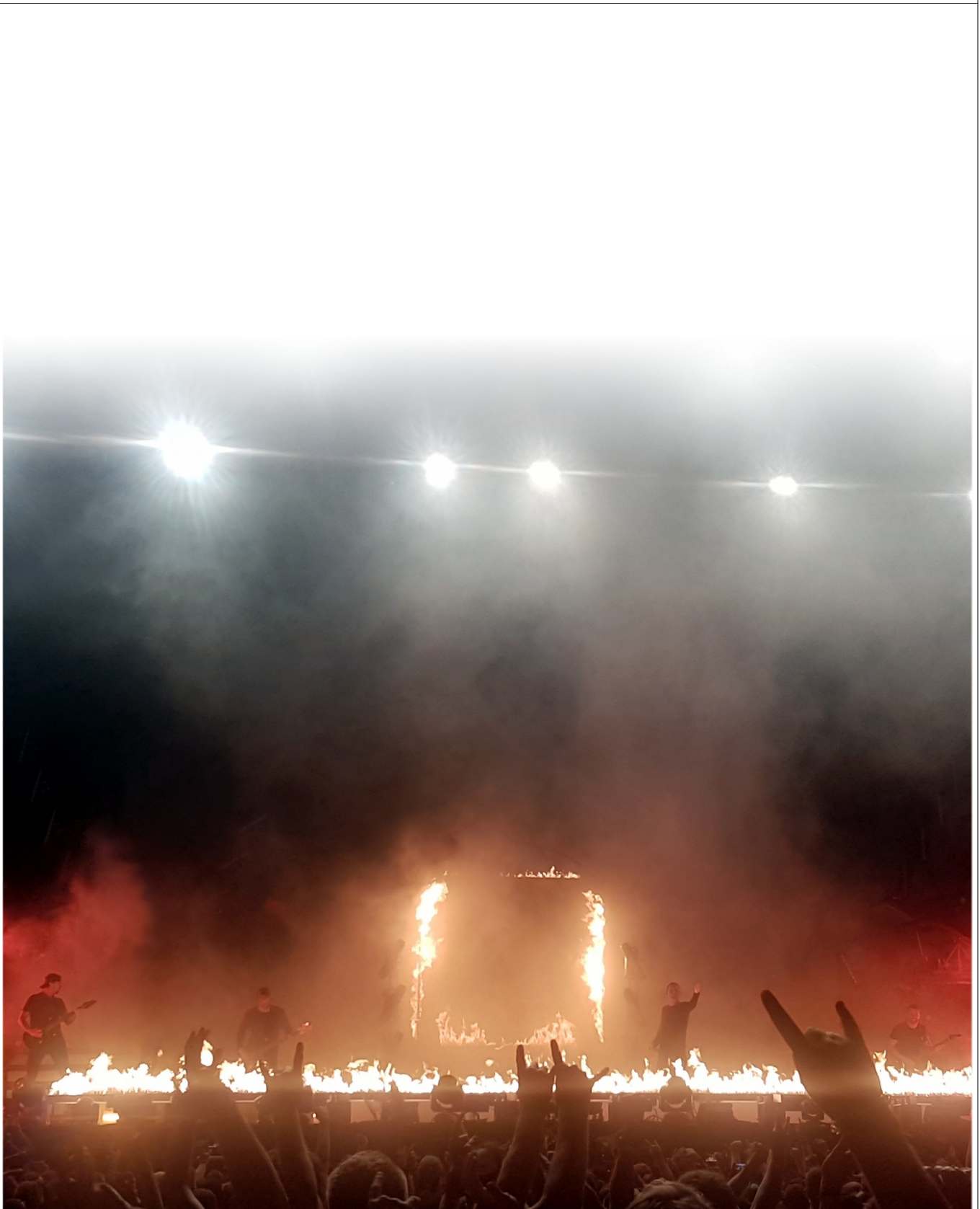


# Konzertkalender

Damian Senn





TSBE

# *Management Summary*

Dipl. Techniker Informatik

**Konzertkalender**

von Damian Senn

Wer kennt es nicht, es ist Wochenende und es sind noch keine konkrete Pläne gemacht. Diese Situation fordert Datenquellen mit welchen man sich einen Überblick von möglichen Optionen erstellen kann. Interessiert man sich für Musik und Live-Konzerte, und ist neu in einer Stadt, oder generell nicht über die lokalen Möglichkeiten genug gut Informiert, wird es schnell schwierig herauszufinden, was für Konzerte sich in der Nähe anbieten. Oder man plant eine Woche oder ein Wochenende in einer fremden Stadt und möchte herausfinden, ob es Konzerte die dem eigenen Musikgeschmack entsprechen stattfinden.

Die Konzertkalender Webapplikation «Gigpillar», umgesetzt im Rahmen einer praktischen Diplomarbeit an der «Telematik Schule Bern», bietet sich als Option an, um ein Konzert in der Nähe des Benutzers zu suchen.

Besuchern wird ermöglicht, in der Applikation nach nach Konzerten in Städten zu suchen und diese in einem Datumsbereich einzugrenzen. Ausserdem bietet die Suche einen Filter an, um Konzerte nach Musik-Genre zu filtern, um das Suchresultat dem Geschmack des Besuchers anzupassen.

Zu Beginn des Projektes wurden Ziele sowie Anforderungen definiert. Anhand dieser wurde eine Nutzwertanalyse sowie eine Risikoanalyse erstellt. In der Nutzwertanalyse wurden Technologie Entscheide evaluiert und getroffen. Die Analyse hat gezeigt, dass moderne Technologien wie zum Beispiel die JavaScript Library «React» zuviel Komplexität mit sich bringen, um im Rahmen einer Diplomarbeit eingesetzt zu werden. Das Ergebnis der Nutzwertanalysen ist, dass mit dem Phoenix Framework auf einen klassischen Server-Side Rendered Ansatz gesetzt wird.

In der Konzeptphase wurden die Ideen für die Applikation konkretisiert. So wurden Mockups, Datenflussdiagramme sowie ein Datenbankschema definiert. Zusätzlich wurde ein Testkonzept und das dazugehörige Testprotokoll anhand der definierten Ziele und Anforderungen erstellt.

Während der Realisierungsphase wurde der grösste Teil an Zielen sowie Anforderungen umgesetzt, kleinere unerwartete Probleme haben zu Verzögerungen von weniger wichtigen Funktionalitäten geführt, welche zu einem späteren Zeitpunkt nachgeliefert werden können.



# Authentizität

Mit meiner Unterschrift bestätige ich, die vorliegende Diplomarbeit selbstständig, ohne Hilfe Dritter und nur unter Benutzung der angegebenen Quellen ohne Copyright-Verletzung, erstellt zu haben.

Unterschrift:

---

Ort:

---

Datum:

---



## *Danksagung*

An dieser Stelle möchte ich mich herzlichst bei meinen beiden Experten, Sandro Bertolino und Severin Rätz, für die Begleitung meiner Projektarbeit bedanken.





# Inhaltsverzeichnis

<b>Management Summary</b>	<b>i</b>
<b>Authentizität</b>	<b>ii</b>
<b>Danksagung</b>	<b>iii</b>
<b>Abbildungsverzeichnis</b>	<b>xi</b>
<b>Tabellenverzeichnis</b>	<b>xiii</b>
<b>1 Initialisierung</b>	<b>1</b>
1.1 Ausgangslage . . . . .	1
1.2 Projektziele . . . . .	2
1.3 Projektorganisation . . . . .	3
1.3.1 Tätigkeiten im Projekt . . . . .	3
1.3.2 Kommunikation . . . . .	3
1.4 Ausgefüllter Projektplan . . . . .	4
1.5 Lieferergebnisse . . . . .	5
1.6 Terminplan . . . . .	5
1.7 Meilensteine . . . . .	5
1.8 Risiken . . . . .	6
1.8.1 Projektrisiken . . . . .	6
1.8.2 Massnahmen . . . . .	7
1.8.3 Risikodiagramm . . . . .	7
1.9 Abgrenzungen . . . . .	8
1.10 Studie . . . . .	9
1.10.1 Informationsbeschaffung . . . . .	9
1.10.2 Anforderungskatalog . . . . .	10
1.11 Evaluation Browser-Technologie . . . . .	12
1.11.1 Variante: React . . . . .	12
1.11.2 Variante: Next.js . . . . .	12
1.11.3 Variante: SSR . . . . .	12
1.12 Entscheid Browser-Technologie . . . . .	12
1.13 Evaluation Server-Technologie . . . . .	13
1.13.1 Variante: Node.js / koa.js . . . . .	13
1.13.2 Variante: Elixir / Phoenix . . . . .	13
1.13.3 Variante: Next.js . . . . .	13
1.14 Entscheid Server-Technologie . . . . .	13
1.15 Evaluation Testing-Technologie . . . . .	14
1.15.1 Jest + Puppeteer . . . . .	14
1.15.2 Wallaby . . . . .	14
1.16 Entscheid Testing-Technologie . . . . .	14
1.17 Wirtschaftlichkeit . . . . .	15

