

**Basis der Desktoplösung PowerPraise durch webbasierte AngularJS-Websocket-App ersetzen**

**Semesterarbeit**im Studiengang Informatik

**Christian Bachmann**an der ZHAW Zürich

Betreuer: Jaime Oberele

Auftraggeber: Technikteam Chrischona Aadorf

# Abstract

Diese Arbeit handelt von der Erstellung einer Liederpräsentation Lösung. Die Lösung Worshipbeamer.net soll Verwaltung von Lieder & Events, Präsentation auf verteilten Plattformen und Präsentation auf kostengünstigen Medium ermöglichen. Anhand einer umfassenden Anforderungsanalyse wird ein Prototyp entworfen und umgesetzt. Die Anforderungsanalyse wird anhand von definierten Kriterien strukturiert, um so nur die für den Auftraggeber wichtigen Anforderungen im Rahmen dieser Arbeit umzusetzen.

Der Student konnte zeigen, dass die gefundenen Anforderungen umgesetzt werden konnten. Der Prototyp zeichnet sich durch eine hohe Modularität, gute Testbarkeit und Flexibilität aus. Durch den Einsatz modernster Web-Technologie ist der Prototyp für die Zukunft gewappnet.

# Inhaltsverzeichnis

[Abstract 1](#_Toc418497937)

[Inhaltsverzeichnis 2](#_Toc418497938)

[1 Einleitung 5](#_Toc418497939)

[1.1 Vision 5](#_Toc418497940)

[1.2 Motivation 5](#_Toc418497941)

[1.3 Ist-Zustand 5](#_Toc418497942)

[1.4 Aufgabenstellung 6](#_Toc418497943)

[1.4.1 Thema 6](#_Toc418497944)

[1.4.2 Ausgangslage 6](#_Toc418497945)

[1.4.3 Ziel der Arbeit 6](#_Toc418497946)

[1.4.4 Aufgabenstellung 6](#_Toc418497947)

[1.4.5 Erwartete Resultate 7](#_Toc418497948)

[2 Projektmanagement 8](#_Toc418497949)

[2.1 Werkzeuge 8](#_Toc418497950)

[2.2 Projektplan 9](#_Toc418497951)

[2.3 Soll / Ist Analyse 10](#_Toc418497952)

[2.4 RandBedienungen 10](#_Toc418497953)

[2.4.1 Zeitlicher Rahmen 10](#_Toc418497954)

[2.4.2 Organisatorischer Rahmen 10](#_Toc418497955)

[2.4.3 Stakeholder und Zielgruppe 11](#_Toc418497956)

[3 Anforderungsanalyse 12](#_Toc418497957)

[3.1 Einleitung 12](#_Toc418497958)

[3.2 Vorgehensweise 12](#_Toc418497959)

[3.3 Use-Cases 13](#_Toc418497960)

[3.3.1 Akteure 13](#_Toc418497961)

[3.3.2 UC-11 User Administration 13](#_Toc418497962)

[3.3.3 UC-12 Verschiedene Modus aufrufbar 13](#_Toc418497963)

[3.3.4 UC-21 Präsentation aktivieren 14](#_Toc418497964)

[3.3.5 UC-22 Präsentationsmedium anmelden 14](#_Toc418497965)

[3.3.6 UC-23 Kommunikation Bedienung- und Präsentationsmodus 15](#_Toc418497966)

[3.3.7 UC-31 Bearbeitung von Liedern 16](#_Toc418497967)

[3.3.8 UC-32 Lieder übersetzen 16](#_Toc418497968)

[3.3.9 UC-40 Verwalten eines Events 17](#_Toc418497969)

[3.3.10 UC-50 Wartbarkeit 17](#_Toc418497970)

[3.4 Anforderungen 18](#_Toc418497971)

[3.4.1 Die Anforderungen sollen basierend auf der Satzschablone erstellt werden. Ziel ist sprachliche Missverständnisse dadurch zu vermeiden. Die Schablone fördert eine syntaktische Eindeutigkeit der Anforderungen und eine optimalen Zeit- und Kostenrahmen für die Verfassung. 18](#_Toc418497972)

[3.5 Aufbau 18](#_Toc418497973)

[3.6 Funktionale Anforderungen 18](#_Toc418497974)

[3.6.1 FREQ-101 User erfassen 19](#_Toc418497975)

[3.6.2 FREQ-102 User bearbeiten 19](#_Toc418497976)

[3.6.3 FREQ-103 User löschen 19](#_Toc418497977)

[3.6.4 FREQ-104 Autorisierter Zugriff 19](#_Toc418497978)

[3.6.5 FREQ-110 Worshipbeamer Modus wechseln 19](#_Toc418497979)

[3.6.6 FREQ-210 Präsentation aktivieren 19](#_Toc418497980)

[3.6.7 FREQ-211 Präsentationsmedium mit Bedienung verknüpfen 20](#_Toc418497981)

[3.6.8 FREQ-212 Präsentation auf Präsentationsmedium starten ohne manueller Input 20](#_Toc418497982)

[3.6.9 FREQ-213 Liedteile präsentieren 20](#_Toc418497983)

[3.6.10 FREQ-214 Pfeiltasten-Steuerung 20](#_Toc418497984)

[3.6.11 FREQ-215 Liedersuche 20](#_Toc418497985)

[3.6.12 FREQ-310 Neues Lied erstellen 20](#_Toc418497986)

[3.6.13 FREQ-311 Lied löschen 21](#_Toc418497987)

[3.6.14 FREQ-312 Titel des Liedes anpassen 21](#_Toc418497988)

[3.6.15 FREQ-320 Liedteile zu einem Lied erfassen 21](#_Toc418497989)

[3.6.16 FREQ-321 Liedteil-Namen bearbeiten 21](#_Toc418497990)

[3.6.17 FREQ-322 Liedteil löschen 21](#_Toc418497991)

[3.6.18 FREQ-330 Slide erstellen 21](#_Toc418497992)

[3.6.19 FREQ-331 Slide bearbeiten 21](#_Toc418497993)

[3.6.20 FREQ-332 Slide übersetzen 22](#_Toc418497994)

[3.6.21 FREQ-333 Slide löschen 22](#_Toc418497995)

[3.6.22 REQ-334 Slide Reihenfolge verändern 22](#_Toc418497996)

[3.6.23 FREQ-340 Liedablauf mit Liedteilen definieren 22](#_Toc418497997)

[3.6.24 FREQ-341 Liedteile aus Liedablauf lösche 22](#_Toc418497998)

[3.6.25 FREQ-342 Reihenfolge Liedteile in Liedablauf ändern 22](#_Toc418497999)

[3.6.26 FREQ-411 Events erstellen 23](#_Toc418498000)

[3.6.27 FREQ-412 Name des Events verändern 23](#_Toc418498001)

[3.6.28 FREQ-413 Event löschen 23](#_Toc418498002)

[3.6.29 FREQ-421 Lieder einem Event hinzufügen 23](#_Toc418498003)

[3.6.30 FREQ-422 Lieder von Event entfernen 23](#_Toc418498004)

[3.6.31 FREQ-423 Liederposition innerhalb eines Events verändern entfernen 23](#_Toc418498005)

[3.6.32 FREQ-430 Eventsuche 23](#_Toc418498006)

[3.6.33 NREQ-10 Betriebsystemunabhängig 24](#_Toc418498007)

[3.6.34 NREQ-15 Wartbarkeit 24](#_Toc418498008)

[3.6.35 NREQ-20 Zeitnahe Kommunikation Bedienung versus Präsentation 24](#_Toc418498009)

[3.6.36 NREQ-20 User über Ladezeiten informieren 24](#_Toc418498010)

[3.6.37 NREQ-01.04 Performance 25](#_Toc418498011)

[3.7 Bewertung der Anforderungen 26](#_Toc418498012)

[3.8 Risiken 27](#_Toc418498013)

[3.8.1 R-01 Akzeptanz 27](#_Toc418498014)

[3.8.2 R-02 Kosten 27](#_Toc418498015)

[3.8.3 R-03 Überkomplexität 27](#_Toc418498016)

[3.8.4 R-04 Systemumfeldänderungen 27](#_Toc418498017)

[3.8.5 R-05 Schlechte/Unzureichende Framework 27](#_Toc418498018)

[3.8.6 R-06 Termineinhaltung 28](#_Toc418498019)

[3.8.7 R-07 Auslastung 28](#_Toc418498020)

[3.8.8 Risikomatrix 29](#_Toc418498021)

[3.8.9 Massnahmen 29](#_Toc418498022)

[4 Design der Software 30](#_Toc418498023)

[4.1 Einleitung 30](#_Toc418498024)

[4.2 Design Ansätze 30](#_Toc418498025)

[4.3 Entscheid 30](#_Toc418498026)

[4.4 Aufteilung der Worshipbeamer.net Lösung 31](#_Toc418498027)

[4.5 Near Real Time Kommunikation 32](#_Toc418498028)

[4.5.1 Polling 32](#_Toc418498029)

[4.5.2 Implementation Websockets 32](#_Toc418498030)

[4.5.3 SignalR 34](#_Toc418498031)

[4.6 AngularJS-App Client Aufteilung 35](#_Toc418498032)

[4.7 Worshipbeamer.net Verwaltung Ansicht 36](#_Toc418498033)

[4.7.1 Ansicht Lieder editieren 37](#_Toc418498034)

[4.7.2 Präsentation Bedienung / Eventverwaltung 39](#_Toc418498035)

[4.8 Präsentationsansicht 40](#_Toc418498036)

[4.8.1 Ablauf Präsentationsansicht mit Showkey 41](#_Toc418498037)

[4.8.2 Problematik erneuter Eingabe desselben Showkeys 41](#_Toc418498038)

[4.9 Launcher App 43](#_Toc418498039)

[4.9.1 Herausforderung Automatisches öffnen Webseite 43](#_Toc418498040)

[4.9.2 Herausforderung Hardware und Betriebssystem für Bildschirme/Beamer 43](#_Toc418498041)

[4.9.3 Lösungsansatz 1 – Mini PC 44](#_Toc418498042)

[4.9.4 Lösungsansatz 2 – Android Stick 45](#_Toc418498043)

[4.9.5 Hardware/Betriebsystem Entscheidung 45](#_Toc418498044)

[4.9.6 App Launch Entwicklung 46](#_Toc418498045)

[4.9.6.1 Apllication Lifecycle 46](#_Toc418498046)

[5 Implementation 47](#_Toc418498047)

[6.1 Authentifizierung 49](#_Toc418498048)

[6.1.1 Traditionelle Authentifizierung 49](#_Toc418498049)

[6.1.2 Token-basierte Authentifizierung 50](#_Toc418498050)

[6.1.3 Vorteile Token-basierter Authentifizierung 50](#_Toc418498051)

[7 Test 51](#_Toc418498052)

[7.1 Unit-Testing (Backend) 51](#_Toc418498053)

[7.2 Unit-Testing (Frontend) 51](#_Toc418498054)

[7.3 Integration-Testing 52](#_Toc418498055)

[7.4 Test der Akzeptanzkriterien 53](#_Toc418498056)

[7.5 Überprüfung Aufgabestellung 54](#_Toc418498057)

[8 Fazit und Schlusswort 55](#_Toc418498058)

[Glossar 56](#_Toc418498059)

[Abbildungsverzeichnis 57](#_Toc418498060)

[Tabellenverzeichnis 58](#_Toc418498061)

[Literaturverzeichnis 59](#_Toc418498062)

[Bestätigung über die Selbständigkeit 60](#_Toc418498063)

# Einleitung

## Vision

Eine einfache zu bedienende Anwendung um Lieder in einer Kirche zu publizieren auf einer möglichst einfachen Hardware und Softwareinfrastruktur.

