

**PerfectPopCorn**

**Semesterarbeit**im Studiengang Informatik

**Christian Bachmann**an der ZHAW Zürich

Betreuer: Peter Egli

# Inhaltsverzeichnis

[Inhaltsverzeichnis 1](#_Toc419400722)

[1 Einleitung 3](#_Toc419400723)

[1.1 Vision 3](#_Toc419400724)

[1.2 Motivation 3](#_Toc419400725)

[1.3 Ist-Zustand 3](#_Toc419400726)

[1.4 Aufgabenstellung 4](#_Toc419400727)

[1.4.1 Thema 4](#_Toc419400728)

[1.4.2 Ausgangslage 4](#_Toc419400729)

[1.4.3 Ziele der Arbeit 4](#_Toc419400730)

[1.4.4 Aufgabenstellung 4](#_Toc419400731)

[1.4.5 Erwartete Resultate 4](#_Toc419400732)

[2 Projektmanagement 5](#_Toc419400733)

[2.1 Projektplan 5](#_Toc419400734)

[2.2 Soll / Ist Analyse 6](#_Toc419400735)

[2.3 RandBedienungen 7](#_Toc419400736)

[2.3.1 Zeitlicher Rahmen 7](#_Toc419400737)

[2.3.2 Organisatorischer Rahmen 7](#_Toc419400738)

[3 Anforderungsanalyse 8](#_Toc419400739)

[3.1 Einleitung 8](#_Toc419400740)

[3.2 Use-Cases 9](#_Toc419400741)

[3.2.1 Akteure 9](#_Toc419400742)

[4 Analyse 10](#_Toc419400743)

[4.1 Zu optimierende Prozesse 10](#_Toc419400744)

[4.2 EVA-Prinzip 13](#_Toc419400745)

[4.2.1 Eingabe 13](#_Toc419400746)

[4.2.2 Verarbeitung 13](#_Toc419400747)

[4.2.3 Ausgabe 13](#_Toc419400748)

[4.3 Verarbeitung von Mikrofonsignalen 14](#_Toc419400749)

[4.3.1 Spektrumanalysator 14](#_Toc419400750)

[4.3.2 FFT-Analysator 14](#_Toc419400751)

[4.4 Technische Umgebung 16](#_Toc419400752)

[5 Design der Software 17](#_Toc419400753)

[5.1 Einleitung 17](#_Toc419400754)

[5.2 Design Patterns 17](#_Toc419400755)

[5.2.1 Target-Action Pattern 17](#_Toc419400756)

[5.2.2 MVC, Model View Controller 17](#_Toc419400757)

[5.3 Mockups 18](#_Toc419400758)

[5.3.1 Ansicht App 18](#_Toc419400759)

[6 Implementation 19](#_Toc419400760)

[6.1 Technologien 19](#_Toc419400761)

[6.1.1 Adobe Air 19](#_Toc419400762)

[6.1.2 ActionScript 3 19](#_Toc419400763)

[6.2 Ansteuerung des Mikrofons 20](#_Toc419400764)

[6.3 Auswertung des Frequenzsspektrums 21](#_Toc419400765)

[6.4 Unit-Testing (Backend) 23](#_Toc419400766)

[6.5 Unit-Testing (Frontend) 23](#_Toc419400767)

[6.6 Integration-Testing 24](#_Toc419400768)

[6.7 Test der Akzeptanzkriterien 25](#_Toc419400769)

[6.8 Überprüfung Aufgabestellung 26](#_Toc419400770)

[7 Fazit und Schlusswort 27](#_Toc419400771)

[Glossar 28](#_Toc419400772)

[Abbildungsverzeichnis 29](#_Toc419400773)

[Tabellenverzeichnis 30](#_Toc419400774)

[Literaturverzeichnis 31](#_Toc419400775)

[Bestätigung über die Selbständigkeit 32](#_Toc419400776)

# Einleitung

## Vision



Abbildung - Verbranntes Mikrowellen-Popcorn von Christian Bachmann

Verbrannte Mikrowellen Popcorns gehören der Vergangenheit an. Langweilige Wartezeiten können für sinnvolleres genutzt werden.

## Aufgabenstellung

### Thema

Das perfekte Mikrowellen Popcorn erstellen durch Android App mit RealTime Soundanalyse und Internet of Things.

### Ausgangslage

Im Detailhandel kann für die Mikrowelle Popcorn mit verschiedenen Zusätzen erworben werden. Je nach Mikrowelle und Umgebung wird die Popcornpackung verschieden lang aufgeheizt. Der Konsument kann da schon einmal 6 Minuten aber min. 2 ¼ Minuten vor der Mikrowelle ausharren. Dann gilt es denn richtigen Zeitpunkt zu finden um möglichst alle Popcorn aufgeplatzt und keine schwarzen Popcorns zu haben. Auf Social Media Kanälen können hunderte von Bilder von schwarzen Mikrowellen-Popcorns gefunden werden. Das mühsame Warten soll der Vergangenheit angehören und perfekte Popcorns sollen mit dieser Seminararbeit geschaffen werden

### Ziele der Arbeit

Das Endprodukt dieser Seminararbeit soll ein funktionierende Native Android App sein. Die App soll das Mikrofon des Telefons auslesen und den Input als einfaches Spektrum Analyse visuell darstellen. Der Mikrowelle soll, über die App ferngesteuert, den Strom entzogen werden können. Die Machbarkeit der Perfect Popcorn App mittels Soundanalyse soll geprüft werden. Wunschziel wäre es das Android Smartphone auf die Mikrowelle zu legen, mittels Mikrophone Input eine Realtime Soundanalyse durch zuführen und den richtigen Zeitpunktes der Mikrowellenabstellung zu finden. Die App stellt zu diesem Zeitpunkt ferngesteuert die Mikrowelle ab.