1.17.1	Projektkosten	15
1.17.2	Break Even Analyse	16
<b>2</b>	<b>Konzept</b>	<b>17</b>
2.1	Portalname	17
2.2	Design	18
2.3	Software	23
2.3.1	Datenfluss	23
2.3.2	Datenbankstruktur	24
2.4	Testkonzept	25
2.4.1	Unit-Tests	25
2.4.2	Integration-Tests	25
2.4.3	Browser-Tests	25
2.4.4	Visual-Tests	25
2.4.5	Akzeptanztests	25
2.5	Fazit	26
2.5.1	Abweichungen	26
2.5.2	Probleme	26
2.5.3	Machbarkeit	26
2.5.4	Wirtschaftlichkeit	26
2.5.5	Erweiterbarkeit	26
2.5.6	Projektplan	26
<b>3</b>	<b>Realisierung</b>	<b>27</b>
3.1	Umsetzung	27
3.1.1	HTML-Prototyp	27
3.1.2	Datenbankschema	28
3.2	Tests	28
<b>4</b>	<b>Einführung</b>	<b>29</b>
4.1	Projektcontrolling	29
4.1.1	Erfüllung der Ziele	29
4.1.2	Erfüllung der Anforderungen	31
4.2	Wirtschaftlichkeit	33
<b>5</b>	<b>Schlussbetrachtung</b>	<b>35</b>
5.1	Planung	35
5.2	Ziele und Anforderungen	35
5.3	Realisierung	35
<b>A</b>	<b>Projektinitialisierungsauftrag</b>	<b>37</b>
<b>B</b>	<b>Sitzungsprotokoll — Kickoff</b>	<b>41</b>
<b>C</b>	<b>Terminplan</b>	<b>43</b>
<b>D</b>	<b>Studie</b>	<b>45</b>
D.1	Zweck des Dokuments	45
D.2	Informationsbeschaffung	45
D.3	Anforderungskatalog	46
D.4	Evaluation Browser-Technologie	48
D.4.1	Variante: React	49

D.4.2 Variante: Next.js . . . . .	49
D.4.3 Variante: SSR . . . . .	49
D.5 Bewertungen Browser-Technologie . . . . .	50
D.6 Entscheid Browser-Technologie . . . . .	50
D.7 Evaluation Server-Technologie . . . . .	51
D.7.1 Variante: Node.js / koa.js . . . . .	51
D.7.2 Variante: Elixir / Phoenix . . . . .	51
D.7.3 Variante: Next.js . . . . .	52
D.8 Bewertungen Server-Technologie . . . . .	53
D.9 Entscheid Server-Technologie . . . . .	53
D.10 Evaluation Testing-Technologie . . . . .	54
D.10.1 Jest + Puppeteer . . . . .	54
D.10.2 Wallaby . . . . .	54
D.11 Bewertungen Testing-Technologie . . . . .	55
D.12 Entscheid Testing-Technologie . . . . .	55
D.13 Wirtschaftlichkeit . . . . .	56
D.13.1 Projektkosten . . . . .	56
D.13.2 Break Even Analyse . . . . .	57
<b>E Projektauftrag . . . . .</b>	<b>59</b>
E.1 Zweck des Dokuments . . . . .	59
E.2 Ausgangslage . . . . .	59
E.3 Projektziele . . . . .	60
E.4 Rahmenbedingungen . . . . .	60
E.5 Terminplan . . . . .	61
E.6 Meilensteine . . . . .	61
E.7 Organigramm . . . . .	62
E.7.1 Tätigkeiten im Projekt . . . . .	62
E.7.2 Kommunikation . . . . .	62
E.8 Abgrenzungen . . . . .	63
E.9 Anforderungskatalog . . . . .	64
E.10 Lösungsbeschreibung . . . . .	66
E.11 Kosten . . . . .	67
E.12 Risiken . . . . .	68
E.12.1 Projektrisiken . . . . .	69
E.12.2 Massnahmen . . . . .	70
E.12.3 Risikodiagramm ohne Massnahmen . . . . .	71
E.12.4 Risikodiagramm mit Massnahmen . . . . .	72
<b>F Wirtschaftlichkeit - Gigboost . . . . .</b>	<b>73</b>
<b>G Wirtschaftlichkeit - Werbung . . . . .</b>	<b>75</b>
<b>H Konzept . . . . .</b>	<b>77</b>
H.1 Zweck des Dokuments . . . . .	77
H.1.1 Teilkonzepte . . . . .	77
H.2 Portalname . . . . .	78
H.3 Design- und Bedienkonzept . . . . .	79
H.3.1 Mockups . . . . .	79
H.3.2 Genre Filter . . . . .	84
H.4 Softwarekonzept . . . . .	85

H.4.1	Datenfluss	85
H.4.2	Datenbankstruktur	89
H.5	Testkonzept	90
H.5.1	Unit-Tests	90
H.5.2	Integration-Tests	90
H.5.3	Browser-Tests	90
H.5.4	Visual-Tests	90
H.5.5	Akzeptanztests	91
H.6	Fazit	103
H.6.1	Abweichungen	103
H.6.2	Probleme	103
H.6.3	Machbarkeit	103
H.6.4	Wirtschaftlichkeit	103
H.6.5	Erweiterbarkeit	103
H.6.6	Projektplan	103
<b>I</b>	<b>Sitzungsprotokoll — Zwischenmeeting</b>	<b>105</b>
<b>J</b>	<b>Realisierung</b>	<b>107</b>
J.1	HTML-Prototyp	107
J.1.1	Screenshots	108
J.2	Projekt Setup	113
J.3	Dependency Management	114
J.4	Datenbankschema	118
J.4.1	Finales Schema	119
J.5	Berechtigungssystem	120
J.6	Location Autocomplete	121
J.7	HTML Erweiterungen	122
J.8	Asset Optimierungen	125
J.9	File Upload	126
J.10	OpenStreetMap	126
J.11	SEO - Microdata	127
J.12	Suche	129
J.13	Probleme	130
J.13.1	Dependency Konflikt	130
J.13.2	Formularbehandlung in Phoenix	131
J.13.3	Datumsbehandlung	131
J.13.4	Behandlung von Datenrelationen	131
J.14	Tests	132
J.15	Testprotokoll	133
J.16	Offene Punkte	145
J.16.1	Genre Zuordnung für Gigs	145
J.16.2	Benutzer Profil	145
J.16.3	Passwort-Reset Funktionalität	145
J.16.4	Screenshot Tests	145
J.16.5	Expliziter Ort Kontext	145
J.16.6	Mögliche Erweiterungen	146
J.17	Installationsanleitung	146
J.17.1	Benötigte Software	146
J.17.2	Benötigte Dienste	147
J.17.3	Umgebungsvariablen	148

J.17.4	Initialen Benutzer erstellen	149
J.17.5	Grundsetup und Start	150
<b>K</b>	<b>Einführung</b>	<b>151</b>
K.1	Projektcontrolling	151
K.1.1	Erfüllung der Ziele	151
K.1.2	Erfüllung der Anforderungen	153
<b>L</b>	<b>Arbeitsjournal</b>	<b>155</b>
L.1	Sonntag 3. März	155
L.2	Dienstag 5. März	155
L.3	Mittwoch 6. März	155
L.4	Samstag 9. März	155
L.5	Dienstag 12. März	155
L.6	Samstag 16. März	156
L.7	Dienstag 19. März	156
L.8	Mittwoch 27. März	156
L.9	Sonntag 31. März	156
L.10	Sonntag 31. März	156
L.11	Freitag 5. April	156
L.12	Samstag 6. April	156
L.13	Mittwoch 10. April	157
L.14	Freitag 12. April	157
L.15	Samstag 13. April	157
L.16	Sonntag 14. April	157
L.17	Montag 15. April	157
L.18	Mittwoch 17. April	157
L.19	Freitag 19. April	157
L.20	Sonntag 21. April	158
L.21	Montag 22. April	158
L.22	Dienstag 23. April	158
L.23	Mittwoch 24. April	158
L.24	Mittwoch 1. Mai	158
L.25	Sonntag 5. Mai	158
L.26	Montag 6. Mai	158
L.27	Dienstag 7. Mai	159
L.28	Mittwoch 8. Mai	159
L.29	Donnerstag 9. Mai	159
L.30	Freitag 10. Mai	159
L.31	Samstag 11. Mai	159
L.32	Sonntag 12. Mai	159
L.33	Montag 13. Mai	160
L.34	Dienstag 14. Mai	160
L.35	Mittwoch 15. Mai	160
<b>M</b>	<b>Biweekly Reports</b>	<b>161</b>
M.1	Kalenderwoche 11-12	161
M.2	Kalenderwoche 13-14	162
M.3	Kalenderwoche 15-16	163
M.4	Kalenderwoche 17-19	164

<b>Abkürzungsverzeichnis</b>	<b>165</b>
<b>Glossar</b>	<b>167</b>

# Abbildungsverzeichnis

1.1	Organigramm	3
1.2	Abgrenzungen	8
1.3	Phoenix Framework Logo	13
1.4	Wallaby Logo	14
1.5	Break-Even Analyse - Gigboost	16
1.6	Break-Even Analyse - Werbung	16
2.1	Mockup: Homepage	18
2.2	Mockup: Suchresultate	19
2.3	Mockup: Gig Ansicht	20
2.4	Mockup: Gig erfassen	21
2.5	Mockup: Benutzerprofil	22
2.6	Datenfluss: Homepage	23
2.7	Konzept: Entity Relationship Diagram	24
3.1	HTML Prototyp	27
D.1	Break-Even Analyse - Gigboost	57
D.2	Break-Even Analyse - Werbung	58
E.1	Organigramm	62
E.2	Abgrenzungen	63
E.3	Phoenix Framework Logo	66
E.4	Wallaby Logo	66
H.1	Mockup: Homepage	79
H.2	Mockup: Suchresultate	80
H.3	Mockup: Gig Ansicht	81
H.4	Mockup: Gig erfassen	82
H.5	Mockup: Benutzerprofil	83
H.6	Datenfluss: Homepage	85
H.7	Datenfluss: Suchfeld	86
H.8	Datenfluss: Gig erstellen - Locationfeld	87
H.9	Datenfluss: Passwort-Reset	88
H.10	Konzept: Entity Relationship Diagram	89
J.1	HTML Prototyp: Homepage	108
J.2	HTML Prototyp: Suche	109
J.3	HTML Prototyp: Gig erfassen	110
J.4	HTML Prototyp: Login	111
J.5	HTML Prototyp: Registrierung	112
J.6	Dependabot: Installation	114
J.7	Dependabot: Konfiguration für Elixir	115
J.8	Dependabot: Konfiguration für JavaScript	116

J.9	Dependabot: Pull-Requests für Updates . . . . .	117
J.10	Realisierung: Entity Relationship Diagram . . . . .	119
J.11	acrshortjsonld: Validierung . . . . .	128
J.12	Tests: Ausführung der automatisierten Tests . . . . .	132
J.13	Initialen Benutzer erstellen . . . . .	149



