size: $font-size--xs;
    font-weight: bold;
  }
}

h1 {
  @extend .heading.-large;
}

h2 {
  @extend .heading.-medium;
}

h3 {
  @extend .heading.-small;
}
```

**styles/_vars.scss**

```
// colors
$font-color--dark: #333;
$font-color--light: #fff;
$background--light: #eee;
$background--dark: #222;
$blue: #1f8de2;
$green: #1fe27b;
$red: #e21f3f;
$primary: #970eb3;

// spacing
$full-space: 40px;
$half-space: 20px;

// font sizing
$font-size--xs: 10px;
$font-size--sm: 12px;
$font-size--md: 16px;
$font-size--lg: 24px;
$font-size--xl: 32px;
```

**styles/main.scss**

```
@import 'vars';
@import 'typography';

html,body {
  font-family: Helvetica, sans-serif;
  color: $font-color--dark;
  background-color: $background--light;
  height: 100%;
  margin:0;
  padding: 0;
  -webkit-user-select: none;  /* Chrome all / Safari all */
  -moz-user-select: none;     /* Firefox all */
  -ms-user-select: none;      /* IE 10+ */
  -o-user-select: none;
  user-select: none;
}

.clearfix {
  clear: both;
}

.uns-main-view {
  height: 100%;
  width: 100%;
  position: absolute;
  overflow: hidden;
  background: url("../images/backgrounds/wood-background.jpg") no-repeat center center fixed;
  background-size: cover;
  -webkit-background-size: cover;
  -moz-background-size: cover;
  -o-background-size: cover;
}

/* Alchemy Table */
.uns-alchemy-nav {
  width: 100%;
  height: 15%;
  background-color: #5C3D1F;
  opacity: 0.5;
}

.uns-alchemy-table {
  height: 85%;
  width: 100%;
}

.uns-energy-container {
  width: 80%;
  height: 100%;
  float: left;
  position: relative;
```

```
}

.uns-navigation-buttons {
  height: 100%;
  width: 20%;
  float: right;
  position: relative;
}

.uns-navigation-buttons a {
  display: block;
  height: 70%;
  position: absolute;
  top: 0;
  bottom: 0;
  margin: auto 0;
}

.uns-navigation-buttons a img {
  height: 100%;
}

.uns-energy-bar {
  height: 50%;
  width: 95%;

  position: absolute;
  top: 0;
  bottom: 0;
  margin: auto 2.5%;
}

.uns-energy-bar > div {
  height: 100%;
  background-color: $primary;
}

.uns-energy-bar,
.uns-energy-bar > div {
  border-radius: 0.5em;
}

.uns-element-container,
.uns-element-container img {
  width: 3em;
  height: 3em;
}

.uns-element-container {
  text-align: center;
  color: #ccc;
}
```

```
/* Flupp */
.uns-flupp-container {
  width: 12em;
  min-height: 1em;
  background: #5C3D1F;
  opacity: 0.8;
  border-radius: 0.5em;
  color: #ccc;
  padding: 1em;
}

.uns-flupp-container:before {
  content: ' ';
  position: absolute;
  width: 0;
  height: 0;
  left: -0.75em;
  top: 1em;
  border-style: solid;
  border-width: 0.75em 0.75em 0.75em 0;
  border-color: transparent #5C3D1F;
}

.uns-flupp-recipes {
  margin-top: 1em;
  width: 100%;
}

.uns-flupp-recipe-ingredient {
  width: 15%;
  float: left;
}

.uns-flupp-recipe-ingredient img {
  width: 100%;
}

/* Game Over */
.uns-gameover-background {
  width: 100%;
  height: 100%;
  background: url("../images/backgrounds/game-over.jpg") no-repeat center center fixed;
  background-size: cover;
  -webkit-background-size: cover;
  -moz-background-size: cover;
  -o-background-size: cover;
}