### Aufgabenstellung

* Erstellen einer Anforderungsanalyse
* Entwicklung einer Android App in Adobe Air
* Ansprechen des Android Mikrophone
* Verarbeitung des Mikrofon Inputs in Real Time
* Visualisierung des Mikrofon Inputs als einfache Spektrum Analyse
* Bau einer Elektronik um mittels Mikrocontroller 230V zuschalten
* Clientprogramm welches Befehle der App empfängt und den Mikroncontroller ansteuern kann
* Prüfen der Machbarkeit Soundanalyse für den richtigen Zeiptunkt der Mikrowellenabstellung und Beendung des Popcornhitzungsprozess
* Finetunning und Verbesserung

### Erwartete Resultate

* Natives Android App verfügbar als APK
* Client Programm als Executable
* Physisch umgesetzte Elektronik
* Schriftlich festgehaltene Ergebnisse über die Machbarkeit
* Dokumentation

# Projektmanagement

Das folgende Kapitel beschreibt die Einzelheiten der Projektplanung. Sie beinhaltet die Werkzeuge für die Umsetzung des Projekts, die Aufteilung in die verschiedenen Phasen, den Zeitplan und die Meilensteine sowie einige administrative Angaben.

## Projektplan

Der Projektplan illustriert die Strukturierung des Projektes über die gut sechs Monate lange Projektzeit.

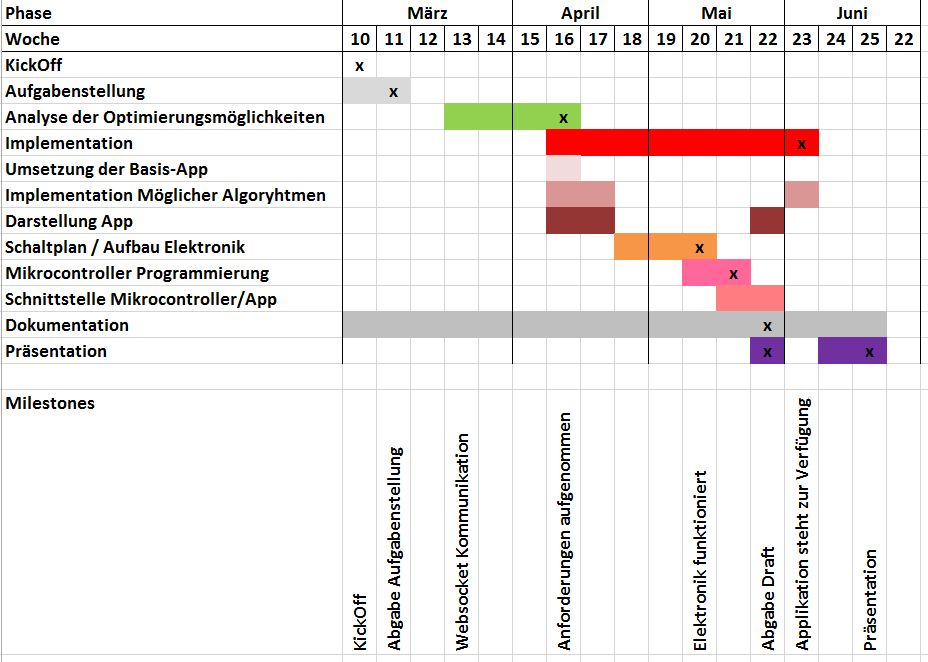


Abbildung - Projektplan

## Soll / Ist Analyse

|  |  |  |
| --- | --- | --- |
| Tätigkeit | Soll Stunden | Ist Stunden |
| Initialisierung | **4** | **3** |
| Dokumentation | **14** | **Zurzeit 15** |
| Analyse | **12** | **11** |
| Design und Architektur | **5** | **4** |
| Entwicklung/Elektronik | **20** | **21** |
| Vorbereitung Abgabe | **3** |  |
| Reserve | **0** |  |
| Total | **58** | **Aktuell 54** |

Tabelle - Soll-Ist Vergleich

## RandBedienungen

### Zeitlicher Rahmen

Der Offizielle Projektstart ist der 4. März 2015. Das Projekt muss bis spätestens 17. Juni 2015 abgegeben werden.

Der Aufwand für die Bearbeitung der Seminararbeit soll mindestens 50 Stunden umfassen.

### Organisatorischer Rahmen

Nachstehend sind alle massgeblich involvierten Personen aufgeführt.

|  |  |
| --- | --- |
| Person | Kontakt |
| Peter Egli, ZHAW  (Lehrperson, Betreuer) | ZHAW Standort Zürich Lagerstrasse 41 / 8004 Zürich  eglp@zhaw.ch |
| Dr. Reto Knaack, ZHAW  (Studiengangs Leiter) | ZHAW Standort Zürich Lagerstrasse 41 / 8004 Zürich  Reto.Knaack@zhaw.ch |
| Christian Bachmann, inaffect AG  (Student) | inaffect AG Beckenhofstrasse 26 / 8006 Zürich 043 210 94 10  bachmch3@students.zhaw.ch |

Tabelle - Beteiligte Personen

# Anforderungsanalyse

Dieses Kapitel beschreibt die Anforderungen an Perfect Popcorn. Die Anforderungen bilden die Basis für die Architektur, das Softwaredesign, die Implementation und die Testfälle. Ihnen ist dem entsprechend ein sehr grosser Stellenwert zuzuschreiben.

## Einleitung

Im Rahmen dieser Seminararbeit wird ein Prototypensystem entwickelt, welches probiert die groben Anforderungen der Aufgabenstellung zu erfühlen. Erst während des Projekts werden diese Anforderungen verfeinert erhoben. Sowohl bei der Anforderungs-Erhebung/Analyse als auch bei der Umsetzung wird der Fokus auf die Festigung des im Unterricht gelernten Stoffs gelegt.

Die Schlüsselwörter „muss“, „muss nicht“, „erforderlich“, „empfohlen“, „sollte“, „sollte nicht“, „kann“ und „optional“ in allen folgenden Abschnitten sind gemäss RFC 2119 zu interpretieren. (Bradner, 1997)

## Use-Cases

Das nachfolgende Diagramm stellt die wichtigsten Anwendungsfälle dar, die aus den Vorgehensschritte 1-3 (Kapitel 3.2) definiert und in Vorgehensschritt 4 (Kapitel 3.2) ausgewählt wurden. Es dient dazu die eigentliche Funktionalität des zu entwickelnden Systems zu erkennen und in den Anforderungen klar zu spezifizieren.