# Tabellenverzeichnis

1.1	Ziele	2
1.2	Tätigkeiten Verteilung	3
1.3	Terminplan	5
1.4	Meilensteine	5
1.5	Projektrisiken	6
1.6	Projektrisiken - Massnahmen	7
1.7	Informationsbeschaffung	9
1.8	Anforderungskatalog	11
1.9	Projektkosten	15
1.10	Betriebskosten	15
4.1	Controlling: Ziele	30
4.2	Anforderungskatalog	32
D.1	Informationsbeschaffung	45
D.2	Anforderungskatalog	47
D.3	Browser-Technologie Kriterien	48
D.4	Browser-Technologie Bewertung	50
D.5	Server-Technologie Kriterien	51
D.6	Server-Technologie Bewertung	53
D.7	Testing-Technologie Kriterien	54
D.8	Testing-Technologie Bewertung	55
D.9	Projektkosten	56
D.10	Betriebskosten	56
D.11	Werbeeinnahmen pro Besucher	58
E.1	Ziele	60
E.2	Terminplan	61
E.3	Meilensteine	61
E.4	Tätigkeiten Verteilung	62
E.5	Anforderungskatalog	65
E.6	Projektkosten	67
E.7	Betriebskosten	67
E.8	Risiken - Schadensskala	68
E.9	Risiken - Eintrittswahrscheinlichkeit	68
E.10	Risiken - Handlungen zur Senkung der Bewertung	68
E.11	Projektrisiken	69
E.12	Projektrisiken - Massnahmen	70
H.1	Akzeptanztest 01	91
H.2	Akzeptanztest 02	91
H.3	Akzeptanztest 03	92
H.4	Akzeptanztest 04	92

H.5 Akzeptanztest 05	93
H.6 Akzeptanztest 06	93
H.7 Akzeptanztest 07	94
H.8 Akzeptanztest 08	94
H.9 Akzeptanztest 09	95
H.10 Akzeptanztest 10	95
H.11 Akzeptanztest 11	96
H.12 Akzeptanztest 12	96
H.13 Akzeptanztest 13	97
H.14 Akzeptanztest 14	97
H.15 Akzeptanztest 15	98
H.16 Akzeptanztest 16	98
H.17 Akzeptanztest 17	99
H.18 Akzeptanztest 18	99
H.19 Akzeptanztest 19	100
H.20 Akzeptanztest 20	100
H.21 Akzeptanztest 21	101
H.22 Akzeptanztest 22	101
H.23 Akzeptanztest 23	102
H.24 Akzeptanztest 24	102
J.1 JavaScript Optimierungen	125
J.2 CSS Optimierungen	125
J.3 Bilder Optimierungen	125
J.4 Akzeptanztest 01	133
J.5 Akzeptanztest 02	133
J.6 Akzeptanztest 03	134
J.7 Akzeptanztest 04	134
J.8 Akzeptanztest 05	135
J.9 Akzeptanztest 06	135
J.10 Akzeptanztest 07	136
J.11 Akzeptanztest 08	136
J.12 Akzeptanztest 09	137
J.13 Akzeptanztest 10	137
J.14 Akzeptanztest 11	138
J.15 Akzeptanztest 12	138
J.16 Akzeptanztest 13	139
J.17 Akzeptanztest 14	139
J.18 Akzeptanztest 15	140
J.19 Akzeptanztest 16	140
J.20 Akzeptanztest 17	141
J.21 Akzeptanztest 18	141
J.22 Akzeptanztest 19	142
J.23 Akzeptanztest 20	142
J.24 Akzeptanztest 21	143
J.25 Akzeptanztest 22	143
J.26 Akzeptanztest 23	144
J.27 Akzeptanztest 24	144
J.28 Betrieb: Umgebungsvariablen	148
K.1 Controlling: Ziele	152

K.2 Anforderungskatalog . . . . .	154
-----------------------------------	-----



## Kapitel 1

# Initialisierung

### 1.1 Ausgangslage

Als regelmässiger Konzertbesucher wünsche ich mir eine Plattform im Internet, auf welcher ich eine zuverlässige Übersicht an Konzerten in meiner Umgebung vorfinde. Heute sind die Events nur verteilt auf verschiedenen Seiten wie die der Venues, des Konzertveranstalters, des Künstlers oder auf Facebook publiziert.

Ich möchte deshalb eine zentrale Plattform entwickeln, die es Benutzern einfach macht, Konzerte für ihren Geschmack zu finden. Die Plattform soll Genre unabhängig sein und entsprechende Filter anbieten. Den Benutzern der Plattform soll es möglich sein, Konzerte selber zu erfassen und pflegen.

Um einen zusätzlichen Service für den Benutzer zur Verfügung zu stellen, ist es auch denkbar, eine Art Notifikationssystem zu bauen um Benutzer über Handy-Notifications oder per Email an Konzerte oder Künstler zu erinnern.

Konzertveranstaltern kann das Erfassen ihrer Events vereinfacht werden, indem auf der Plattform erfasste Veranstaltungen direkt auf den Sozialen Medien wie Facebook, Twitter oder Instagram geteilt werden können.

## 1.2 Projektziele

Folgende Ziele sind in der Initialisierungsphase definiert worden:

Nr.	Zielbeschreibung	Muss/Kann
<b>Produktziele</b>		
1.1	Besucher können im Produkt nach Konzerten suchen	<b>Muss</b>
1.2	Suchresultate können nach Musik-Genre und Ort gefiltert werden	<b>Muss</b>
1.3	Besucher können Details zu einem Konzert ansehen	<b>Muss</b>
1.4	Das Produkt soll ein modernes responsives Design vorweisen	<b>Muss</b>
1.5	Konzerte sollen von Suchmaschinen indexiert werden können	<b>Muss</b>
1.6	Benutzer können sich im Produkt registrieren	<b>Muss</b>
1.7	Benutzer können ihr Passwort nach Verlust neu setzen	<b>Muss</b>
1.8	Inhalte des Portals sind durch die Benutzer erfassbar und bearbeitbar	<b>Muss</b>
1.9	Kompatibilität mit aktuellem Google Chrome und Mozilla Firefox Browser	<b>Muss</b>
1.10	Konzerte können vom Produkt nach Facebook exportiert werden	Kann
1.11	Ein angemeldeter Benutzer kann vermerken ob er einem Konzert teilnimmt	Kann
1.12	Das Produkt soll sich an die Security Best-Practices von Open Web Application Security Project (OWASP) halten	<b>Muss</b>
<b>Abwicklungsziele</b>		
2.1	Das Projekt soll nach HERMES 5 unter Berücksichtigung der Richtlinien von der Telematikschule Bern (TSBE) dokumentiert werden	<b>Muss</b>
2.2	Das Produkt muss bis Projektende fertiggestellt, getestet und bereit für die Einführung sein	<b>Muss</b>
2.3	Die Technische-Umsetzung wird durch Damian Senn erstellt	<b>Muss</b>
2.4	Die Kommunikation zwischen Experten und Diplomanden erfolgt wie im Projektauftrag E.7.2 beschrieben.	<b>Muss</b>
2.5	Das Projekt muss bis Ende Mai 2019 abgeschlossen sein	<b>Muss</b>

TABELLE 1.1: Ziele

## 1.3 Projektorganisation

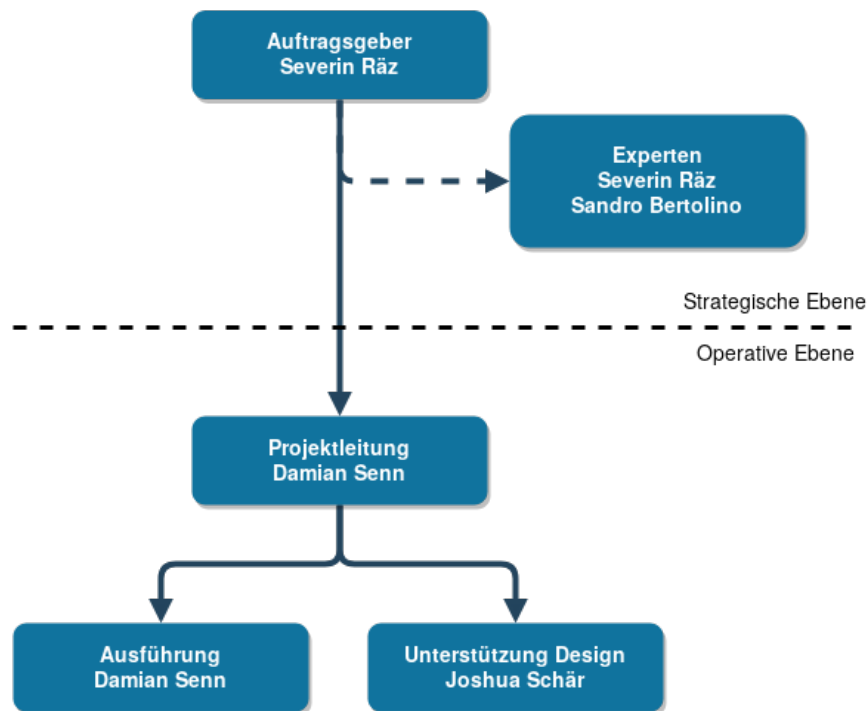


ABBILDUNG 1.1: Organigramm

### 1.3.1 Tätigkeiten im Projekt

Für die Freigaben der Phasen ist nach Absprache mit Severin Rätz Damian Senn selbstständig verantwortlich.

Name	Funktions- und Tätigkeitsbereich
Severin Rätz	Auftraggeber, externer Experte
Sandro Bertolino	Interner Experte
Damian Senn	Projektleiter, Ausführung
Joshua Schär	Unterstützung Design

TABELLE 1.2: Tätigkeiten Verteilung

### 1.3.2 Kommunikation

Im Kickoff-Meeting wurde besprochen, dass Damian Senn alle zwei Wochen einen kurzen Bericht an Sandro Bertolino und Severin Rätz per E-Mail schicken wird. Im Bericht soll erläutert werden, was in der Zwischenzeit erledigt wurde und was die nächsten Schritte im Projekt sind. Die Reports sind im Anhang **M** zu finden.