## Motivation

Die Entwicklung des beliebten Liederpräsentationstools PowerPraise wurde eingestellt. Die Kirchen möchten auch längerfristig eine Applikation nutzen welche in derselben Einfachheit die Möglichkeiten des Lieder Publizierens ermöglicht. Die technische Entwicklung hat auch vor den Kirchen nicht halt gemacht. Dadurch ist auch die Vielfalt der Betriebssysteme gestiegen. Auch die Arbeitsweise hat sich verändert so sind Bedürfnisse der Mobilität und Zusammenarbeit gestiegen.  
Ich möchte es wagen mit einer Near Realtime WebbApplikation in Angular JS diese neuen Bedürfnisse zu befriedigen. Dabei möchte ich mich in die Gebiete von Near Realtime, Microsoft Web API und AngularJs einarbeiten.

## Ist-Zustand

Die Entwicklung von Powerpraise wurde kürzlich eingestellt. Weiter muss jede Kirche die Liedtexte selber eingeben, es gibt keine zentrale Datenbank, Windows wird vorausgesetzt und der Beamer muss direkt am Verwaltungscomputer mit einer Dual-Grafikkarte angeschlossen sein. Die Chrischona Aadorf ist auf der Suche nach einer neuen Lösung und hat sich bereit erklärt als Auftraggeber für diese Arbeit zur Verfügung zu stehen.

## 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.

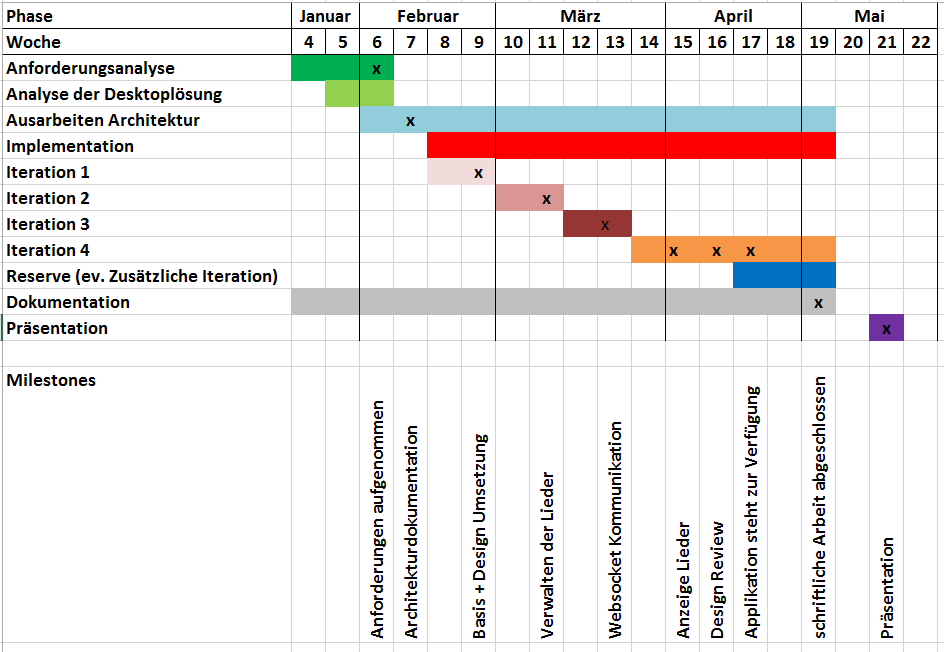


Abbildung 1 - Projektplan

## Soll / Ist Analyse

|  |  |  |
| --- | --- | --- |
| Tätigkeit | Soll Stunden | Ist Stunden |
| Initialisierung | **8** |  |
| Dokumentation | **20** |  |
| Einarbeitung | **18** |  |
| Anforderungsanalyse | **18** |  |
| Design und Architektur | **18** |  |
| Setup Environment | **18** |  |
| Entwicklung | **80** |  |
| Vorbereitung Abgabe | **8** |  |
| Reserve | **0** |  |
| Total | **188** |  |

Tabelle 1- 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 2 - 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

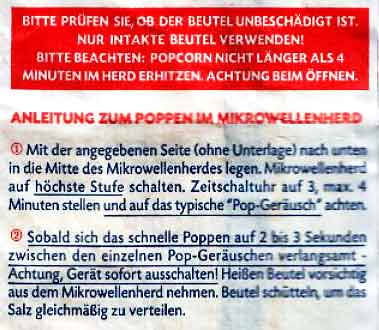
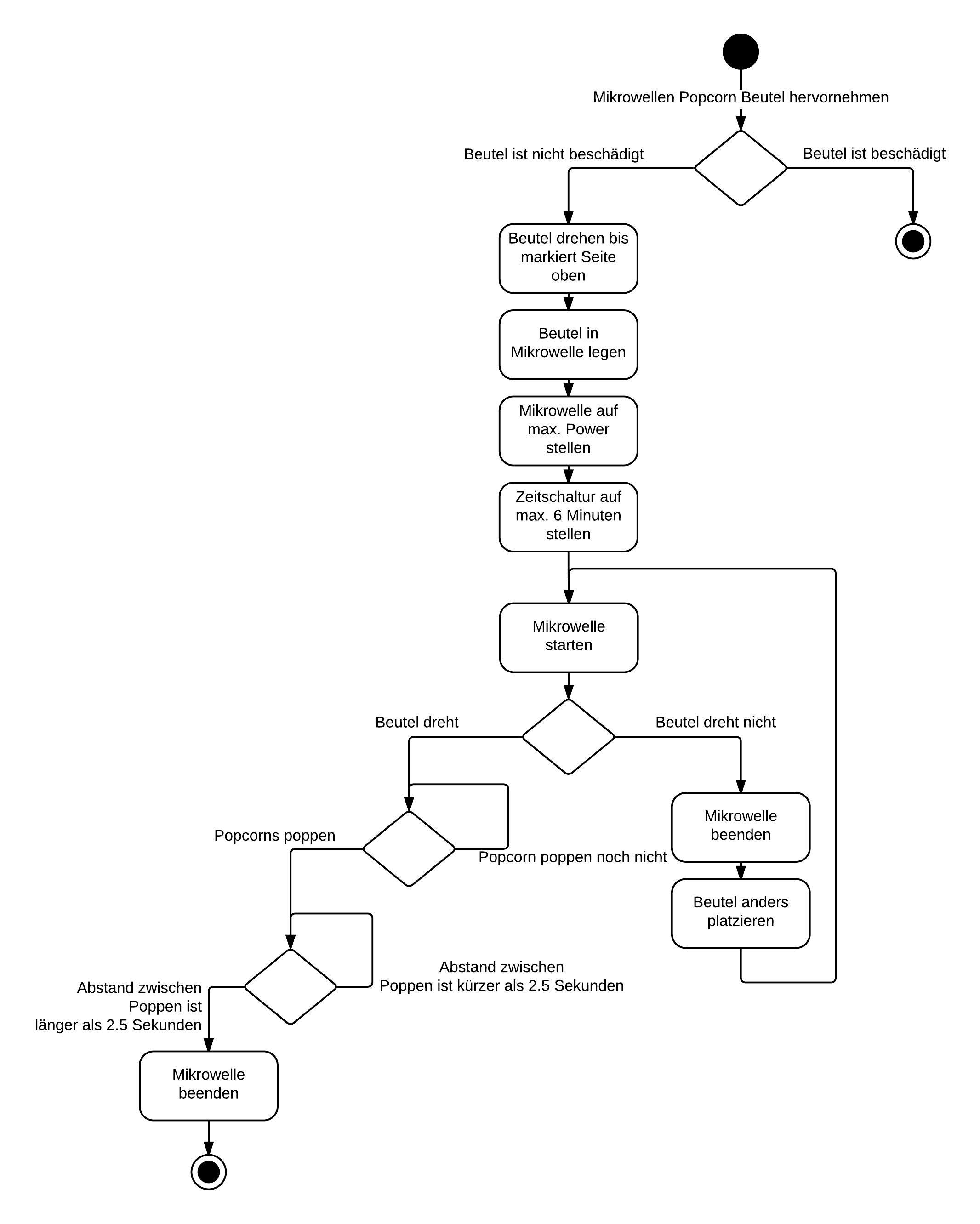


Abbildung 2 - Anleitung Zubereitung Mikrowellen Popcorn

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 3 - 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.