Im Nachfolgenden werden alle Use-Cases aufgelistet die im Rahmen des Projektes gefunden wurden.

### Akteure

|  |  |
| --- | --- |
| User | Popcorn Kocher |
| Administrator | Der Administrator hat alle Rechte des Users. Kann aber zusätzlich noch User verwalten. |
| Präsentationsmedium | Das Präsentationsmedium kann sich an eine aktuelle Präsentation anmelden und Präsentationsinformationen empfangen |

# Analyse

## Zu optimierende Prozesse

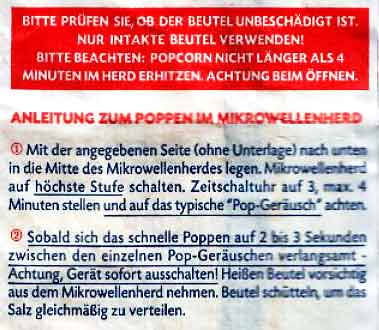


Abbildung - Anleitung Zubereitung Mikrowellen Popcorn

Der Zubereitungs-Prozess von Popcorn in einer Mikrowelle ist auf der Packung definiert. Nun sollen die zu automatisierenden optimierenden Teile dieses Workflows definiert werden. Dafür wird in einem ersten Schritt der Workflow des Zubereitungs-Prozess mit einem Ablaufs UML Diagramm visualisiert.