Aktivität	Dauer [h]			Status	Wer																											
						Februar				März				April					Mai				Juni									
	Soll	Ist	Abw.			06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27					
Initialisierung	64	71	7																													
1.1 Projektinitialisierung erstellen	4	4	0	erledigt	DS																											
1.2 Projektorganisation	2	2	0	erledigt	DS																											
1.3 Projektziele und Abgrenzungen	4	4	0	erledigt	DS																											
1.4 Vorbereitung Kick Off & Meeting	8	8	0	erledigt	DS,SB,SR																											
1.5 Projektplan	12	16	4	erledigt	DS																											
1.6 Anforderungskatalog	4	6	2	erledigt	DS																											
1.7 Risikoanalyse	4	4	0	erledigt	DS																											
1.8 Varianten beschreiben	8	7	-1	erledigt	DS																											
1.9 Varianten evaluieren & auswählen	2	2	0	erledigt	DS																											
1.10 Wirtschaftlichkeit evaluieren	4	6	2	erledigt	DS																											
1.11 Projektauftrag erstellen	12	12	0	erledigt	DS																											
Konzept	66	45	-21																													
3.1 Portalnamen finden	2	2	0	erledigt	DS																											
3.2 Screens definieren	8	4	-4	erledigt	DS																											
3.3 Screens designen	24	16	-8	erledigt	DS,JS																											
3.4 Software Architektur	12	9	-3	erledigt	DS																											
3.5 Test Konzept	12	6	-6	erledigt	DS																											
3.6 Zwischen-Meeting	8	8	0	erledigt	DS,SB,SR																											
Realisierung	136	108	-28																													
4.1 Screens in HTML/CSS umsetzen	24	24	0	erledigt	DS																											
4.2 Initialisierung Backend	8	2	-6	erledigt	DS																											
4.3 Implementation Registrierung/Login	8	8	0	erledigt	DS																											
4.4 Implementation Passwort Reset	8		-8	verworfen	DS																											
4.5 Implementation der Screens	24	55	31	erledigt	DS																											
4.6 Implementation Suche	16	8	-8	erledigt	DS																											
4.7 Tests erstellen	48	11	-37	unvollständig	DS																											
Abschluss	36	12	-24																													
5.1 Management Summary	4	4	0	erledigt	DS																											
5.2 Bericht ausdrucken, binden & senden	8	8	0	in arbeit	DS																											
5.3 Diplomarbeit bewerten	16		-16	geplant	SB,SR																											
5.4 Abschluss Meeting	8		-8	geplant	DS,SB,SR																											

Total / bereits benötigt / Restliche Stunden:

302 236 66

Geplant Wie geplant Ausserhalb Planung Meilenstein

#### Legende

Name	Abk.
Damian Senn	DS
Sandro Bertolino	SB
Severin Rätz	SR
Joshua Schär	JS



## 1.5 Lieferergebnisse

Folgende Ergebnisse wurden in der Phase Initialisierung erstellt:

- Studie (Anhang **D**)
- Projektauftrag (Anhang **E**)

## 1.6 Terminplan

Nachfolgend ist der grobe Terminplan der während der Initialisierungsphase erstellt wurde für die geplanten Phasen.

Phase	Datum	Stunden
Initialisierung	06.03.2019 - 31.03.2019	64
Konzept	01.04.2019 - 21.04.2019	66
Realisierung	22.04.2019 - 19.05.2019	136
Abschluss	20.05.2019 - 21.05.2019	36
Total:		302

TABELLE 1.3: Terminplan

## 1.7 Meilensteine

Im Projektplan wurden folgende Meilensteine und Termine festgelegt:

Nr.	Meilenstein	KW	Datum
1	Kickoff-Meeting	10	06.03.2019
2	Abschluss Phase Initialisierung	13	31.03.2019
3	Zwischen-Meeting	18	24.04.2019
4	Abschluss Phase Konzept	16	21.04.2019
5	Abschluss Phase Realisierung	20	14.05.2019
6	Abschluss Phase Abschluss	21	19.05.2019
7	Abschluss-Meeting	22	21.05.2019

TABELLE 1.4: Meilensteine

## 1.8 Risiken

Die Risikobewertung erfolgt mit folgender Formel:

$$\text{Bewertung} = \text{Schaden} \times \text{Eintrittswahrscheinlichkeit}$$

Weitere Details zur Bewertung der Risiken sind im Projektauftrag Anhang [E.12](#) zu finden.

### 1.8.1 Projektrisiken

Nr.	Risiko	Auswirkung	Schaden	Wahrsch.	Bewertung
1	Ausfall des Entwicklers oder Projektleiters	Verzögerungen von Arbeiten	4	3	Mittel
2	Unvollständige Projektdokumentation	Schlechtere Diplomarbeit Bewertung	4	2	Mittel
3	Schlechter Projektplan	Verzögerungen und eventuelle Qualitätsverluste	4	3	Mittel
4	Keine Benutzer	Das Produkt wird nicht von Benutzern eingesetzt	3	4	Mittel
5	Technisch nicht umsetzbare Features	Das Produkt kann nicht wie angedacht benutzt werden	4	3	Mittel

TABELLE 1.5: Projektrisiken

### 1.8.2 Massnahmen

Nr.	Massnahme	Handlung	Bewertung nach Massnahme		
			Schaden	Wahrsch.	Bewertung
1	Arzt aufsuchen, ggf. Projekt-Pause oder Abbruch	Akzeptanz	4	3	Mittel
2	Statusbericht alle zwei Wochen, bei Fragen sofort Hilfe suchen	Verminderung	2	1	Gering
3	Genügend Buffer-Zeit einplanen, ggf. Ferientage für Projekt einsetzen	Verminderung	2	1	Gering
4	Das Produkt löst vor allem ein persönliches Interesse	Akzeptanz	3	4	Mittel
5	Vereinfachte Alternativen in Konzept-Phase untersuchen	Verminderung	2	2	Gering

TABELLE 1.6: Projektrisiken - Massnahmen

### 1.8.3 Risikodiagramm

Die Risikodiagramme sind im Projektauftrag im Anhang [E.12.3](#) und Anhang [E.12.4](#) zu finden.

## 1.9 Abgrenzungen

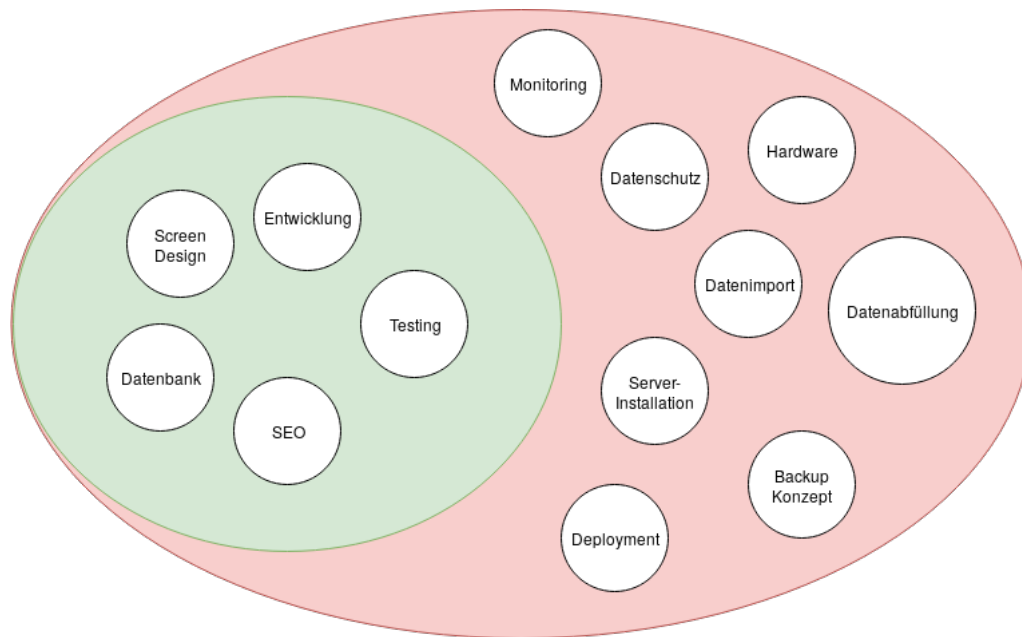


ABBILDUNG 1.2: Abgrenzungen

### Hardware, Server-Installation, Deployment und Monitoring

Da das Projekt ein reines Software-Entwicklungs Projekt ist, werden keine Operativen tätigkeiten wie Hardwarebeschaffung, Server-Installation, Deployment und das einrichten eines Monitoring-Systems vorgenommen.

### Datenimport

Da wir bisher keine existierenden Konzertdaten besitzen, ist es nicht nötig, einen Datenimport zu implementieren.

### Datenabfüllung

Die Projektarbeit beinhaltet kein Datenset, Tests werden mit Testdaten abgewickelt. Es liegt nicht in der Verantwortung des Projektleiters, dass Daten in die Applikation abgefüllt werden.

### Weitere Abgrenzungen

Weitere detaillierte Erklärungen zu den einzelnen Abgrenzungen sind im Projektauftrag [E.8](#) zu finden.