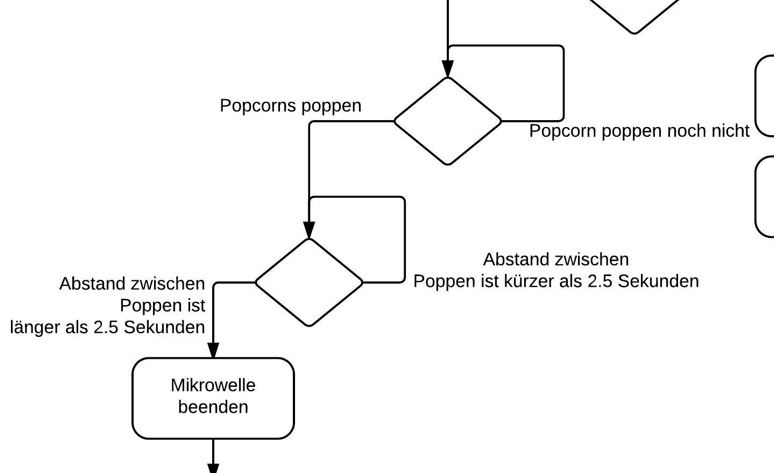


Abbildung 4 Zu optimierenden Schritte

## 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

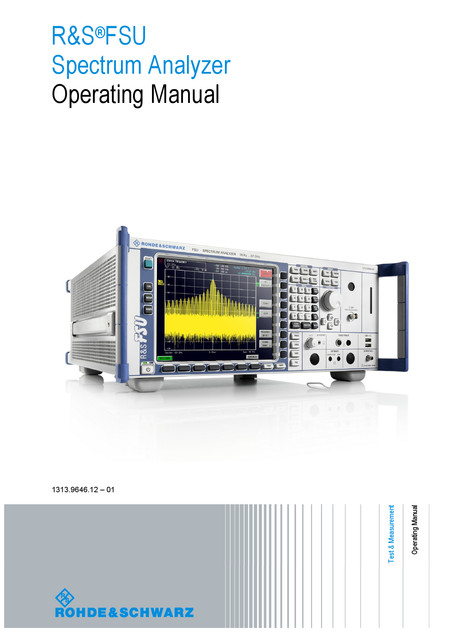


Abbildung 5 - Spektrumanalysator *Quelle home.zhaw.ch*

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

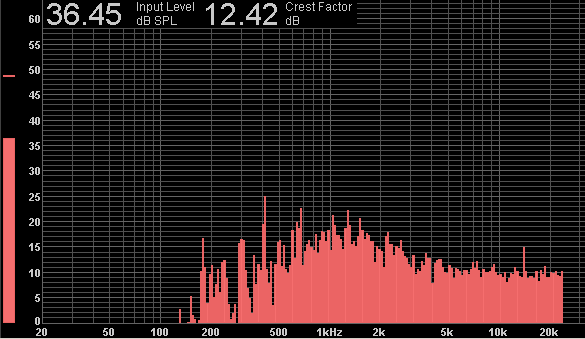


Abbildung 6 - Frequensspektrum eines FFT-Analysators Quelle www.tomshardware.de

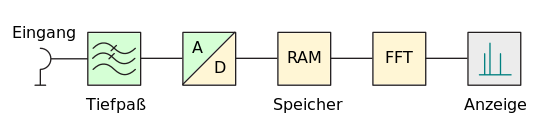
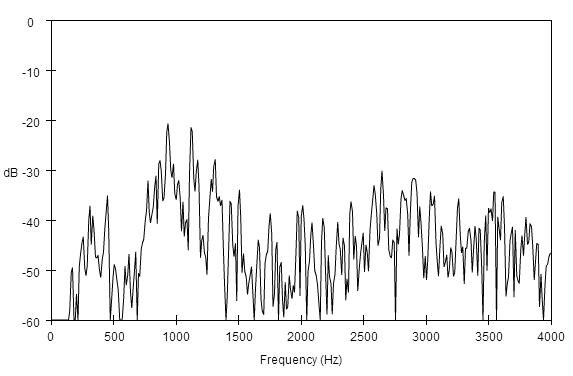
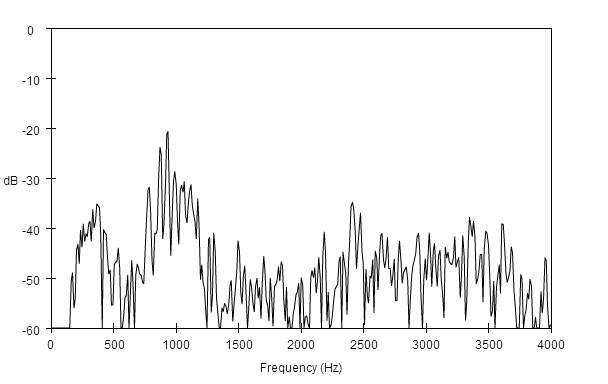
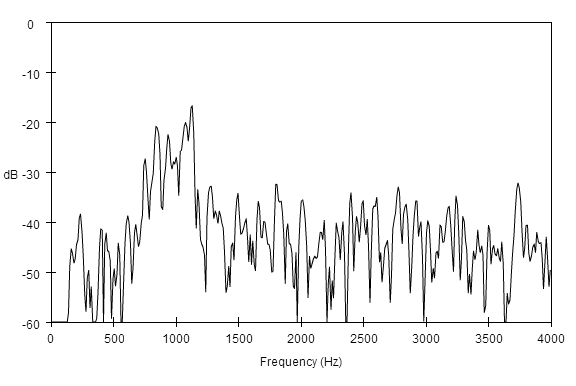


Abbildung 7 - Ablauf FFT-Analyse / Vereinfachtes Blockschaltbild *Quelle Wikipedia*

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.







# 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 8 - 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.

## Wahl des Applikation Hosters

### Asp.net Shared Hosting

Ein Asp.net Shared Hosting ist durchaus für komplexere Webapplikationen wie Worshipbeamer.net ausgerichtet. Die Kosten sind jährlich fix und nicht abhängig von der eigentlichen Nutzung. Überschreitet die Applikation den Speicherbedarf, Zugriffszahlen oder Traffic kann auf ein grösseres Paket aktualisiert werden. Wechsel zu einem kleineren Paket ist meist nur jährlich möglich. Die Skalierbarkeit ist stark eingeschrenkt. Die Daten können so innerhalb der Schweiz gespeichert werden. Der zuständige Systemtechniker ist meist direkt oder indirekt kontaktierbar. Spezielle Konfigurationen am Hosting sind nicht möglich. Die Datencenter sind meist nicht redundant geführt. Fällt das Datencenter aus ist, die Applikation down.

### Azure Hosting

Die Serverkosten sind direkt von der eigentlichen Nutzung abhängig. Die Serverstrutur ist skalierbar und kann sich automatisiert an den aktuellen Nutzungsbedürfnissen anpassen. Die Daten sind in der Cloud redundant geführt. Fällt ein Datencenter aus kann ein anderes dessen Aufgabe übernehmen. Die Daten sind nicht in der Schweiz gespeichert. Die genutzten Serverdienste können komplett an seinen eigenen Bedürfnissen angepasst werden. Die Azure-Cloud wird direkt von der Microsoft betrieben und enthält deshalb immer die neusten Entwicklungen.

### **Entscheidung**

Skalierbarkeit, nutzungsabhängige Kosten, Freiheit in der Serverdienst-Konfiguration überwiegen der Speicherung der Daten in der Schweiz. Ausserdem wird das einfache publishen (veröffentlichen) einer Web-Application aus dem Visual Studio unterstützt, was den Development Workflow erheblich unterstützt.

## Aufteilung der Worshipbeamer.net Lösung

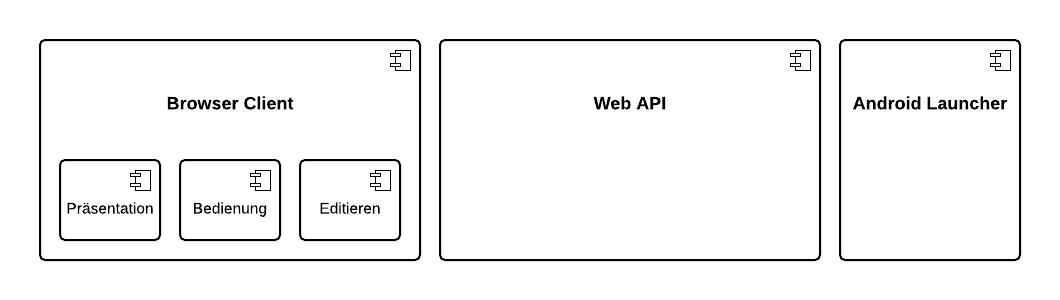


Abbildung 9 - Bausteinübersicht

Die Worshipbeamer.net Lösung setzt sich aus 3 verschiedenen Softwarepaketen zusammen. Die zu erfüllenden funktionalen Anforderungen und Zuständigkeiten unterscheiden die Pakete klar. Um die Anforderungen zu erfüllen wird pro Baustein auf eine komplett andere Technologie gesetzt. Die Begründung der Technologiewahl wird innerhalb jedes Bausteins erklärt.

Browser Client

## AngularJS-App Client Aufteilung

Der AngularJS-Client soll sowohl Präsentationsansicht, Bedienung, Editionsmodus und Administration darstellen. Bedienung, Editionsmodus und Administraion werden im Backend Charakter umgesetzt. Diese Entscheidung beruht darauf, dass in allen Modus Verwaltungstätigkeiten gemacht werden, welche sich im klassischen Backend wiederspiegeln. Die verschiedenen Modus basieren auf derselben Authentifizierung. Deshalb ist es naheliegend, dass diese 3 Modus alle in derselben AngularJs-App umgesetzt werden.

Andere Authentifizierung und Darstellung im Präsentationsmedium  
Die Präsentationsansicht soll über einen Key Authentifiziert werden und wird im Vollbild Präsentationscharakter dargestellt. Nun stellt sich die Frage ob dieser andere Designansatz und Authentifizierungscharakter zu einer eigenen App führt oder ob die Präsentationsansicht in derselben App entwickelt werden soll. Die klaren Unterschiede in der Authentifizierung und Präsentationsansicht sprechen für eine eigene App. Jedoch müssen dazu zwei Deployments aufgebaut werden und der Wartungsaufwand wird dadurch grösser.

Diese Mehraufwendungen führen zum Entscheid, dass auch die Präsentationsansicht mit den anderen Ansichten zu einer App verschmolzen wird. Für die Wiederverwendung wird der Code modular entwickelt und mit dem MVC-Desing Pattern getrennt.