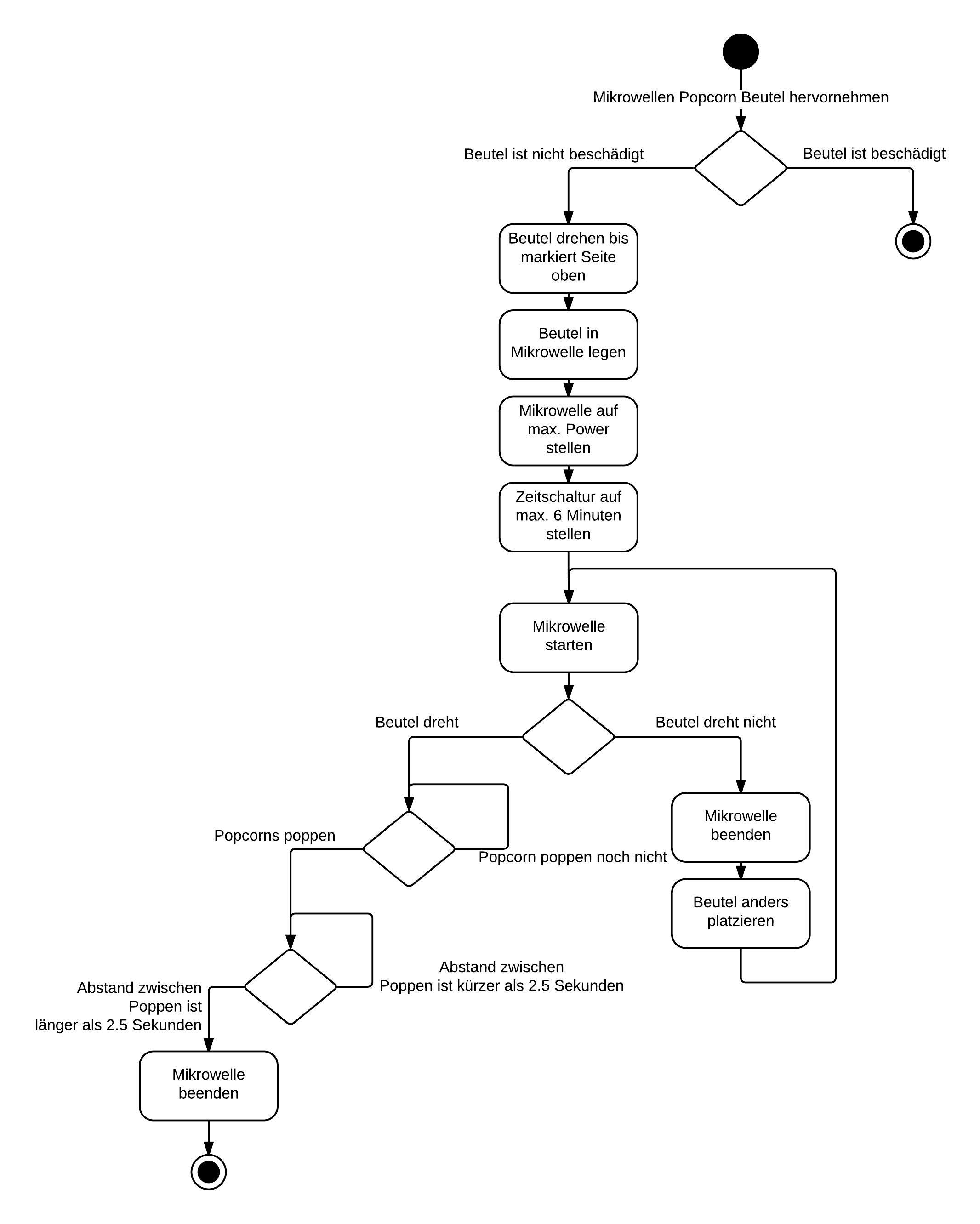


Abbildung - Ablaufdiagramm Zubereitung Mikrowellen Popcorn

Diese ersten Schritte, bis der Beutel von der Mikrowelle erwärmt wird, sind Straight Forward. Diese Schritte werden nicht als Probleme in Forums oder auf Social Media genannt. Diese oberflächliche Erkenntnis wird psychologisch unterstützt. Psychologische Studien haben bewiesen, dass die Fehlerraten bei Menschen besonders hoch sind, wenn ein Workflow durch Wartezeiten / Langweile unterbrochen wird. [[1]](#footnote-1) Bei den ersten Schritten ist der Mensch beschäftigt und in Bewegung. Sowohl körperlich wie auch geistig wird er gefordert.  
Während des Popcorn-Wärmeprozess steht oder sitzt der Mensch wartend vor der Mikrowelle. Seine Konzentration sinkt und die Gedanken schweifen ab. Die Versuchung etwas anderes während der Wartezeit zu erledigen ist gross. Durch Ausführen von anderen Prozessen sinkt die Konzentration und Priorisierung für den Popcorn-Wärmeprozess drastisch. Zusätzlich kommt dazu, dass es schwierig ist, einzuschätzen wie lange der Abstand zwischen dem Poppen ist. Insbesondere weil die Zeit sich während dem Warten gefühlsmässig verlängert. Der Mensch nimmt also kürzere Zeiten länger wahr. Durch das geht es länger bis der Mensch findet, dass das der Poppabstand nun nur noch 2.5 Sekunden dauert.   
Die in den letzten Sätzen erklärten Umstände werden durch das Teilen von negativen Erfahrungen in Forums oder Social Media untermauert. Sätze wie „Ich habe die Popcorns vergessen“ oder „Das waren niemals nur 2.5 Sekuden abstand. FAIL!“ sind Beispiele für Bildbeschreibungen auf den verschiedenen Kanälen im Web.  
Nun gilt es also diese zwei repetitiven Schritte maschinell zu optimieren.

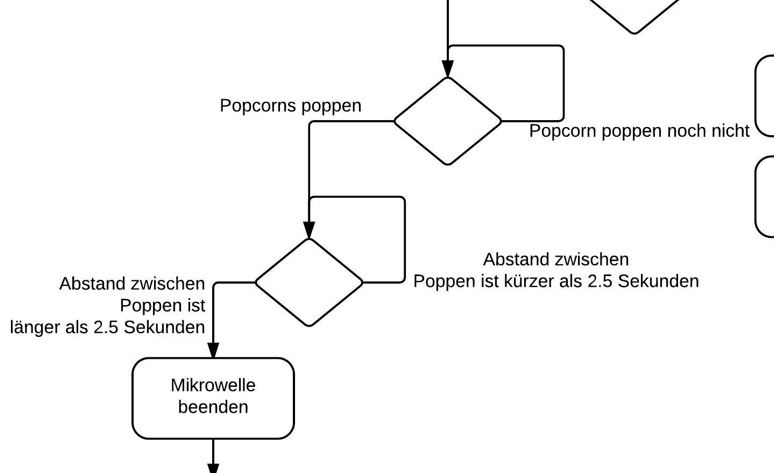


Abbildung Zu optimierenden Schritte

## EVA-Prinzip

Um den Prozess zu untersuchen bedienen wir uns dem EVA-Prinzip. Eingabe, Verarbeitung, Ausgabe. Wir analysieren wie der Mensch bei den Prozessen die Eingabe erhält wie er diese verarbeitet und welche Ausgabe der Mensch gibt. Dabei erhoffen wir an Erkenntnisse zu gelangen um bei den Prozessen maschinell den Input zu erlangen, ihn maschinell zu verarbeiten und maschinell eine Ausgabe zu tätigen.

### Eingabe

Ausgangslage des Prozesses ist, dass die Popcorns in der Mikrowelle am Wärmen sind. Der Mensch nimmt irgendwann war, dass die Popcorns poppen und sollte dann wahrnehmen wenn der Abstand zwischen zwei Poppen weniger als 2.5 Sekunden beträgt. Die Popcorns werden verschlossen in einem Beutel gewärmt. Alle anderen Sinne liefern auch keine genauen Werte. Einzig das Gehör kann das Poppen zuverlässig wahrnehmen. Die Eingabe erhält der Mensch über das Gehör. Daher ist es naheliegend das die Eingabe von der Maschine per Mikrofon möglich wäre.

### Verarbeitung

Die wahrgenommenen Geräusche werden vom Menschlichen Hirn gefiltert. Die Abstände zwischen werden mittels Zeitgefühl gemessen. Messungen von Abständen kann eine Maschine dank Time Clock genau. Die Erkennung von Geräuschen ist da durchaus schwieriger.

### Ausgabe

Der Mensch drückt mittels Hände auf den Ausschalte-Knopf. Um den Prozess 1:1 seitens Maschine auszuführen bräuchte man Robotics-Instrumente. Dieses Variation können wir direkt ausschliessen. Generell können wir das Ziel hinter der Ausgabe analysieren. Es geht darum die Mikrowelle auszuschalten als vom Strom zunehmen. Eine Maschine kann also durchaus den Strom unterbrechen. Eine weitere Variante wäre die Ausgabe mittels Alarm vom Menschen an die Maschine weiterzuleiten.

## Verarbeitung von Mikrofonsignalen

Aus Kapitel 4.2.1 und 4.2.2 resultiert die Anforderung, dass PerfectPopcorn aus Mikrofonsignal entgegennehmen, verarbeiten und das Pop-Geräusch erkennen muss.   
Um ein Abbild von Frequenzen des Audiosignals zu machen, wird ein Spektrumanalysator eingesetzt. Wir erhoffen uns, dass ein Abbild des Pop-Geräuschs spezielle Merkmale aufweist um es von Umgebungsgeräuschen abzugrenzen.

### Spektrumanalysator

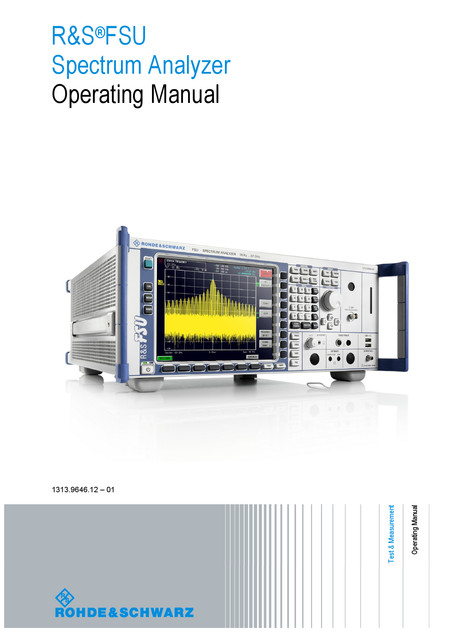


Abbildung - Spektrumanalysator *Quelle home.zhaw.ch*

Der Spektrumanalysator ist ein elektrisches Messgerät. Es stellt die erfassten Signale als Frequenzspektrum dar. Das besser bekannte Oszilloskop stellt im Gegensatz vom Spektrumanalysator den Signalverlauf im Zeitbereich dar und nicht das Frequenzspektrum.