## 1.10 Studie

In der Studie wurden folgende Arbeiten abgehandelt:

- der Anforderungskatalog wurde definiert
- die Evaluation der Browser Software-Technologien
- die Evaluation der Server Software-Technologien
- die Evaluation der Testing Software-Technologien
- eine Kostenschätzung und mögliche Wirtschaftlichkeit ausgerechnet

### 1.10.1 Informationsbeschaffung

Folgende Quellen wurden in diesem Projekt für die Informationsbeschaffung genutzt:

Quelle	Beschreibung
Schulwissen / Berufserfahrung	Die Grundlage für die Umsetzung dieses Projekts wird durch mein existierendes Schulwissen sowie meine langjährige Berufserfahrung in der Software-Entwicklung gesetzt.
Internet	Ein Grossteil der Informationen werden heute über das Internet bezogen, für die Evaluation von Technologien und Lösungsansätzen wird einiges über das Internet recherchiert werden müssen.
Externer Experte	Bei konzeptionellen sowie technischen Fragen kann der externe Experte um Rat gefragt werden.

TABELLE 1.7: Informationsbeschaffung

### 1.10.2 Anforderungskatalog

Im Anforderungskatalog werden die Muss- und Kann-Kriterien definiert. Muss-Kriterien sind zwingend zu erfüllen, Kann-Kriterien sind als optionale Erweiterung zu verstehen.

Feature	Titel	Nr.	Kriterium	Ziel	Muss
Suche	Suche nach Konzertname	1.1	Listet alle Konzerte die Wörter der Suche im Konzertnamen beinhalten	1.1	<b>Muss</b>
	Suche nach Konzertlocation	1.2	Schränkt die Such-Resultate nach gegebener Konzertlocation ein	1.2	<b>Muss</b>
	Suche nach Ort	1.2	Schränkt die Such-Resultate nach gegebenem Ort ein	1.2	<b>Muss</b>
	Suche nach Genre	1.2	Schränkt die Such-Resultate nach gegebenem Musik-Genre ein	1.2	<b>Muss</b>
Design	Desktop	2.1	Alle Ansichten haben eine Desktop-Optimierte Variante	1.4	<b>Muss</b>
	Tablet	2.2	Alle Ansichten haben eine Tablet-Optimierte Variante	1.4	<b>Muss</b>
	Mobile	2.3	Alle Ansichten haben eine Mobile-Optimierte Variante	1.4	<b>Muss</b>
	Browser Kompatibilität	2.4	Alle Ansichten müssen in aktuellem Google Chrome und Mozilla Firefox dem Grundlayout folgen	1.9	<b>Muss</b>
SEO	Indexierbarkeit	3.1	Das Produkt ist von Suchmaschinen indexierbar	1.5	<b>Muss</b>
	Linked Data	3.2	Konzert Detailseiten sind mit dem Event-Schema <sup>1</sup> ausgestattet	1.5	<b>Muss</b>
Benutzer	Registrierung	4.1	Besucher können sich einen Benutzer registrieren, Benutzernamen und E-Mail Adressen müssen einzigartig sein	1.6	<b>Muss</b>
	Passwort-Vergessen	4.2	Benutzer können sich einen Passwort-Reset Link anfordern	1.7	<b>Muss</b>
	Social	4.3	Benutzer können auf Konzerten vermerken ob sie teilnehmen oder nicht	1.11	<b>Kann</b>

<sup>1</sup><https://schema.org/MusicEvent>

Feature	Titel	Nr.	Kriterium	Ziel	Muss
Erfassung	Artist	5.1	Benutzer können Artisten mit einem Genre erfassen	1.8	<b>Muss</b>
	Location	5.2	Benutzer können eine Konzertlocation mit Ort/Strasse erfassen	1.8	<b>Muss</b>
	Konzert	5.3	Benutzer können ein Konzert mit Konzertlocation und Artisten erfassen	1.8	<b>Muss</b>
	Facebook	5.4	Benutzer können ein Konzert in ein Facebook-Event exportieren	1.10	Kann
Security	SQL-Injection	6.1	Das Produkt soll resistent gegen SQL-Injection sein	1.12	<b>Muss</b>
	HTML-Injection	6.2	Das Produkt soll resistent gegen HTML-Injection / XSS sein	1.12	<b>Muss</b>
	Passwort encryption	6.3	Passwörter von Benutzer müssen mit einem sicheren Verfahren gespeichert werden	1.12	<b>Muss</b>
	Session	6.4	Session-Cookies dürfen nicht durch JavaScript ausgelesen werden	1.12	Kann
Performance	Ladezeit	7.1	Die Seitenansichten dürfen nicht länger als 6 Sekunden auf einem 3G Netz laden		<b>Muss</b>
Sonstiges	User Tracking	8.1	Benutzerverhalten soll analysiert und nachvollziehbar sein.		Kann

TABELLE 1.8: Anforderungskatalog

## 1.11 Evaluation Browser-Technologie

Die genaue Erklärung zur Gewichtung der Browser-Technologie Evaluation sind in der Studie im Anhang [D.4](#) zu finden.

### 1.11.1 Variante: React

Die JavaScript Library **React** ist heute die wohl beliebteste Technologie um interaktive Applikationen im Web zu bauen.

React ermöglicht es den Entwicklern Screens auf eine deklarative Weise zu definieren, so dass sich Elemente jeweils dem Zustand der Applikation anpassen.

Die Features von React sind minimal gehalten und sehen auf den ersten Blick einfach aus, jedoch wird für Anwendungen schnell klar, dass zusätzliche Software-Libraries benötigt werden um eine grössere Applikation zu entwickeln.

Durch das Hinzufügen von weiteren Libraries steigt die Komplexität dieser Variante stark an, da es im React Ökosystem für viele Lösungen diverse verschiedene Lösungsansätze gibt.

### 1.11.2 Variante: Next.js

**Next.js** ist ein JavaScript Framework, das auf der **React** Library aufbaut und zusätzliche Features sowie gängige Konventionen mitbringt.

Dadurch dass Next.js ein komplettes Framework ist und nicht nur eine Library, leidet diese Variante weniger der Komplexität eines kompletten Eigenbaus mit React. Features wie die Navigation von Seite zu Seite bringt Next.js bereits mit.

### 1.11.3 Variante: SSR

**SSR** steht für **Serverside Rendering** und beschreibt die klassische Methode vom Erstellen von Webseiten, indem man Hypertext Markup Language (HTML) auf dem Server generiert und zum Browser schickt.

Dies hat nach wie vor seine Daseinsberechtigung, da dies wenig Komplexität mit sich bringt, einen schnelleren Seitenaufbau garantiert und ohne zusätzlichen Aufwand von Suchmaschinen indexiert werden kann.

## 1.12 Entscheid Browser-Technologie

Die Bewertung der Browser-Technologie sind in der Studie im Anhang [D.5](#) zu finden.

Durch die Evaluierung wurde klar, dass das Einsetzen eines JavaScript-Frameworks zuviel zusätzliche Komplexität und gewisse Einbussungen in Performance und Stabilität unvermeidbar ist. Somit ist ein die Wahl für eine klassische Server-Side Rendered (SSR) Webseite favorisierend.

Es ist durchaus vorstellbar, dass in einem zweiten Schritt, nach diesem Projekt, die SSR Applikation durch eine Next.js Applikation ersetzt werden könnte.

### Kosten / Wirtschaftlichkeit

Da die Programmiersprache Elixir sowie das Framework Phoenix unter einer Open Source Lizenz verfügbar sind, fallen bei dieser Lösung keine zusätzlichen Kosten an.



## 1.13 Evaluation Server-Technologie

Die genaue Erklärung zur Gewichtung der Server-Technologie Evaluation sind in der Studie im Anhang D.7 zu finden.

### 1.13.1 Variante: Node.js / koa.js

Auch auf dem Server gewinnt JavaScript immer mehr an Beliebtheit. Mit Node.js und koa.js können schnell kleinere und simplere Applikationen erstellt werden, die dennoch sehr performant sind.

koa.js ist eine Library für Server-Applikationen und bietet nur eine einfache Basis und bringt keine Features für Datenpersistenz oder HTML Templates mit sich. Diese Features müssen durch zusätzliche Module installiert werden.

### 1.13.2 Variante: Elixir / Phoenix

Elixir ist eine Programmiersprache die eine sehr stabile und performante Grundlage bietet. Durch das Framework Phoenix, wird im Elixir Ökosystem ein starkes feature umfangreiches Web-Framework angeboten. Phoenix beinhaltet eine grosse Menge an Features mit sich und gibt dem Entwickler direkt Funktionalitäten wie HTML Templates, Datenpersistenz und ein einfaches Test-Framework.

### 1.13.3 Variante: Next.js

Next.js wurde bereits als Variante für die Browser-Technologie in Betracht gezogen. Ein zusätzliches Feature von Next.js ist, dass die Applikation auch auf dem Server betrieben werden kann. Das Einsetzen der selben Technologie kann bedeutende Vorteile mit sich bringen, so muss man nur ein Framework lernen und kann Programmcode auf dem Server mit der Applikation im Browser geteilt werden.

## 1.14 Entscheid Server-Technologie

Die Bewertung der Server-Technologie sind in der Studie im Anhang D.8 zu finden.



ABBILDUNG 1.3: Phoenix Framework Logo

Quelle: <https://github.com/phoenixframework/phoenix>

Durch das grosse Featurset von Phoenix sowie tieferer Komplexität gegenüber den beiden anderen Varianten hat sich Phoenix für die Server-Technologie klar durchgesetzt.

### Kosten / Wirtschaftlichkeit

Da die Programmiersprache Elixir sowie das Framework Phoenix unter einer Open Source Lizenz verfügbar sind, fallen bei dieser Lösung keine zusätzlichen Kosten an.

## 1.15 Evaluation Testing-Technologie

Die genaue Erklärung zur Gewichtung der Testing-Technologie Evaluation sind in der Studie im Anhang [D.10](#) zu finden.

### 1.15.1 Jest + Puppeteer

Jest ist ein JavaScript-Test Framework von Facebook. Durch die Kombination der Puppeteer Library von Google ist es möglich, automatisierte Browser-Tests durchzuführen.

### 1.15.2 Wallaby

Wallaby ist ein Elixir Browser-Test Framework, welches sich nahtlos mit Phoenix integrieren lässt. Wallaby unterstützt parallelisierung von Tests und ist daher ein guter Kandidat für eine hohe Anzahl von automatisierten Tests.