## Mockups Worshipbeamer.net

### Verwaltung Ansicht

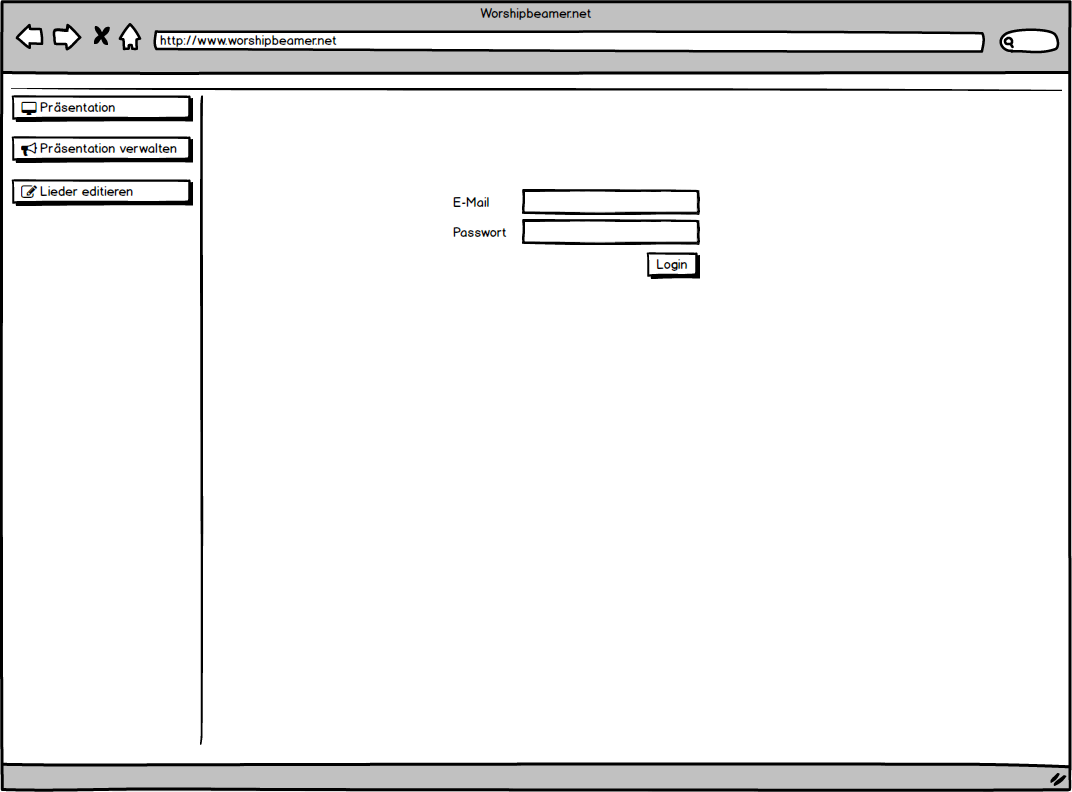


Abbildung 10 – Mokup der Startseite von Worshipbeamer.net Verwaltung mit Login

Die Verwaltungsansicht soll das Verwalten von worshipbeamer.net für Techniker, Band, Pastor, Moderator und allen anderen User oder Administratoren möglichst intuitiv und effizient gestalten. Dabei wurde das Risiko der Akzeptanz (3.8.1 R-01) besonders beachtet. Auf der Startseite kann sich der Benutzer gleich authentifizieren. Die tokenbasierte Authentifizierung wird im Kaptiel 6.1 beschrieben. Wenn der User bereits erfolgreich authentifiziert ist wird er automatisiert Willkommen geheissen.

### Ansicht Lieder editieren

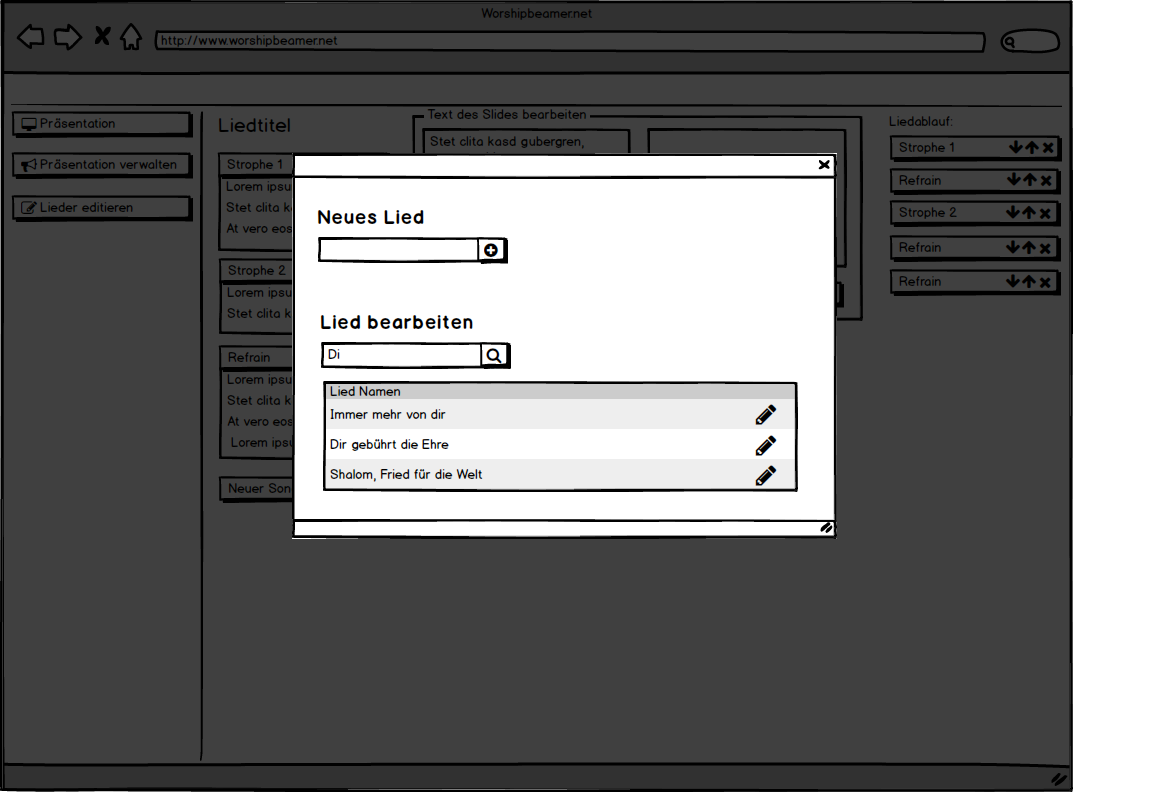


Abbildung 11 - Mokup Start-Lighbox von Lieder editieren

Der User möchte eines der zwei Tasks umsetzten wenn er in den Editiermodus von Liedern geht. Ein neues Lied erfassen oder ein bestehendes bearbeiten. Beide Wege weren, wie im Mokup skizziert, gleich zu Beginn möglich sein. Durch Eingabe eines neuen Liedenames und Bestätigung durch Plus-Button wird das neue Lied erstellt und der User kann beginnen es zu bearbeiten. Alternativ kann er über die Volltextsuche das gewünschte Lied aussuchen und mit der Edition beginnen.

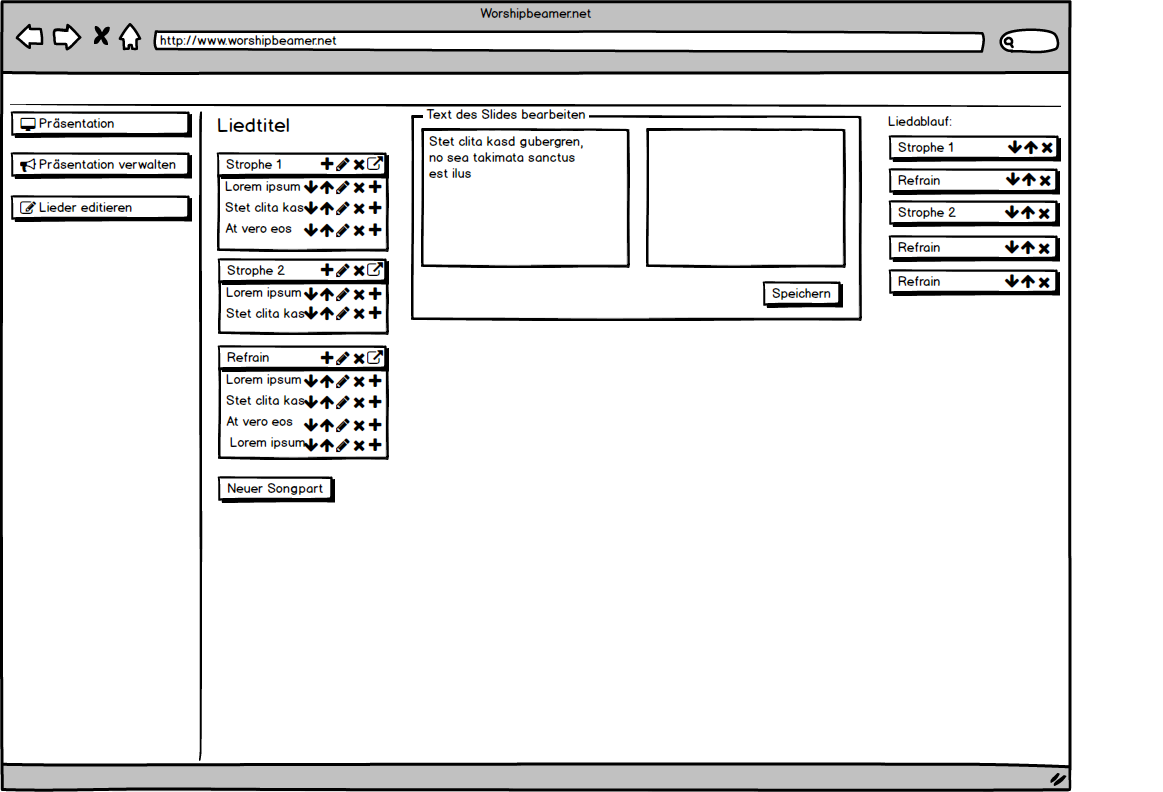


Abbildung 12 Mockup Lieder editieren

Die optische Umsetzung des Lieder Editierens wurde in Anlehnung an das PowerPraise umgesetzt. Das visuelle Konzept ist beliebt bei den Anwendern von PowerPraise und man kann deshalb davon ausgehen, dass auch neue Anwender daran Freude haben.

**Aufteilung in 3 Spalten**  
In der ersten Spalte werden die Songparts/Liederteile mit ihren Slides dargestellt. Da Songparts z.b. der Refrain mehrfach vorkommen kann wird die Reheinfolge separat in der rechten Spalte verwaltet. In der Mitte wird der Text eines einzelnen Songparts eingegeben.