### FFT-Analysator

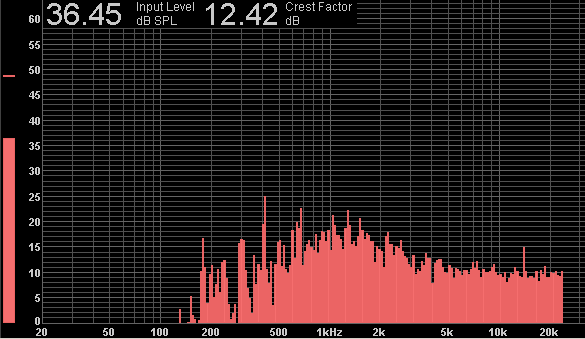


Abbildung - Frequensspektrum eines FFT-Analysators Quelle www.tomshardware.de

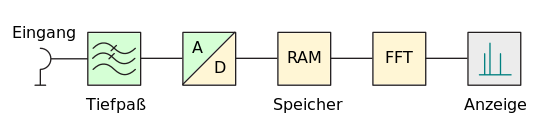


Abbildung - Ablauf FFT-Analyse / Vereinfachtes Blockschaltbild *Quelle Wikipedia*

Der FFT-Analysator macht zuerst eine Tiefpassfilterung um Störsignale zu vermeiden. Danach wird das analoge Signal in ein digitales Signal verwandelt. Das digitale Signal wird im RAM zwischen gespeichert. Mit Hilfe der Fourier-Transformation kann aus den Daten das Frequenzspektrum errechnet werden. Üblicherweise wird die schnelle Fourier-Transformation verwendet. Daher auch die Abkürzung FFT für Fast Fourier Transformation. Die Ergebnisse der FFT werden danach visuell wiedergegeben. Beim dargestellte Frequensspektrum ist die horizontale Achse, auch Abszisse genannt, die Frequenzachse und die vertikale Achse, auch Ordinate genannt. die Amplitude der Frequenz. Dadurch können die Frequenzen fast Echtzeit analysiert werden. Dieser Vorgang wiederholt einige Male pro Sekunde.

**Ein Spektrumanalysator soll nun innerhalb von PerfectPopcron nachgebaut werden.**

## Technische Umgebung

Im Kapiel 4.2 und 4.3 wurden die Anforderungen für die maschinelle Optimierung erörtert. Unser maschinelles System sollte daraus folgend per Mikrofon die Audiosignale entgegennehmen können, diese digitalisieren, zwischenspeichern, eine FFT-Analyse durchführen, diese optisch darstellen und den Menschen über die fertigen Popcorns informieren und die Mikrowelle abstellen.

Ein Smartphone besitzt ein Mikrofon, RAM, CPU sowie Audioausgabe um generell die ersten Punkte umzusetzen. Ob die Genauigkeit des Mikrofons und der Power des CPU für die FFT reicht wird sich bei der Implementation zeigen.

# Design der Software

In diesem Kapitel soll ein System entworfen werden. Das System soll den Anforderungen, welche in Kapitel 3 definiert wurden, entsprechen.

## Einleitung

Die Software Architektur bildet das Gerüst für die zu entwickelnde Applikation. Sie definiert welche Technologien in welchen Komponenten eingesetzt werden und wie über die Schnittstellen kommuniziert wird.

## Design Patterns

### Target-Action Pattern

Im Target-Action Design Pattern enthalten die Objekte die wichtigen Informationen um eine Message an ein anderes Objekt zu senden sofern ein Event ausgelöst wird.  
Erklärt an einem einfachen Beispiel: Rezept-Verwaltung. Wir möchten ein neues Rezept erfassen. Wir haben einen Rezeptedier-View mit dem Rezept-Formular. In einer anderen View, der Navigation-View, sind Abbruch und Speicher Button enthalten. Der Speicher-Button muss die Rezepteditier-View darüber informieren, wenn er gedrückt wurde. So kann die Rezepteditier-View veranlassen, dass die Informationen gespeichert werden. Um dies umzusetzen braucht der Button 2 Sachen: Ziel (target) und die Aktion-Methode.



Abbildung - Target-Action Pattern *Quelle https://developer.apple.com*

### MVC, Model View Controller

Für die Umsetzung des Webservices soll das MVC-Pattern verwendet werden. Dadurch wird der Code in 3 Teilen unterteil:

* **Model**  
  Im Model wird die Businesslogik abgebildet. Das Model enthält Daten welche zur Darstellung benötigt werden.
* **View**In der View werden die Daten dargestellt. Die View nimmt Benutzerinteraktionen entgegen und leitet sie an den Controller weiter.
* **Controller**Der Controller ist die Steuereinheit. Er dient als Schnittstelle zwischen View und Model. Der Controller führt Benutzeraktionen aus und validiert diese, entscheidet welches View angezeigt werden soll oder ob ein anderer Controller angezeigt werden soll.

## Mockups

### Ansicht App



Abbildung – Mokup der Android App

Die Android App soll übersichtlich alle Informationen darstellen.

Im oberen Bereich sollen Near Real Time die Ergebnisse der FFT (Fast Fourier Transformation) als Frequenzspektrum angezeigt werden. So können die Frequenzspektrum Abbildungen des Poppens und natürlich auch allen anderen Geräuschen mitverfolgt werden.

Die Info-Leiste im oberen Mittel informiert den User ob die Verbindung zum Mikrocontroller steht.

Über die Button-Leiste soll der User den Prozess starten und beenden könnenAblauf Präsentation mit ShowkeyImplementation

# Implementation

## Technologien

### Adobe Air

(Adobe, 2015)  
Mithilfe der *Adobe AIR*-Laufzeit können Entwickler den gleichen Code in native Anwendungen und Spiele sowohl für Windows und Mac OS-Desktop-Computer als auch iOS- und Android-Geräte einbinden und somit über eine Milliarde Desktopsysteme und mobile App Stores für über 500 Millionen Geräte erreichen.