## 1.16 Entscheid Testing-Technologie

Die Bewertung der Testing-Technologie sind in der Studie im Anhang [D.11](#) zu finden.



ABBILDUNG 1.4: Wallaby Logo

Quelle: <https://github.com/keathley/wallaby>

Dadurch dass sich Wallaby einfach mit der ausgewählten Server-Technologie verwenden lässt, hohe Performance und Stabilität aufweist, ist Wallaby die knapp bessere Variante als eine Jest + Puppeteer kombination.

Leider hat Wallaby keine Visualtesting Integration mit dem Dienst percy.io, dies könnte aber im verlaufe der Umsetzung eventuell im Rahmen dieser Arbeit umgesetzt werden.

### Kosten / Wirtschaftlichkeit

Auch Wallaby ist unter eines Open Source Lizenz veröffentlicht und erzeugt somit im Projekt keine direkten zusätzlichen Kosten. Durch fehlende Unterstützung von Visualtesting, wird im Verlauf der Realisierung zusätzlicher Aufwand erzeugt, entweder durch eigene Implementation solcher Tests oder durch unbemerkte Fehler die sich bei Änderungen eingeschlichen haben.

## 1.17 Wirtschaftlichkeit

### 1.17.1 Projektkosten

Für die Berechnung der Projektkosten wird ein Stundensatz von 150.- CHF angenommen.

Phase	Geplante Stunden	Kosten
Initialisierung	64	9'600.- CHF
Konzept	66	9'900.- CHF
Realisierung	136	20'400.- CHF
Abschluss	36	5'400.- CHF
<b>Total:</b>	302	42'900.- CHF

TABELLE 1.9: Projektkosten

Die geplanten Projektkosten betragen somit **42'900.- CHF**.

Kostenstelle	Jährliche Kosten
Software	Keine
.com Domain	20.- CHF
Hosting	1'800.- CHF
<b>Total:</b>	1'820.- CHF

TABELLE 1.10: Betriebskosten

Für die Betriebskosten eines Hostings wird einen durchschnittlichen monatlichen Preis von 150.- CHF angenommen, da das Deployment für dieses nicht vorgesehen ist, ist dies eine von Damian Senn geschätzte Zahl.

### 1.17.2 Break Even Analyse

Beim Modell «Gigboost» wird eine Option angeboten bei der publizierten Gigs auf der Startseite sowie in der Suche anderen Einträgen bevorzugt dargestellt werden. Für einen Gegenpreis von 10.- CHF kann man einen Gig «boosten».

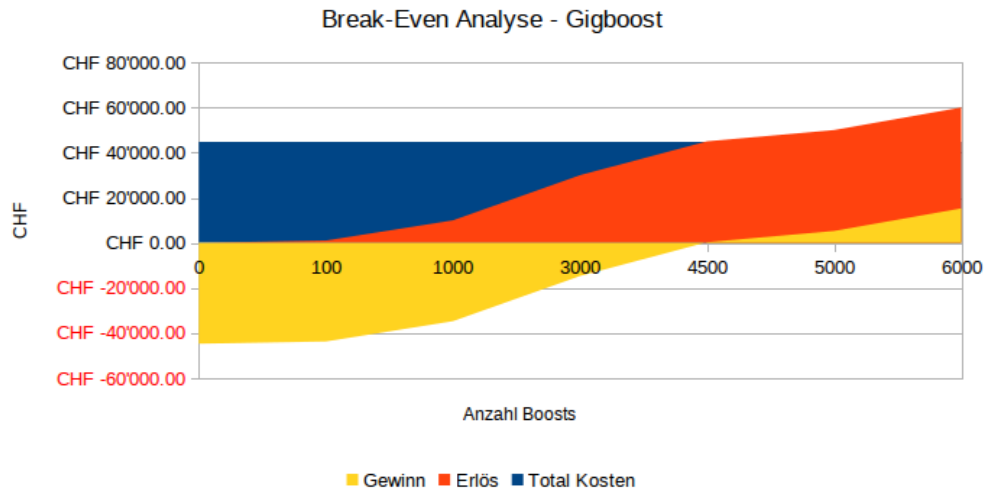


ABBILDUNG 1.5: Break-Even Analyse - Gigboost

Im Modell «Werbung» wird ausgerechnet wieviele aktive Benutzer das Produkt benötigt um in den nächsten Jahren Gewinn zu erzielen.

Durch Annahme von einem Erlös von 140.- CHF pro 40'000 Besucher<sup>2</sup> erhalten wir folgendes Bild:

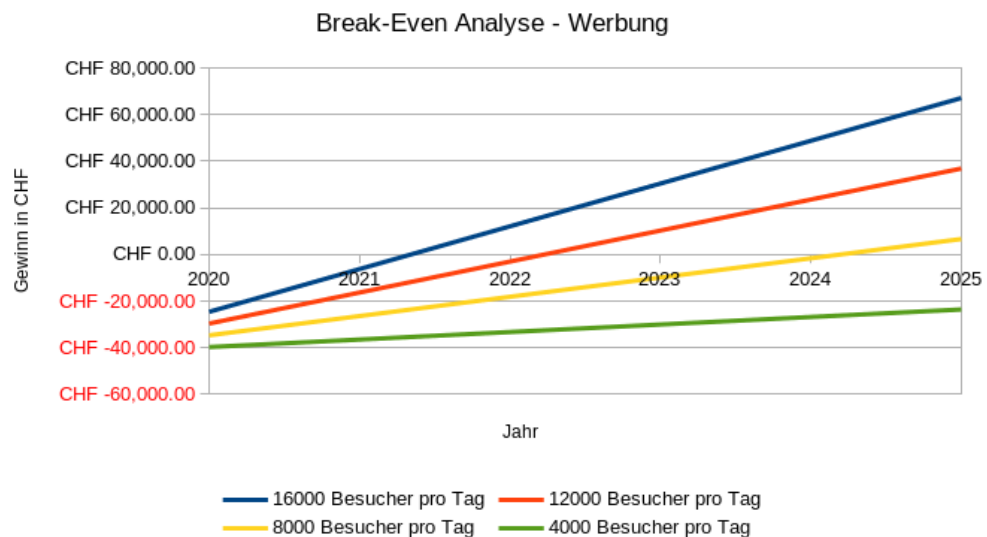


ABBILDUNG 1.6: Break-Even Analyse - Werbung

Die genauere Ausrechnung der Wirtschaftlichkeit im Modell Werbung sind in der Studie im Anhang D.13.2 zu finden.

<sup>2</sup><https://www.quora.com/How-much-does-Google-AdSense-pay-for-3-banners-on-a-webpage-per-1-000-views/answer/Manas-Sahu-59>

## Kapitel 2

# Konzept

Durch die in der Studie gewonnenen Erkenntnissen, werden in der Phase Konzept verschiedene Teilkonzepte erstellt.

Im Teilkonzept «Portalname» wird der Name des Produktes erarbeitet.

Im Teilkonzept «Design- und Bedienkonzept» werden die Ansichten der Applikation in Mockups umgesetzt. Es werden die Benutzer Use-Cases vom Besucher sowie der Konzert-Erfasser aufgezeigt.

Im Teilkonzept «Softwarekonzept» werden die Datenflüsse hinter den Mockups aufgezeigt, sowie die Datenbankstruktur aufgebaut.

Im Teilkonzept «Testkonzept» werden die einzelnen Systemtests aufgelistet sowie ausgearbeitet wie granular welche Teile der Software getestet werden sollen.

Im letzten Teil des Konzept-Dokuments wird im Fazit dokumentiert, wie und warum das Konzept von den vorhergehenden Phasen des Projekts abweicht.

### 2.1 Portalname

Der Portalname wurde in einer Brainstorming-Session von Damian Senn auf den Namen «**Gigpillar**» festgelegt. Der Name ist angelehnt an die Werbepfeiler in Städten, wo oft Werbeplakate für Konzerte hängen.

Im Konzept im Anhang **H.2** ist eine Liste von Ideen für alternative Namen zu finden.

## 2.2 Design

### Homepage

Die Homepage ist die erste Seite, die der Besucher sieht, wenn er/sie die Applikation direkt über **gigpillar.com** aufruft. Auf den ersten Blick ist die Suche sowie ein grosses Bild (Banner) eines Gigs zu erblicken. Weiter sind Links zu gängigen Funktionalitäten wie Gig hinzufügen sowie das Login in einer Navigation erreichbar.

Unter dem Banner werden Gigs in nächster Nähe des Besuchers aufgelistet, der Link «change location» führt weiter zur Suchresultate Seite um den entsprechenden Filter anzupassen.