### Präsentation Bedienung / Eventverwaltung

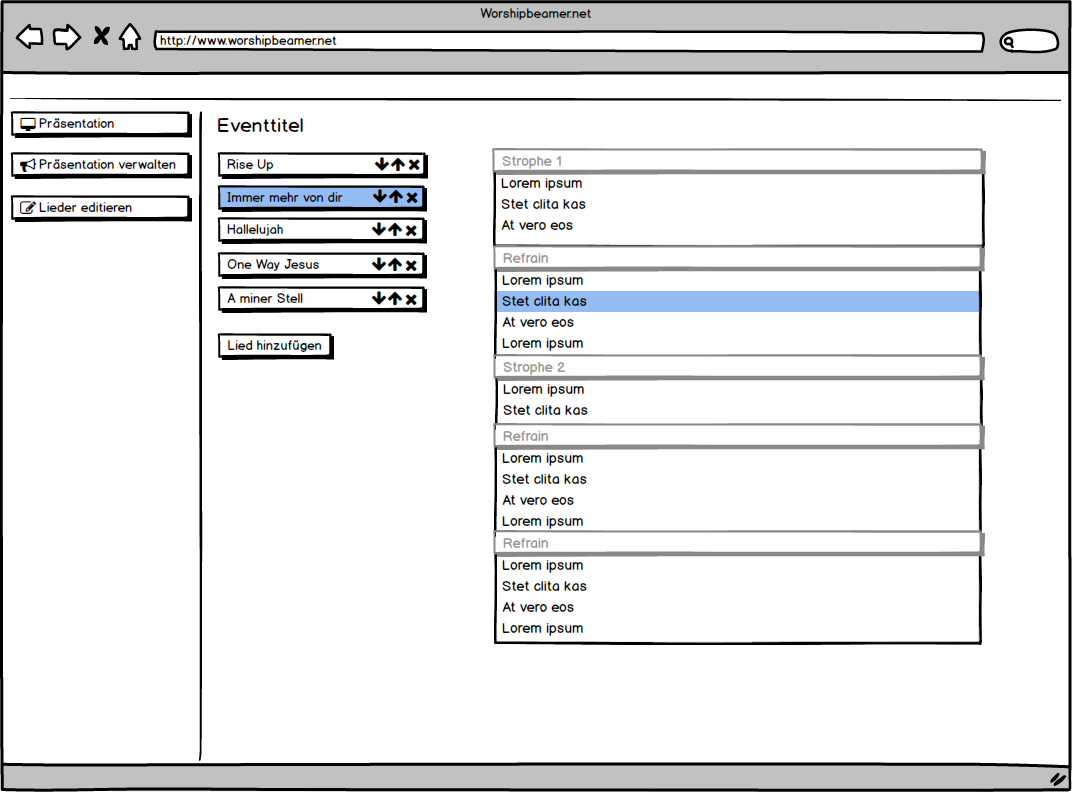


Abbildung 13 - Mokup der Präsentation Bedienung

Die Bedienung der Präsentation und Eventverwaltung sind in einer Ansicht verschmolzen und in Abbildung 10 skizziert. Dadurch kann während Präsentation flexibel auf Änderungen reagiert werden. Links wird die Lieder Reihefolge definiert oder während der Präsentaion abgearbeitet. Im grösseren rechten Bereich sind die Songpartslides nach dem im Liedereditor (Kapitel 374.7.1) definierten Ablauf aufgelistet und es sollte per Maus oder Pfeiltasten durchnavigiert werden können.

### Präsentationsansicht

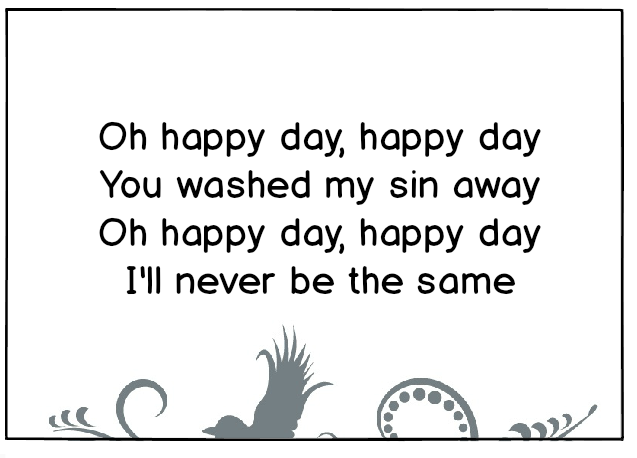


Abbildung 14 - Mockup Präsentationsansicht

Die Präsentationsansicht widmet sich dem Publikum. Der in der Bedinungsansicht aktuell ausgewählte Content wird in der Präsentationsansicht Fullscreen angezeigt.   
  
Identifikation/Authentifkation der Präsentationshow

Es sollen parallel verschiedenen Shows durchgeführt werden. Eine Präsentationsshow soll eindeutig identifiziert werden können.   
  
Ansatz 1: Der Webservice könnte der Präsentationsanischt alle verfügbaren Präsentationshows ausliefern. In der Präsentationsansicht könnte man die gewünschte Präsentation auswählen. Dieser Ansatz bedarf dem Wissen darüber wann eine Show gestartet und beendet wurde. Dieser Status zu verwalten bedarf eines manuellen Inputs eines Akteurs. Es könnte fälschlicherweise auch eine fremde Präsentationsshow angezeigt werden. Letzterer Nachteil könnte über eine Authentifikation in der Präsentationsansicht und anschliessendem Anzeigen aller offenen Präsentationsshows nur der eigenen Organisation umgangen werden.  
  
Ansatz 2: Die Präsentationsshow wird über einen Showkey eindeutig identifiziert. Der ShowKey bietet so einen minimalen Schutz. Um die Präsentationsshow sehen zu können muss der ShowKey bekannt sein. Der Showkey kann in der Präsentationsansicht auch erfasst werden, wenn die Show noch nicht eröffnet wurde. Sobald die Show eröffnet wird erhält dann die Präsentationsansicht die Daten. Der Webservice muss die Shows nicht verwalten. Bei der Eingabe des Showkeys kann man sich leicht vertippen oder der Showkey kann vergessen werden. Der Ansatz 2 hat keine Verwaltung der Shows daher kann nur über visuellen Input festgestellt werden ob der Showkey richtig war.

Entscheid: Die Möglichkeit im Vornherein eine Präsentationsansicht einzustellen wird vom Auftraggeber nach Absprache als wichtig erachtet. Der Ansatz 2 erfüllt ausserdem besser die Anforderungen FEQ mehr und Spart an Resourcen beim Webserver. Deshalb wird Ansatz 2 implementiert

## Ablauf Präsentation mit Showkey

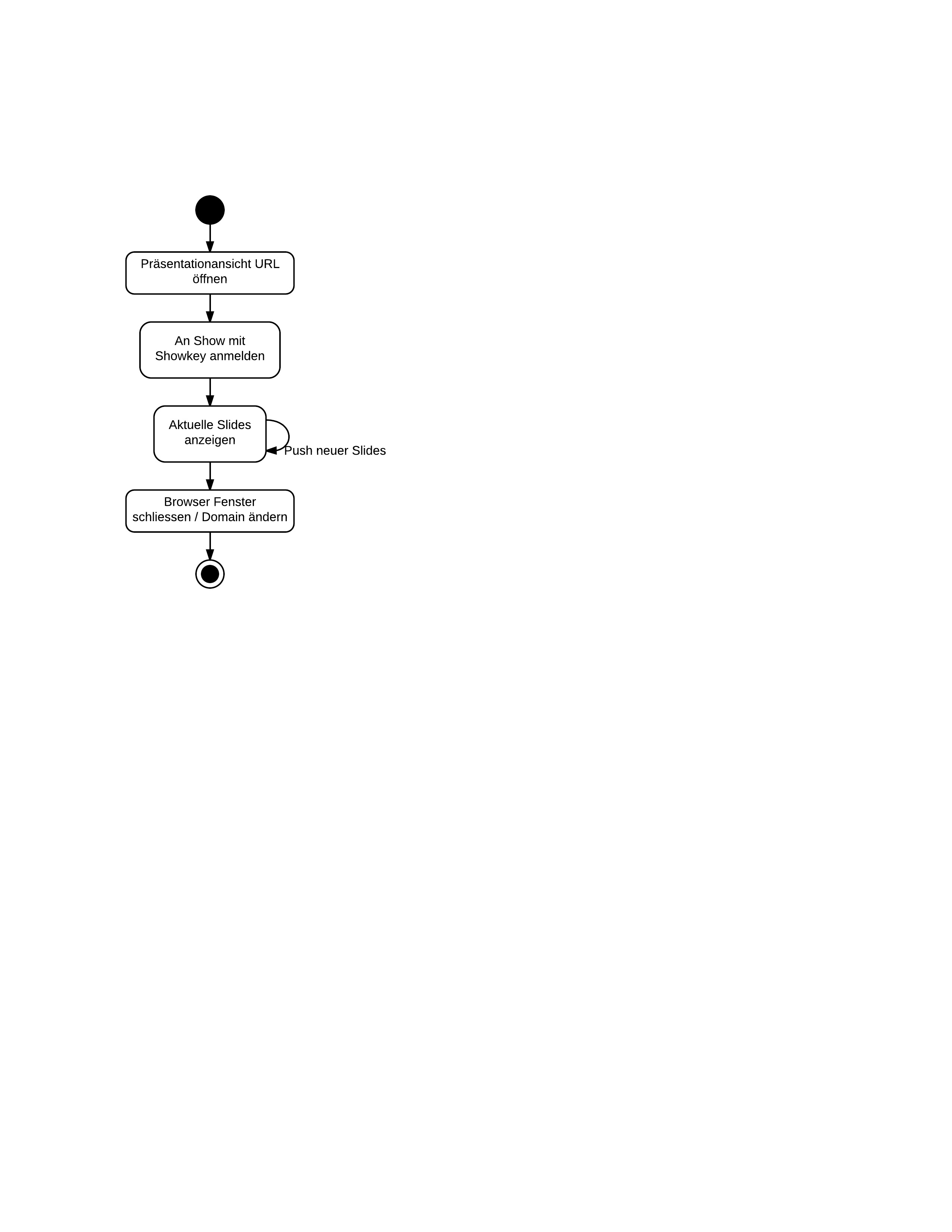


Abbildung 15 - Ablauf Präsentationsansicht

### Problematik erneuter Eingabe desselben Showkeys

FEQ fordert eine einfache Bedienung. Das starten der Präsentationsansicht auf einem Clientgerät ist einfach und die Eingabe des Code bedarf keiner weiteren Kenntnisse.  
Bei der Besprechung mit dem Arbeitsgeber wurde ein Spezialfall bemerkt. Wird ein Bildschirm mit MiniPC oder Android Stick unzugänglich montiert ist die jeweilige Eingabe des Showkeys mühsam. Da eine Kirche mit einer festinstallierte Hardware eine und deshalb am besten auch dieselbe Präsentationsshow hat wird sich der Showkey auch nicht ändern. Daher soll der Showkey im Cookie gespeichert werden. Beim Starten soll nach Ablauf von 5 Sekunden automatisch der letzte im Cookie gespeicherte Showkey abgerufen werden und der Präsentationsmodus mit der Show verbunden werden.