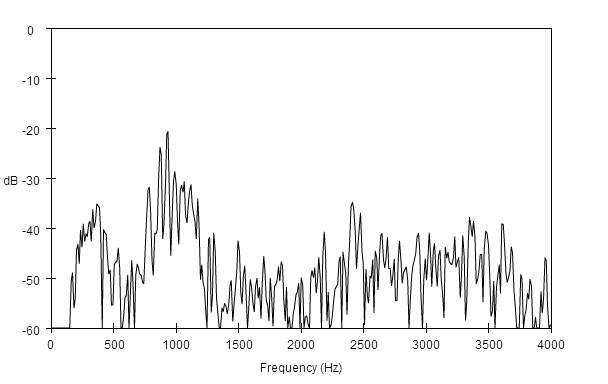
### **ActionScript 3**

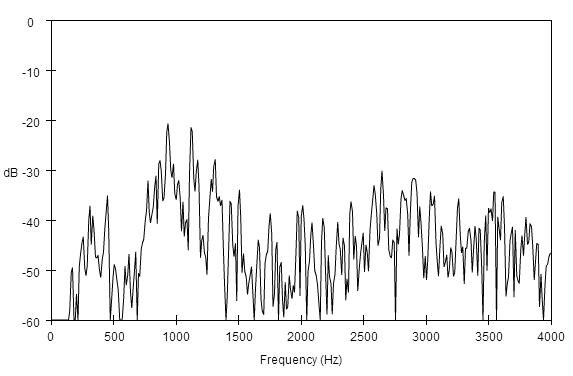
(Adobe Learning, 2015)  
*ActionScript 3* wird umgangsprachlich *AS3* bezeichnet und ist die Programmiersprache für Runtime Environments *Adobe Air* und *Adobe Flash Player*. *ActionScript 3* wurde 2006 von Adobe eingeführt. Es unterscheidet sich komplett von den Versionen 1 und 2. *AS 3* ist eine objektorientierte Programmiersprache und enthält alle Grundprinzipen der Objektorientiertheit.

## Ansteuerung des Mikrofons

## Auswertung des Frequenzsspektrums

Die entstandenen Frequenzsspektrums aus den Popcorn-Pop Tönen gilt es nun zu analysieren und möglichst einfach von den Umgebungsgeräuschen abzugrenzen.





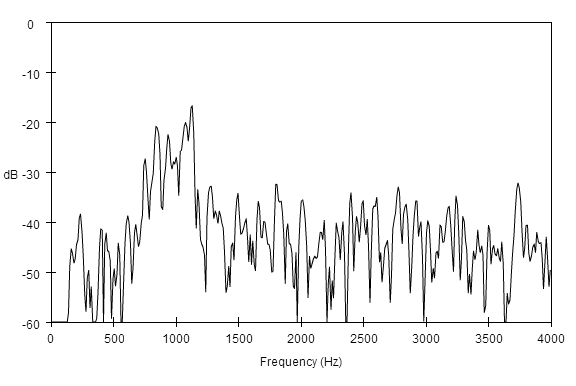


Abbildung - 3 Screenshots der Popcorn Pop Töne

### Feststellung

Das Poppen erhöht die Amplitude fast durchgehend über die alle gemessenen Frequenzen. Im Bereich von 1000 Herz ist es besonders hoch.

Test

Da während der Entwicklung viel Wert auf eine stabile Implementation gelegt wird, wird der Prototyp mehrstufig getestet. In den nachfolgenden Kapiteln, werden die unterschiedlichen Methoden kurz erläutert.

## Unit-Testing (Backend)

Django bringt bereits Unit-Testing Fähigkeiten mit. So können beliebige Testfälle erstellt und ausgeführt werden. Dabei wird für eine Testsuite jeweils eine eigene frische InMemory-Datenbank bereitgestellt. So ist sichergestellt dass die Testergebnisse reproduzierbar und schnell zur Verfügung stehen.

Neben den Unit-Testing Fähigkeiten, können auch Testfälle erstellt werden, welche eine komplette Django-Instanz starten und per HTTP darauf zugreifen. Auf diese Weise können Integrationstests erstellt werden, die auch das Zusammenspiel mit dritten Applikationen (Plug-Ins) testen.

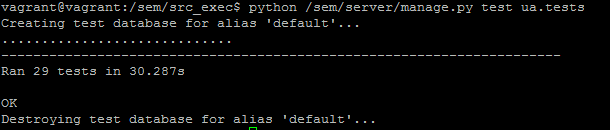


Abbildung 12 - Unittests - Backend

## Unit-Testing (Frontend)

Mit Mocka können einzelne Funktionen und Funktionsgruppen sehr einfach getestet werden. Durch den Einsatz von Sinon können auch Komponenten und Schnittstellen auf effektive Art und Weise getestet werden.

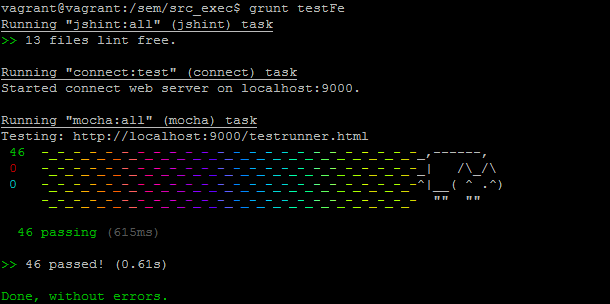


Abbildung - Unittets - Frontend

## Integration-Testing

Da die Entwicklungsumgebung auch eine GLPI Instant zur Verfügung stellt, können Integrationstests sehr realitätsnah und beliebig oft durchgeführt werden.

Um das Zusammenspiel zwischen Backend und Frontend zu verifizieren, wird Selenium eingesetzt. Selenium kann in den Django-Test-Suiten eingebunden werden. Dies ermöglicht eine automatisierte benutzerähnliche Bedienung eines Browsers und die Verifikation des angezeigten Inhalts auf verlässliche Art und Weise.