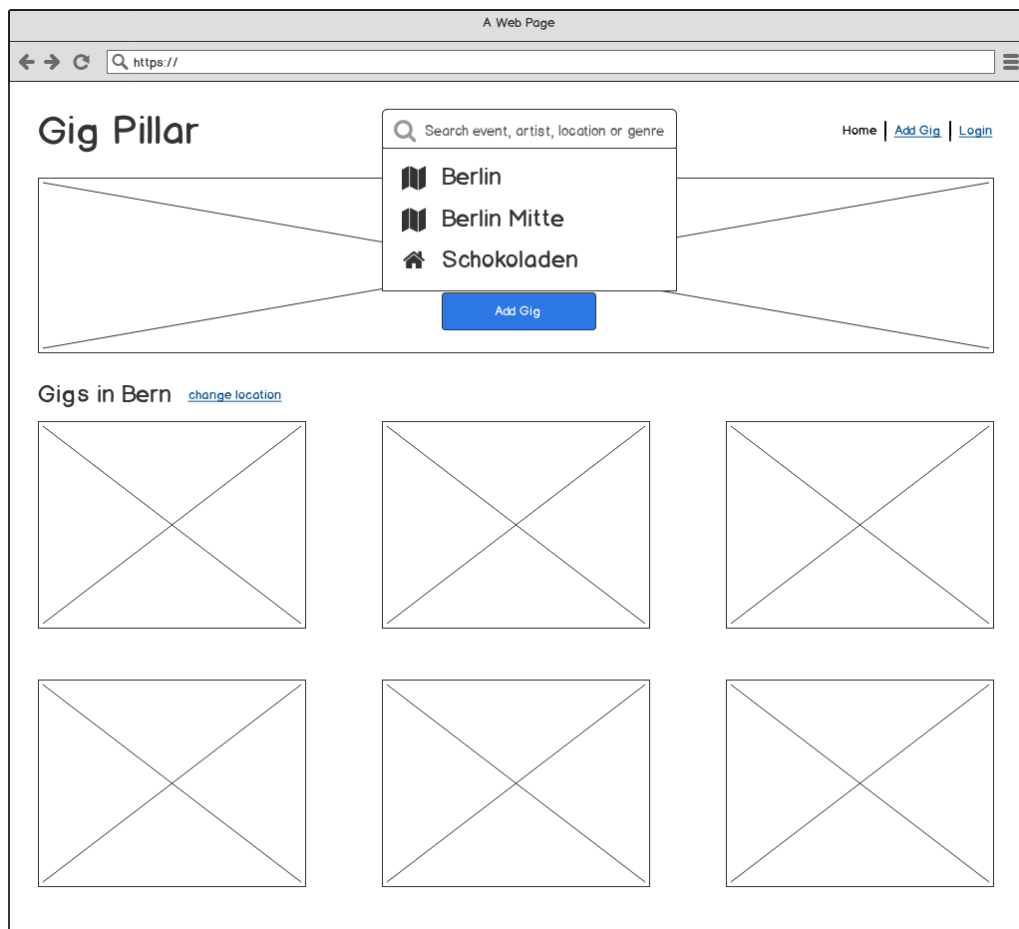


ABBILDUNG 2.1: Mockup: Homepage

## Suchresultate

Auf der Suchresultate Seite sieht der Benutzer seine Suchresultate der von der globalen Suchbox ausgelösten Suche. Die Seite bietet weitere Filter an um die Resultate weiter einzugrenzen.

Folgende Filter stehen den Benutzern zur Verfügung:

- Ort
- Datum von
- Datum bis
- Musik Genre

Das Anwählen eines Suchresultates führt den Benutzer weiter zur detaillierten Gig Ansicht.

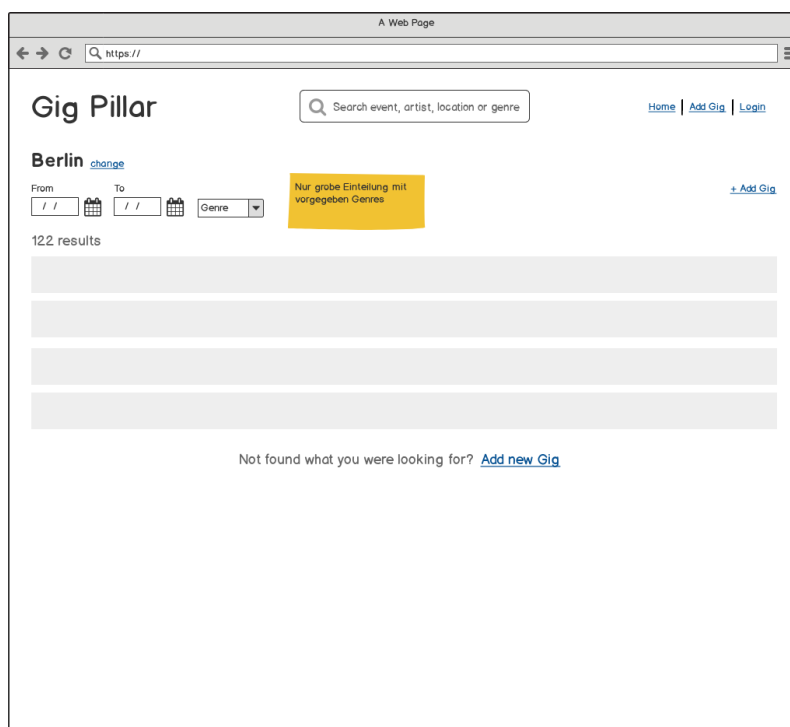


ABBILDUNG 2.2: Mockup: Suchresultate

## Gig Ansicht

In der Gig Ansicht werden alle Details zu einem Event aufgelistet.

- Datum des Events
- Zeit wann das Event beginnt, bzw die Location die Türen öffnet
- Liste aller Künstler mit optionaler Startzeit
- Eine Beschreibung des Events
- Die Adresse der Location mit Link auf Google Maps

Ausserdem soll es den Benutzern möglich sein, über einen «Add to my calendar» Link das Event zu seiner Kalender-Applikation zu importieren.

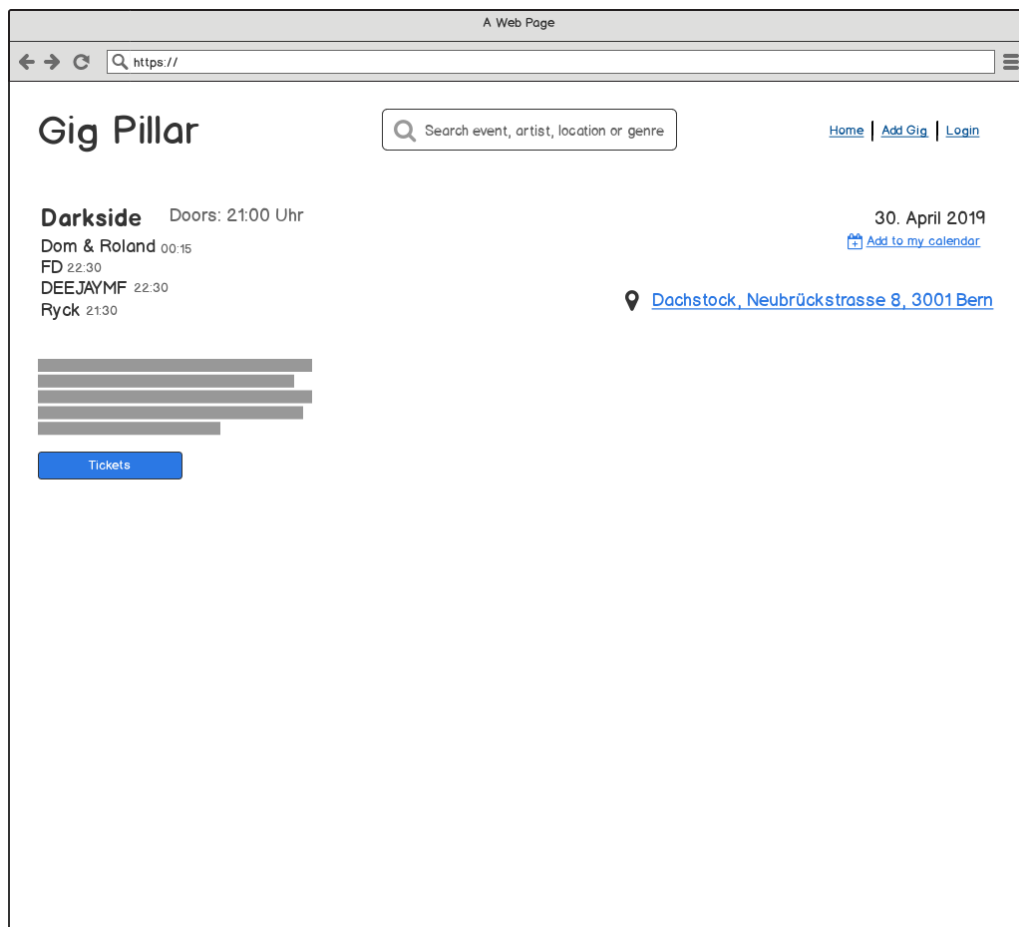


ABBILDUNG 2.3: Mockup: Gig Ansicht



## Gig erfassen

Benutzer können Gigs erfassen.

Folgende Daten sind für einen Gig zu erfassen:

- Name
- Bild (*optional*)
- Location
- Datum
- Zeit
- Eine Liste von Artists mit optionaler Startzeit
- Beschreibung
- Link zum Ticketvertreiber

The mockup shows a web browser window titled 'A Web Page' with the URL 'https://'. The page header features the 'Gig Pillar' logo, a search bar, and links for 'Home' and 'Add Gig'. The main content area is titled 'Add new Gig' and contains the following form elements:

- Gig name:** A text input field.
- Image:** A circular placeholder with an 'Add image' button.
- Where:** A text input field with a search icon and the placeholder text 'search location'.
- When:** A date and time selector with a calendar icon and a 'Doors: 20:00' label.
- Who:** A dropdown menu with a search icon and the placeholder text 'Choose artist'. An autocomplete dropdown is shown with the option 'Pennywise at 20:00'. A green arrow points from the 'Pennywise' text in the dropdown to a search input field on the right.
- Describe Gig:** A large text area for the gig description.
- Tickets:** A text input field with the placeholder text 'https://whosellsthesetickets.example.com'.
- Save Gig:** A blue button at the bottom of the form.