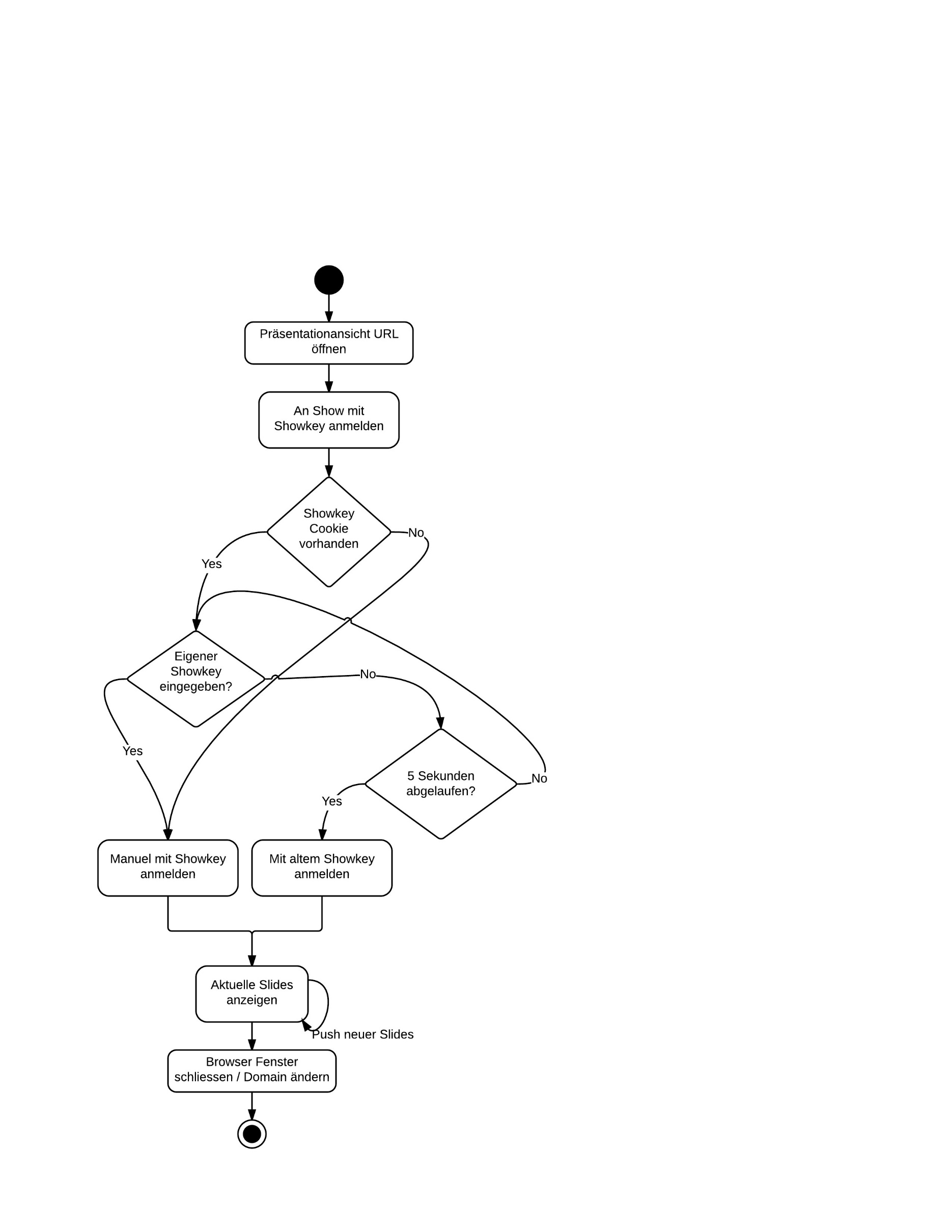


Abbildung 16 - Ablauf Präsentationsansicht mit Cookie

## Launcher App

Das Bedürfnis nach der Launcher App hat sich erst bei der Aufnahme der Anforderungen entwickelt. Dabei wurden das lösen der folgenden Herausforderungen als essenziell erachtet.

### Herausforderung Automatisches öffnen Webseite

Nach FEQ soll es möglich sein den Präsentationsmodus automatisiert zu starten. Trotzdem muss der Präsentationsmodus eindeutig mit einer Bedinung verknüpft werden können.

Es kann davon ausgegangen werden, dass in über 90% (Aussage von Auftraggeber Sven Wanzenried) dieselbe Installation des PC’s und Bildschirme oder Beamer verwendet wird. Generell kann man also davon ausgehen, dass innerhalb einer Location/Kirche/Gemeinde eine Bedinung immer dieselben Präsentationsmedium mit neuen Liedern bestückt. Generell kann man also seitens AngularJS WebApp (wie im Kapitel 4.8.2 beschrieben) immer wieder über den Showkey die Applikation öffnen. Jedoch muss eine Lösung erarbeitet werden um die Webseite mit der AngularJS App automatisch zuöffnen.

### Herausforderung Hardware und Betriebssystem für Bildschirme/Beamer

Nach FEQ sollen mehrere Bildschirme oder Beamer als Präsentationsmedium angesprochen werden können. Nach FEQ sollen zukünftig keine Spezialgrafikarten und keine Kabel für die Übertragung von Bildschirmsignalen verlegt werden. Dies bedarf unweigerlich für jedes Medium (z.b. Bildschirm oder Beamer) einer Hardware und eines Betriebsystem mit Browser welche den Browser Client / die Angular JS App darstellen kann.

### Lösungsansatz 1 – Mini PC

Ein Mini PC mit Windows oder ein Mac Betriebssystem (Mac Mini) besitzt die Möglichkeit beim Starten automatisiert Prozesse auszuführen. Ein Mini PC kennt ein Anwender aus seinem privaten oder geschäftlichen Umfeld und kann damit umgehen.

**Vorteil:**

* Bekanntes Betriebssystem
* Wird von Werbe-/Infoscreenanbieter eingesetzt (z.b. Mini PC Coop Pronto Shops, Mac Mini Berufsschule Winterthur)

**Nachteil**

* Stabilität mittelmässig
* Betriebsystem und Hardware teuer
* Ladezeit Betriebsystem
* Netzteil nötig, nicht nur über USB Strombezug

### Lösungsansatz 2 – Android Stick

Die Android Home Anwendung/ der Android Launcher kann mit einer eigenen Android-App übersteuert werden. Dadurch wird gar kein Betriebssystem gestartet. Die Stromversorgung des Androidsticks kann über ein 5V-USB Ausgang sichergestellt werden. Alle neueren Fernseher/Bildschirme oder Beamer verfügen über USB-Schnittstellen. Dadurch kann ein 2ter Stromanschluss für die betreibende Hardware ausgeschlossen werden.

**Vorteil:**

* Kein zusätzliches Netzteil nötig
* Günstige Hardware und kostenloses Betriebsystem
* Platzsparende Hardware
* Stabilität
* Betriebssystem nicht sichtbar

**Nachteil**

* Launcher App muss entwickelt werden
* Keine grösseren Werbe-/ Infoscreenanbieter

### Hardware/Betriebsystem Entscheidung

Die Vorteile des Android-Sticks überwiegen stark. Gerade Platz, Kosten und Stromversorgung sind entscheidend für eine einfache und anschauliche Installation. Über den Inititialaufwand für die Entwicklung der Launcher App kann schnell hinweggesehen werden, wenn die Kosteneinsparungen für die einzelnen Geräte gerechnet werden. Insbesondere dass die Launcher App innerhalb der Semesterarbeit entwickelt werden kann.

### App Launch Lifecycle

Die Worshipbeamer.net Android App soll den Standart Launcher von Android übersteuern. Nach Start der App wird die Internet Connection geprüft und der Browser mit der Worshibeamer.net AngularJS App geöffnet.

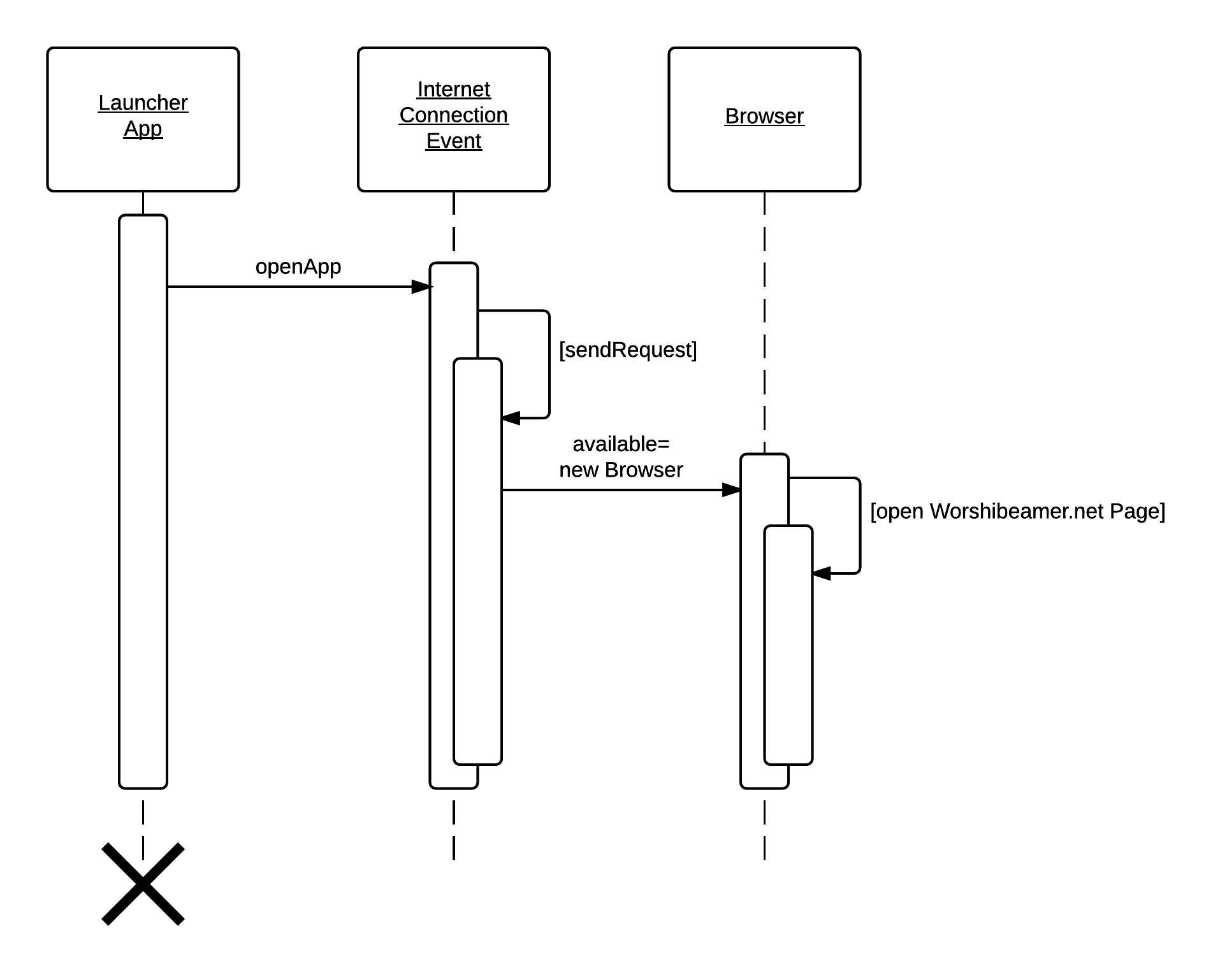


Abbildung 17 - Ablaufsdiagramm App Launcher

# Implementation

## Technologien

### C-Sharp

(Microsoft, 2015)  
Im Rahmen der Einführung von .net veröffentlichte Microsoft 2002 die Programmiersprache C-Sharp oder verkürzt *C#*. *C#* orientiert sich stark an *Java*, *C++*, *Haskell* und *Delphi*. Daher liegt es Nahe das *C#* eine objektorientierte Programmiersprache ist und der Wechsel von den zu vorgenannten Programmiersprachen auf *C#* einfach fällt.  
Neben Grundprinzipen der objektorientierten Programmierung resultiert aus folgende innovativen Sprachkonstrukte eine vereinfachte Programmierung:

* Gekapselte Methodensignaturen, Delegaten genannt, die typsichere Ereignisbenachrichtigungen ermöglichen
* Eigenschaften, die als Accessoren für private Membervariablen dienen
* Attribute, die zur Laufzeit deklarative Metadaten zu Typen bereitstellen
* Inline-XML-Dokumentationskommentare
* Sprachintegrierte Abfrage (Language-Integrated Query, LINQ), die integrierte Abfragefunktionen für eine Vielzahl von Datenquellen bereitstellt

Der C#-Erstellungsprozess ist im Vergleich zu *C* und *C++* einfach und flexibler als in *Java*. Es gibt keine separaten Headerdateien und es ist nicht erforderlich, Methoden und Typen in einer bestimmten Reihenfolge zu deklarieren. Eine *C#-*Quelldatei kann eine beliebige Anzahl von Klassen, Strukturen, Schnittstellen und Ereignissen definieren.