.uns-btn-gameover {
  position: absolute;
  width: 50%;
```

```
  top: 45%;
  left: 25%;
}

.uns-btn-gameover img {
  width: 100%;
}


/* List  */
.uns-list-container {
  overflow:auto;
}

.uns-list-square-element {
  text-align: center;
  margin: 1em;
  height: 4em;
  width: 4em;
  float: left;
}

/*Element List */
.uns-detail-view {
 margin: 2.5%;
 height: 75%;
}

.uns-detail-view {
  color: #ccc;
  background-color: rgba(92, 61, 31, 0.5);
}

.uns-detail-view .uns-detail-view-left{
  width: 60%;
  height: 100%;
  float:left;
}

.uns-detail-view .uns-detail-view-right{
  height: 100%;
  width:39%;
  float:left;
  border-left: 2px solid rgba(92, 61, 31, 1);
}

.uns-details{
  margin: 1em;
}

.uns-details-heading{
  float: left;
```

```
}

.uns-details-heading img{
  float:left;
}

.uns-details-heading .info{
  margin-top: 1em;
  float:left;
}

.uns-detail-recipe-ingredient {
  margin-right: 1em;
  width: 1em;
  float: left;
}
```

**views/alchemy-table.html**

```
<div class="uns-alchemy-nav">
    <div class="uns-energy-container">
        <unsustainable-energy-bar energy="vm.energy"></unsustainable-energy-bar>
    </div>
    <div class="uns-navigation-buttons">
        <a ui-sref="elementList"><img src="images/buttons/element-list.png" /></a>
    </div>
    <div class="clearfix"></div>
</div>

<div id="uns-alchemy-table" class="uns-alchemy-table">
    <unsustainable-element ng-repeat="element in vm.elements" element-
data="element"></unsustainable-element>
    <flupp ng-repeat="element in vm.elements" element-data="element"></flupp>
</div>
```

**views/element-list.html**

```
<div class="uns-alchemy-nav">
    <div class="uns-navigation-buttons">
        <a ui-sref="alchemyTable"><img src="images/buttons/element-list.png" /></a>
    </div>
    <div class="clearfix"></div>
</div>
<div class="uns-detail-view">
    <div class="uns-list-container  uns-detail-view-left">
        <div ng-click="vm.selected = element;" ng-repeat="element in vm.unlockedElements"
             class="uns-list-square-element">
            <img ng-src="images/{{element.Image}}" />
            {{element.Name}}
        </div>
    </div>

    <div class="uns-detail-view-right">
        <div class="uns-details">
            <div class="uns-details-heading">
                <img ng-src="images/{{vm.selected.Image}}" />
                <div class="info">
                    <div>{{vm.selected.Name}}</div>
                    <div ng-hide="vm.selected.isBaseElement">Energy Usage:
{{vm.selected.EnergyUsage}}</div>
                    <div ng-show="vm.selected.isBaseElement">Base Element</div>
                </div>
            </div>
            <div class="clearfix"></div>

            <p>{{vm.selected.Description}}</p>

            <div ng-repeat="celement in vm.selectedChildren" class="uns-detail-recipe-
ingredient">
                <img style="height: 2em;width: 2em" ng-src="images/{{celement.Image}}">
            </div>
        </div>
    </div>

    <div class="clearfix"></div>
</div>
```

**views/game-over.html**

```
<div class="uns-gameover-background">
    <a ng-show="vm.show" ui-sref="alchemyTable">
        <div class="uns-btn-gameover">
            <img src="images/buttons/try-again.png" />
        </div>
    </a>
</div>
```

**views/templates/flupp.html**