Da Seleniumtests eine laufende Instanz eines Browser benötigen, der Browser jedoch eine grafische Oberfläche besitzt und daher nicht von der Konsole ausgeführt werden kann, wird mit pyVirtualDisplay ein Bildschirm simuliert. Damit jedoch trotzdem für jeden Test auch das sichtbare Resultat nachgeprüft werden kann, werden Screenshots erstellt und gespeichert. Nachfolgend der Screenshot nach erfolgreichem Login, während die Daten noch geladen werden.

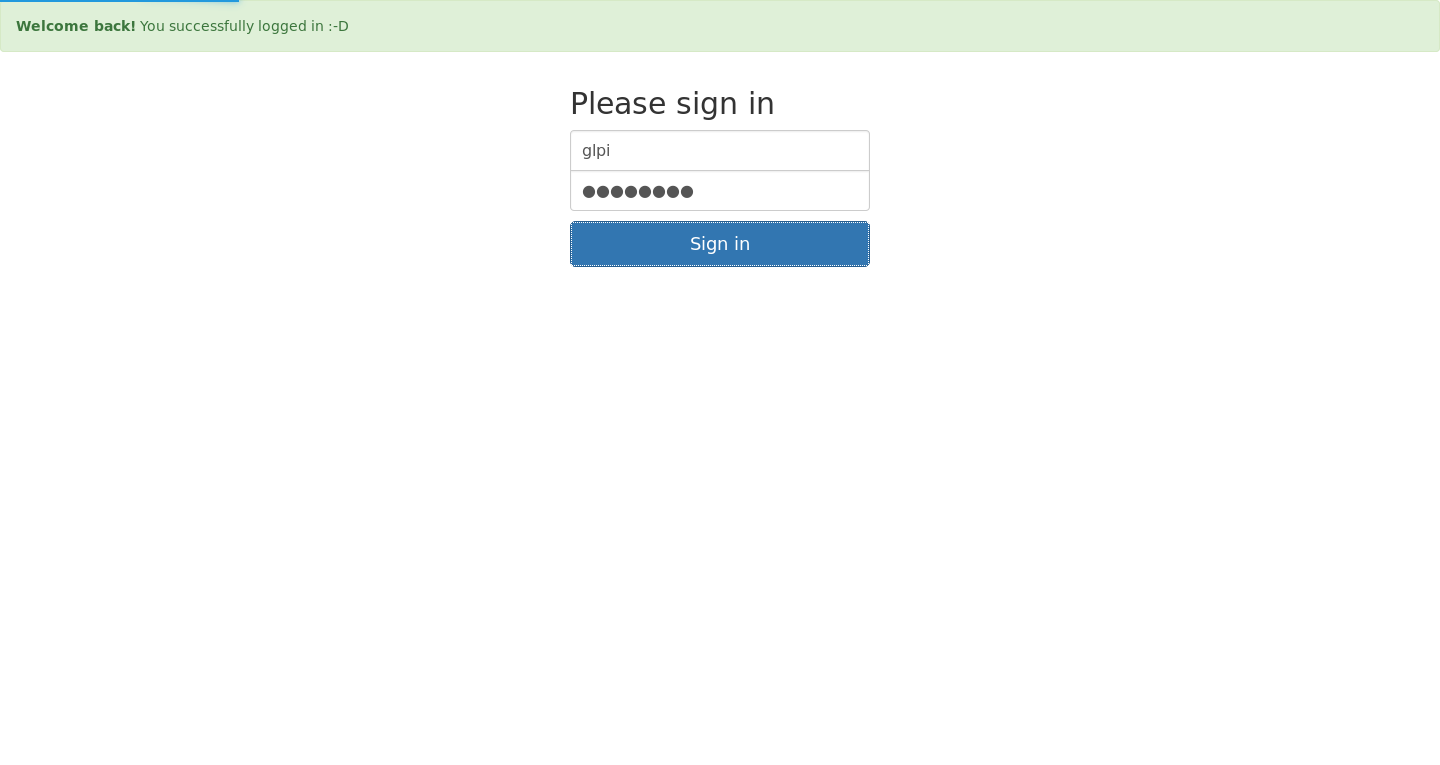


Abbildung - Screenshot Integration-Test

## Test der Akzeptanzkriterien

Anschliessend sind die Tests der Akzeptanzkriterien aufgeführt.

|  |  |  |  |
| --- | --- | --- | --- |
| Kriterium | manuell | automatisch |  |
| AC-01 Ticket verwalten |  | x |  |
| AC-02 Ticket synchronisieren |  | x |  |
| AC-03 Berechtigung aus GLPI |  | x |  |
| AC-04 Zusätzliche ticketbezogene Informationen | x |  |  |
| AC-05 Ticket bearbeiten |  | x |  |
| AC-07 Anmeldung mittels AD-User |  | x |  |
| AC-08 Work-Queue Abwesenheiten |  | x |  |
| AC-09 Work-Queue Anzeige I |  | x |  |
| AC-10 Work-Queue Anzeige II |  | x |  |
| AC-11 Priorisierung |  | x |  |
| AC-14 Zugänglichkeit | x |  |  |
| AC-15 Benachrichtigung | x |  |  |

## Überprüfung Aufgabestellung

Nachfolgend wird jeder Punkt der Aufgabenstellung überprüft.