### ASP.net Web API 2 / ASP.net MVC Framework

(Microsoft, 2015)  
Microsoft entwickelte mit dem *ASP.net MVC Framework* ein schlankes und einfach zu testendes Präsentationsframework. Wie im Namen enthalten basiert das Framework auf dem MVC-Pattern.  
Die klare Trennung von Eingabelogik, Geschftslogik und Präsentationslogik wird durch die vom Framework bereitgestellten Komponenten unterstützt.  
Um *RESTful-Webservices* einfach entwickeln zu können stellt Microsoft mit ASP.net Web API 2 eine einfache zu verwendendes und starkes Software Paket zur Verfügung. ASP.net Web API 2 basiert auf dem ASP.net MVC Framework.

### **Entity Framework**

(Microsoft MSDN, 2015)  
Entity Framework (EF) ist eine objektrelationale Zuordnung, die .NET-Entwicklern über domänenspezifische Objekte die Nutzung relationaler Daten ermöglicht. Ein Grossteil des Datenzugriffscodes, den Entwickler normalerweise programmieren, muss folglich nicht geschrieben werden.

### AngularJS

(Panda, 2014)  
Mittels AngularJS wird die Client-Browser App entwickelt. AngularJS ist ein Javascript Framework, welches OpenSource von Google Inc. veröffentlicht wurde. AngularJS macht einen Grossteil des Codes, den man normalerweise schreibt, überflüssig. Die Reduktion des Codes begründet sich durch die Automatisierung von Standardaufgaben. Die manuelle DOM-Selektion, DOM-Manipulation und Event-Behandlung werden durch AngularJS überflüssig. Durch Einsatz von Direktiven und Modulen wird die Wiederverwendbarkeit von Code ermöglicht.  
Die normalen Datentypen von JavaScript können verwendet werden. Dadurch ist es sehr einfach möglich, fremde Bibliotheken einzubinden, ohne eine weitere Zwischenschicht (Glue Code) zu implementieren. Die Methode, die AngularJS dazu verwendet nennt sich Dirty-Checking und wird im Vertiefungskapitel näher erklärt.

### JSON

Zwischen der AngularJS WebApp und dem Webservice dient JSON(*JavaScript Object Notation*) als Datenübertragungsformat. JSON zeichnet sich durch seine schlanke Notation und der objektnahen Darstellung aus.

### Adobe Air / AS3

(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.  
  
(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.

## Authentifizierung



Abbildung 18 Hauptsoftware Komponenten für Authentification

(Babal, 2014)

„Autehtication is one oft he most importent parts of any web application“ (Babal, 2014) Dieser Wichtige Part betrifft auch den Web Application Teil von Worshipbeamer.net.

### Traditionelle Authentifizierung

Bei der traditionellen Authentifizierung, ➊ sendet der Client die Login Daten an den Server. ➋ Sind die Daten valid, öffnet der Server eine Session und sendet die Session Informationen im Header an den Client. Der Client speichert die Session-ID im Cookie. ➌ Bei einer überprüft der Server ob die Session Informationen valid sind und genügend Rechte für die angeforderte Abfrage zur Verfügung stehen. Ist dies der Fall wird die Abfrage ausgeführt und ➍ wenn vorhanden, Content an den Client zurückgesendet.

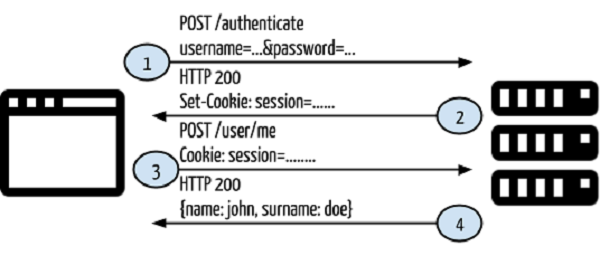


Abbildung 19 - Ablauf traditionelle Authentifizierung Quelle (Babal, 2014)

### Token-basierte Authentifizierung

Wie bei der traditionellen Authentifizierung, ➊ sendet der Client auch bei der token-basierten Authentifizierung die Login Daten an den Server. ➋ Sind die Daten valid, erstellt der Server ein Token und sendet dieses dem Client. Der Client speichert den Token. ➌ Um nun eine Abfrage auszuführend sendet der Client das Token im Header an den Server mit. Der Server überprüft ob das Token valid ist und genügend Rechte für die angeforderte Abfrage zur Verfügung stehen. Ist dies der Fall wird die Abfrage ausgeführt und ➍ wenn vorhanden, Content an den Client zurückgesendet.

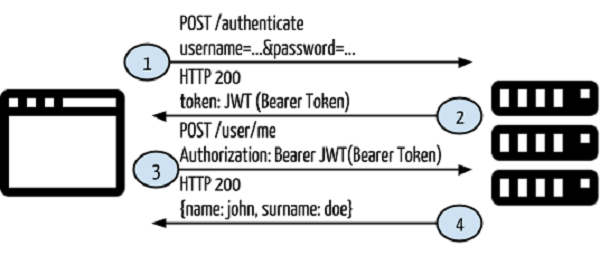


Abbildung 20 - Ablauf Token-basierte Authentifizierung Quelle (Babal, 2014)

### Vorteile Token-basierter Authentifizierung

Statuslos Die Authentificationsdaten werden nicht in Sessions oder Cookies gespeichert sondern bei jedem REQUEST an den Server mitgesendet.

Client Type unabhängig Das Token kann von jedem HTTP-Request fähigen Client gesendet werden.

Trennung Klare Trennung zwischen Client und Server.

Keine Cookie-Sessions Bei Cookies-Session kann die Request-Source nicht auf Vertrauenswürdigkeit überprüft werden. Das Token wird bei jeder Abfrage im Header überprüfbar mitgesendet.

## Near Real Time Kommunikation

Die Funktionale und Nichfunktionalen Anforderungen verlangen eine Kommunikation zwischen Bedienung und Präsentation, die möglichst Real Time agiert.   
Im Bereich Kommunikation in der Informatik ist seit je eher die Schnelligkeit des Informationsaustauschs wichtig. Dieser Umstand wird auf allen Ebenen des OSI-Layer erforscht und verbessert. Worshipbeamer.net überträgt bei einer Präsentation nur eine kleine Datenmenge. Deshalb ist nicht die Übertragungsgeschwindigkeit interessant sondern vielmehr die Reaktionszeit bei einer Veränderung. Unser Augenmerk liegt also Hauptsächlich auf OSI-Layer 7. Die Wahl der Technologien in OSI-Layer 7 kann Veränderungen in den anderen OSI-Layern bewirken.

### Polling

Bei Polling fragt der Client zyklisch beim Server nach um eine Wertänderung feststellen zu können. Die Abfrage kann mittels einfacher HTTP-Verbindung umgesetzt werden und erfordert keine zusätzlichen technischen Anforderungen. Polling ist mit allen bekannten Browsern einsetzbar. Je mehr Zyklen pro Zeiteinheit umso geringer ist die Zeit bis der Client auf eine Wertänderung reagieren kann. Mehr Zyklen pro Zeiteinheit bedeutet auch mehr Rechenzeit bei Client und Server sowie mehr Traffic auf dem Netzwerk, egal ob eine Wertänderung stattfindet oder nicht. Dieselbe Steigerung findet bei der Zunahme der Clients statt. Polling skaliert also nicht.

### Implementation Websockets

(Wikipedia - Websockets, 2015)

Im Gegensatz zu einer einfachen HTTP-Verbindung kann der Server bei Websockets Informationen an den Client ausliefern ohne vorhergehende Anfrage des Clients. Der Client braucht zu Beginn nur die Verbindung zum Server zu öffnen. So kann der Server dann Daten nur dann Daten ausliefern wenn auch neue Informationen vorhanden sind. Dies spart einiges an Netzwerk Overhead. Zusätzlich wird kein http-Header benötigt, welches einige 100 Bytes einspart.  
Websockets sind jedoch nicht von allen aktiven Browser verfügbar.