```
<div ng-show="show" class="uns-flupp-container"
     ng-style="{'position':'absolute','top':getPositionY() +'px', 'left':getPositionX() +'px'}">
    <strong>{{elementData.Name}}</strong><br />
    {{elementData.Description}}
    <div class="uns-flupp-recipes">
        <div ng-repeat="partial in elementParts" class="uns-flupp-recipe-ingredient">
            <img ng-src="images/{{partial.Image}}" />
        </div>
        <div class="clearfix"></div>
        <div ng-show="elementParts.length == 0">
            Base element
        </div>
    </div>
</div>
```

**views/templates/unsustainable-element.html**

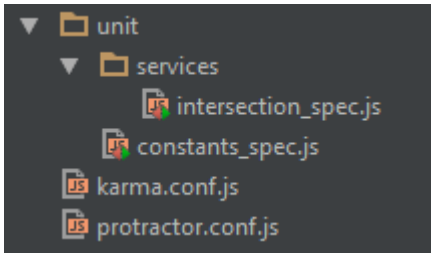
```
<div ng-style="{'position':'absolute','top':getPositionY() +'px', 'left':getPositionX() +'px'}"
ng-class="elementClass">
    <div class="uns-element-container">
        <img draggable="false" ng-src="images/{{elementData.Image}}" />
        {{elementData.Name}}
    </div>
</div>
```

**views/templates/unsustainable-energy-bar.html**

```
<div class="uns-energy-bar">
    <div ng-class="energyClass" ng-style="{'width': (100 / 1000) * energy + '%'}"></div>
</div>
```

## Anhang B: Testcode Unit-Tests

### Übersicht




Es wird nur der Intersection Service getestet, da die anderen Services auf die Datenbank zugreifen, und dieser Zugriff von unserem Testrunner leider nicht unterstützt wird. Deshalb konnten wir nur denjenigen Service testen, der nicht auf die Datenbank zugreift.

### services/intersection_spec.js

```
/*global angular */

'use strict';

describe('Unit: IntersectService', function () {

    var service;
    var $rootScope;
    var $timeout;

    beforeEach(angular.mock.module('app'));
    beforeEach(angular.mock.module('app.services'));


    beforeEach(angular.mock.inject(function (intersectService, _$rootScope_, _$timeout_) {
        service = intersectService;
        $rootScope = _$rootScope_;
        $timeout = _$timeout_;
    }));

    it('should exist', function () {
        expect(service).toBeDefined();
    });

    it('should find an Intersection', function () {
        var position1 = {x: 100, y: 100};
        var position2 = {x: 100, y: 100};
        var tolerance = 20;

        expect(service.checkIntersection(position1, position2, tolerance)).toEqual(true);
    });

    it('should not find an Intersection', function () {
        var position1 = {x: 200, y: 179};
        var position2 = {x: 200, y: 200};
        var tolerance = 20;
```

```
        expect(service.checkIntersection(position1, position2, tolerance)).toEqual(false);
    });

    it('should get only the intersecting elements with tolerance 20', function () {
        var element = {Id: 1, Name: 'Water', Location: {x: 200, y: 200}, $$hashKey: "1"};
        var elements = [
            {Id: 1, Name: 'Water', Location: {x: 200, y: 200}, $$hashKey: "1"},
            {Id: 2, Name: 'Fire', Location: {x: 200, y: 200}, $$hashKey: "2"},
            {Id: 1, Name: 'Water', Location: {x: 221, y: 179}, $$hashKey: "3"},
            {Id: 3, Name: 'Steam', Location: {x: 179, y: 221}, $$hashKey: "4"},
            {Id: 3, Name: 'Steam', Location: {x: 180, y: 220}, $$hashKey: "5"},
            {Id: 3, Name: 'Steam', Location: {x: 220, y: 180}, $$hashKey: "6"}
        ];

        var result;
        service.getIntersectingElements(element, elements).then(function (res) {
            result = res;
        });

        $rootScope.$apply();
        $timeout.flush();

        expect(result).toEqual([
            {Id: 2, Name: 'Fire', Location: {x: 200, y: 200}, $$hashKey: "2"},
            {Id: 3, Name: 'Steam', Location: {x: 180, y: 220}, $$hashKey: "5"},
            {Id: 3, Name: 'Steam', Location: {x: 220, y: 180}, $$hashKey: "6"}
        ]);
    });
});
```