|  |  |  |  |
| --- | --- | --- | --- |
| Aufgabenstellung | Kapitel / Code | Status | Datum |
| 01 Anforderungsanalyse | 3 Anforderungsanalyse | Ok | 09.02.2014 |
| 02.1 Findung der Anforderungen | 3 Anforderungsanalyse | Ok | 16.03.2014 |
| 02.2 Analyse der Anforderungen | 3 Anforderungsanalyse | Ok | 16.03.2014 |
| 03 Schnittstelle zum Ticketsystem (GLPI) | Code | Ok | 09.02.2014 |
| 04 Daten können gelesen und geschrieben werden. | Code | Ok | 09.02.2014 |
| 05 Konzeption mindestens einer weiteren Schnittstelle zu einem Umsystem | **Fehler! Verweisquelle konnte nicht gefunden werden.** **Fehler! Verweisquelle konnte nicht gefunden werden.** | Ok | 16.03.2014 |
| 06 Proof of concept der Datenverarbeitung im Backend | Code | Ok | 09.02.2014 |
| 07 Konzeption sowie teilweise Implementierung sodass rudimentäre Funktionstests durchgeführt werden können | Code | Ok | 09.02.2014 |
| 08. teilweise Implementierung der Priorisierung der Arbeitspakete | Code | Ok | 16.03.2014 |
| 09 teilweise Implementierung der Job-Queue | Code | Ok | 16.03.2014 |
| 10 teilweise Implementierung Ticketplanung | Code | Ok | 16.03.2014 |
| 11 Konzeption der vollständigen Nachvollziehbarkeit jedes Arbeitspakets | Code | Ok | 16.03.2014 |
| 12 Proof of concept der API zum Backend | Code | Ok | 16.03.2014 |
| 13 Konzeption sowie teilweise Implementierung sodass rudimentäre Funktionstests durchgeführt werden können | Code | Ok | 16.03.2014 |
| 14 Ein rudimentäres Webinterface zur Bedienung des Backend durch die API | Code | Ok | 09.02.2014 |
| 15 Es ist möglich neue Arbeitspakete zu erstellen sowie Arbeitspakete zu bearbeiten | Code | Ok | 16.03.2014 |
| 16 Es ist möglich die aufgewendete Arbeitszeit pro Arbeitspaket darzustellen | Code | Ok | 16.03.2014 |

# Fazit und Schlusswort

# Glossar

**Code First**  
Die zielgerichtete Entwicklung für eine Datenbank, die noch nicht vorhanden ist und mittels Code First erstellt wird.   
*Microsoft MSDN*

**Entity Framework**  
Entity Framework ist die empfohlene Datenzugriffstechnologie von Microsoft für neue Anwendungen.  
*Microsoft MSDN*

**Github**Ein Cloud basierter Sourcecode Verwaltungs Dienst der Git einsetzt.  
*https://github.com/*

**Liedteile**  
Ein Liedteile prägen den Aufbau eines Liedes. Beispiele dafür sind Strophe, Refrain, Bridge, Liedteil 1, Liedteil 2,… Liedteile können mehrfach in einem Lied vorkommen oder mehrfach wiederholt werden.

**WEB-API**  
API steht für WEB Application Programming Interface.  
*http://en.wikipedia.org/wiki/Web\_API*

# Abbildungsverzeichnis

[Abbildung 1 - Projektplan 9](#_Toc418498064)

[Abbildung 2 Basis Schablone (Rupp, 2011) 18](#_Toc418498065)

[Abbildung 3 - Erweiterte Schablone (Rupp, 2011) 18](#_Toc418498066)

[Abbildung 4 - Bausteinübersicht 31](#_Toc418498067)

[Abbildung 5 - Browser Kompatibilität Websockets, rot=keine Kompatibilität, hellgrün eingeschränkte Kompatibilität, grün volle Kompatibilität Quelle http://caniuse.com/#feat=websockets 33](#_Toc418498068)

[Abbildung 6 Visualisierung SignalR API Quelle asp.net/signalr 34](#_Toc418498069)

[Abbildung 7 – Mokup der Startseite von Worshipbeamer.net Verwaltung mit Login 36](#_Toc418498070)

[Abbildung 8 - Mokup Start-Lighbox von Lieder editieren 37](#_Toc418498071)

[Abbildung 9 Mockup Lieder editieren 38](#_Toc418498072)

[Abbildung 10 - Mokup der Präsentation Bedienung 39](#_Toc418498073)

[Abbildung 11 - Mockup Präsentationsansicht 40](#_Toc418498074)

[Abbildung 12 - Ablauf Präsentationsansicht 41](#_Toc418498075)

[Abbildung 13 - Ablauf Präsentationsansicht mit Cookie 42](#_Toc418498076)

[Abbildung 14 - Ablaufsdiagramm App Launcher 46](#_Toc418498077)

[Abbildung 15 Hauptsoftware Komponenten für Authentification 49](#_Toc418498078)

[Abbildung 16 - Ablauf traditionelle Authentifizierung Quelle (Babal, 2014) 49](#_Toc418498079)

[Abbildung 17 - Ablauf Token-basierte Authentifizierung Quelle (Babal, 2014) 50](#_Toc418498080)

[Abbildung 18 - Unittests - Backend 51](#_Toc418498081)

[Abbildung 19 - Unittets - Frontend 51](#_Toc418498082)

[Abbildung 20 - Screenshot Integration-Test 52](#_Toc418498083)

# Tabellenverzeichnis

[Tabelle 1- Soll-Ist Vergleich 10](#_Toc418498084)

[Tabelle 2 - Beteiligte Personen 10](#_Toc418498085)

[Tabelle 3 - Stakeholder 11](#_Toc418498086)

[Tabelle 4 - Risikomatrix 29](#_Toc418498087)

# Literaturverzeichnis

Babal, H. (11. 12 2014). *tutsplus.com*. Von tutsplus.com: http://code.tutsplus.com/tutorials/token-based-authentication-with-angularjs-nodejs--cms-22543 abgerufen

Bradner, S. (1997). *RFC 2119.* Cambridge.

*Introduction to SignalR*. (04. 02 2015). Von http://www.asp.net/signalr abgerufen

Rupp, K. P. (2011). *Basiswissen Requirements Engineering.* dpunkt.verlag.

Weyer, C. (19. 01 2015). *henriquat.re*. Von http://henriquat.re/server-integration/signalr/integrateWithSignalRHubs.html abgerufen

*Wikipedia - Websockets*. (21. 02 2015). Von http://de.wikipedia.org/wiki/WebSocket abgerufen

# Bestätigung über die Selbständigkeit

Hiermit versichere ich, die vorliegende Semesterarbeit eigenständig und ausschliesslich unter Verwendung der angegebenen Hilfsmittel angefertigt zu haben.

Alle öffentlichen Quellen sind als solche kenntlich gemacht. Die vorliegende Arbeit ist in dieser oder anderer Form zuvor nicht als Semesterarbeit zur Begutachtung vorgelegt worden.

Aadorf 30.05.2015

Christian Bachmann

1. (Althof, 199) [↑](#footnote-ref-1)