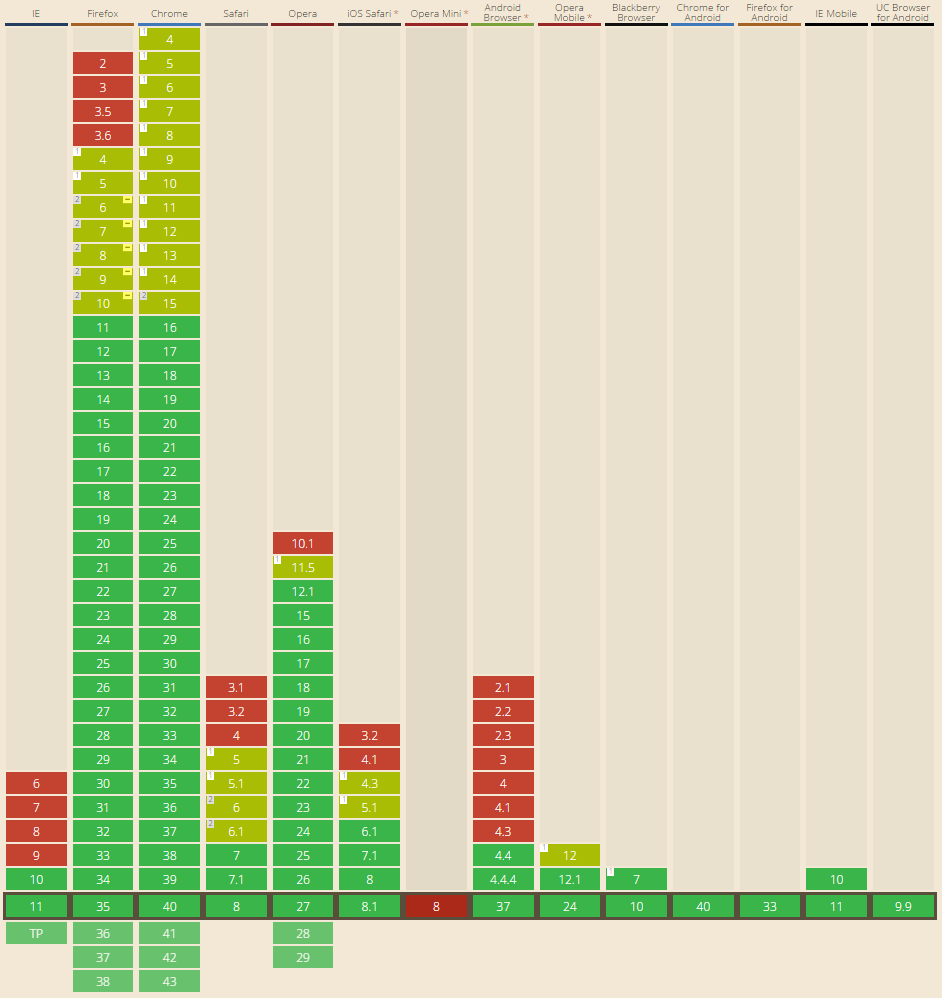


Abbildung 21 - Browser Kompatibilität Websockets, rot=keine Kompatibilität, hellgrün eingeschränkte Kompatibilität, grün volle Kompatibilität Quelle http://caniuse.com/#feat=websockets

### SignalR

Beide zu vorgenannten Konzepte haben, wie in den vorhergehenden Kapiteln beschrieben, viele Vorteile und leider auch zu viele Nachteile. Einzel können die Anforderungen nur unbefriedigend erfüllt werden. Deshalb sollte je nach Umgebung eine andere Technologie dynamisch ausgewählt und verwendet werden. Mit dieser Problematik wurden verschiedene Personen bereits konfrontiert. Die Nachfrage ist so gross, dass ein mittlerweile 5 köpfige Team von SignalR probiert mit ihrem gleichnamigen Framework SignalR genau diese Idee umzusetzen.   
SignalR stelle eine einfache API zur Verfügung. Über diese können RPC (Remote Procedure calls) aufgerufen werden, welche JavaScript functions in Web Browsern oder Functionen von anderen Clientanwendungen aufrufen. SignalR wählt automatisch das schnellste verfügbare Transfer-Konzept aus über welchem die Calls gemacht werden können. Es definiert zur Laufzeit ob WebSockets, Server-Sent Events, Forever Frame, or just Long Polling verwendet werden.



Abbildung 22 Visualisierung SignalR API Quelle asp.net/signalr

## Datenbank Design

Der *Code First* Ansatz des *Entity Frameworks* wurde verwendet um die Datenbank von Worshipbeamer.net zu erstellen. *Code First* bedeutet, dass zuerst die *Model* Klassen mit ihren *Properites* programmiert werden.   
Das *Entity-Framework* generiert eigenständig Tabellen und deren Spalten und Werte in Form eines Migrations Files. Mittels Migration wird bei Änderungen der Model-Klassen/Datenmodelle die Datenbank aktualisiert. Die aus den Änderungen automatisierte Migration enthält sowohl up wie au downgrade Befehle für die Datenbank. So können Änderungen einfach ein gespielt werden oder wieder entfernt werden. Der Publish-Prozess der Azure Cloud kann die Migation-Files ausführen. So kann die Test- und Liveumgebung einfach auf den neuen Datenbankstand gebracht werden und bei Auftreten eines Fehlers wieder zurückgefahren werden.

### Convention over configuration

Der *Code First* Ansatz des *Entity Frameworks* kann verschieden umgesetzt werden. Bietet das Framework doch eine Vielzahl von Möglichkeiten an. Überall wo es das eingesetzt Framework erlaubt, soll nach Kapitel 4.3, das Entwickler-Paradigma *Convention over configuration* verwendet werden. Deshalb wurden die Datenmodelle nach den Konventionen des Entity Frameworks erstellt.  
  
Die Relationen zwischen Entitäten ergeben sich durch die virtual gekennzeichneten Properties.

### Datenbankdiagramm

Das Datenbankdiagramm wird automatisch vom *Entity Framework* generiert. Deshalb ist es nur als visuelle Unterstützung hier abgebildet.

## Webservice

Die Webservice, welche vom Server zur Verfügung gestellt wird, ist basiernd auf *ASP.net Web API 2* implementiert worden. ASP.net Web API unterstützt den Programmierer beim Erstellen eines validen RESTful-Webservices.

### API-Services

In die nächsten Unterkapitel werden die verschiedenen Services unseres Webservices erläutert.

### Screen Konfiguration

Endpunkt  
Antwort

|  |  |  |
| --- | --- | --- |
| Feldname | Typ | Beschreibung |
| ZHAW, FHS | Hoch | Die ZHAW legt sowohl den organisatorischen Rahmen, als auch formale Vorgaben fest. |
| Chrischona Aadorf Technik Mitarbeiter | Sehr hoch | Die technischen Mitarbeiter der Chrischona sind Verantwortlich dass an den Events die Liedertexte zur richtigen Zeit dem Publikum angezeigt werden.. |
| Chrischona Aadorf Bandmitglieder | Hoch | Die Bandmitglieder erfassen die Liedertexte. |
| Chrischona Aadorf Organisator, Pastor | Niedrig | Der Event/Gottesdienst Organisator stellt in Absprache mit dem Pastor den Ablauf zusammen. |

Beispiel

Json code

# 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.

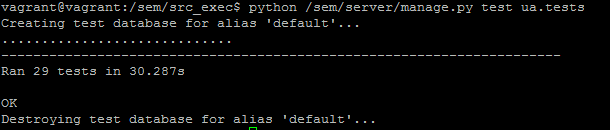


Abbildung 23 - Unittests - Backend

## 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.

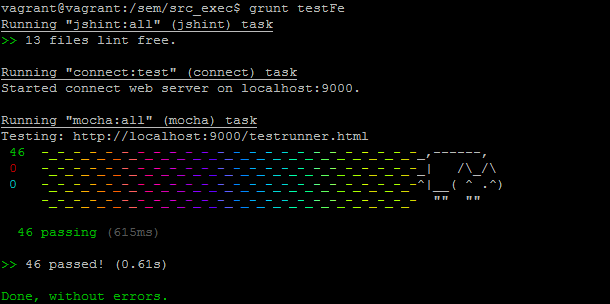


Abbildung 24 - Unittets - Frontend

## 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.

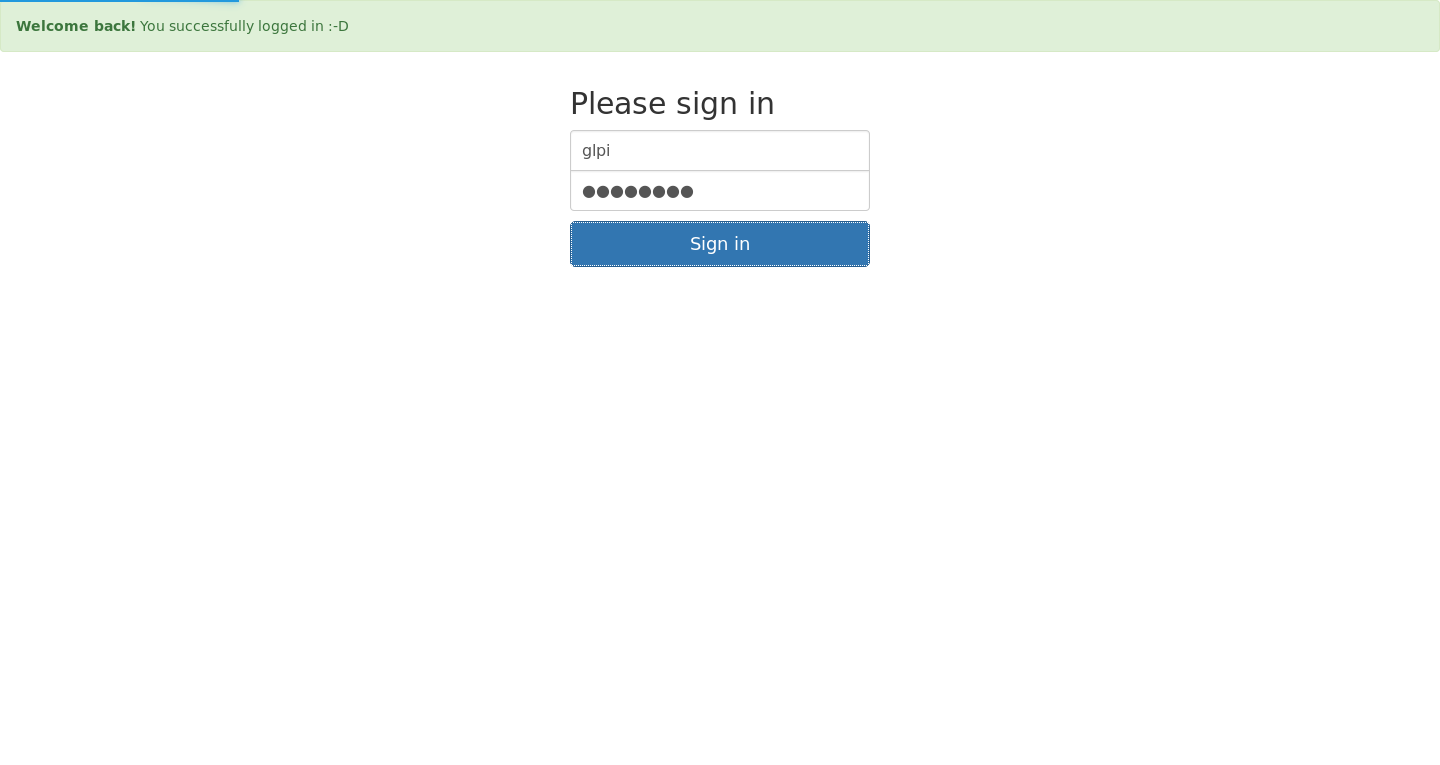


Abbildung 25 - Screenshot Integration-Test

## 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)