ABBILDUNG 2.4: Mockup: Gig erfassen

## Benutzerprofil

Benutzer können ihr eigenes Profil verwalten und folgende Tätigkeiten verrichten:

- Anzeigename ändern
- E-Mail Adresse ändern (*mit E-Mail Bestätigung*)
- Passwort ändern (*muss vorher altes Passwort bestätigen*)
- Account löschen (*muss doppelt bestätigt werden!*)

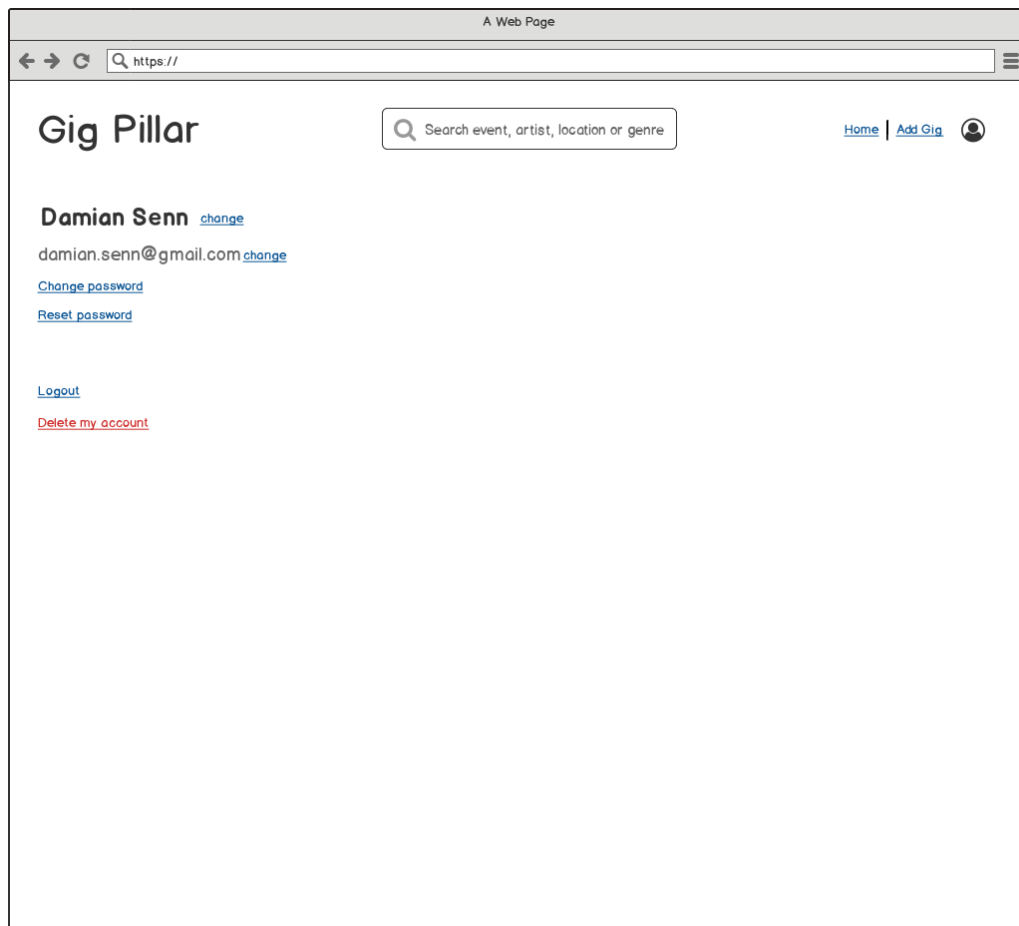


ABBILDUNG 2.5: Mockup: Benutzerprofil

## 2.3 Software

### 2.3.1 Datenfluss

Die Homepage zeigt den Besuchern Gigs in ihrer Nähe an, dazu muss über eine GeoIP API die IP-Adresse des Besuchers auf ein Land zurückverfolgt werden. Dazu wird beim ersten Besuch die GeoIP API abgefragt und das Land des Benutzers in eine Session geschrieben. Bei weiteren Aufrufen wird das Land direkt aus der Session bezogen.

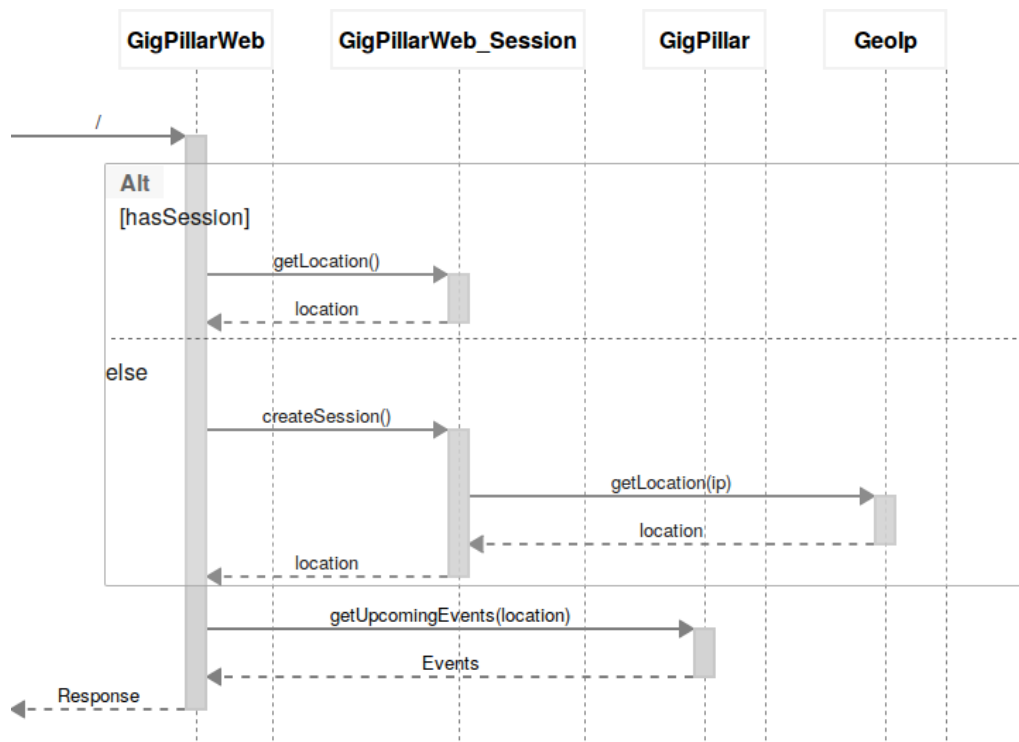


ABBILDUNG 2.6: Datenfluss: Homepage

Für weitere Datenflussdefinitionen, siehe Konzept Anhang [H.4.1](#).

### 2.3.2 Datenbankstruktur

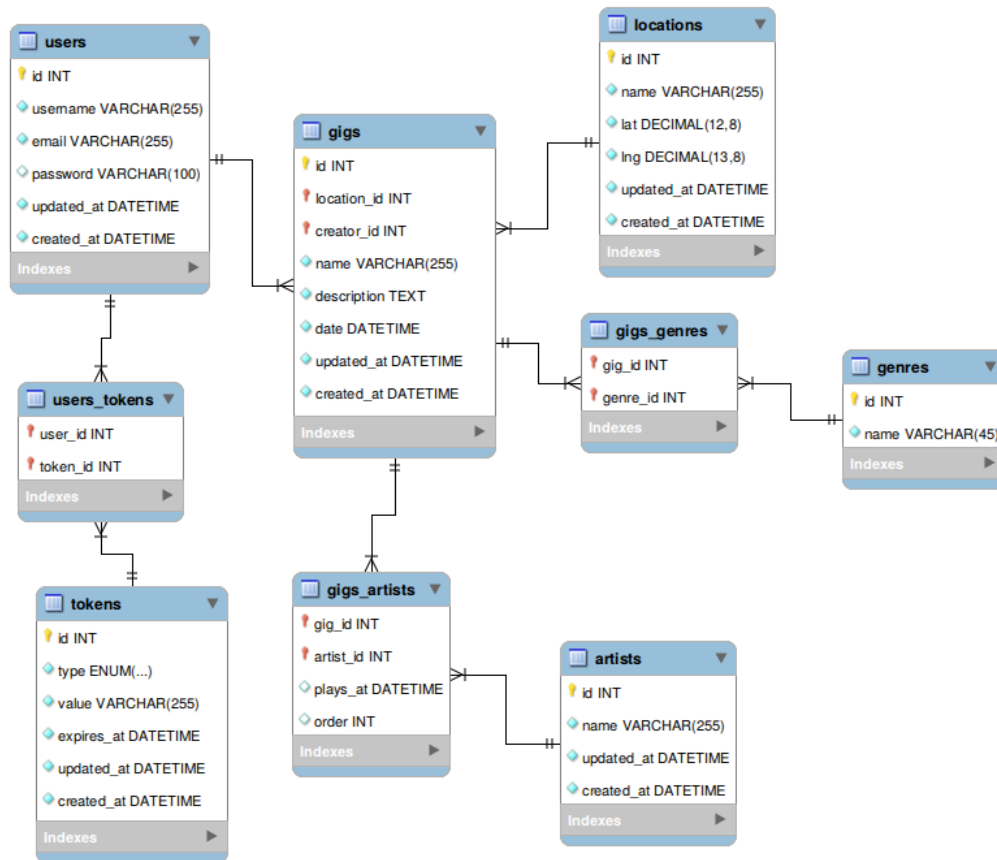


ABBILDUNG 2.7: Konzept: Entity Relationship Diagram

## 2.4 Testkonzept

### 2.4.1 Unit-Tests

Der Applikationscode soll mit Unit-Tests getestet werden.

Unit-Tests testen einzelne Funktionen, d.h. hier sind Aufrufe über Hypertext Transfer Protocol (HTTP), wie sie durch den Browser ausgelöst würden, oder Datenbankabfragen ausgeschlossen.

### 2.4.2 Integration-Tests

HTTP Requests sowie Datenabfragen sollen über Integration-Tests abgedeckt werden. In den Integration-Tests wird vor allem Business-Logik wie z.B. Validierungen von Daten getestet.

### 2.4.3 Browser-Tests

Mit der in der Studie evaluierten Testing Library «Wallaby» sollen die gängigsten Benutzer Use-Cases getestet werden.

### 2.4.4 Visual-Tests

Es soll für jede Ansicht während den Tests mindestens ein Screenshot für [percy.io](https://percy.io) erstellt werden.

### 2.4.5 Akzeptanztests

Die Akzeptanztest sind vom Projektleiter vor Abschluss der Realisierung durchzuführen. Der Umfang der Akzeptanztests basiert auf den Kriterien die im Anforderungskatalog definiert wurden.

Technische Kriterien wie die Indexierbarkeit wurden in den Akzeptanztests bewusst ausgelassen, die Tests sollen möglichst unabhängig vom System durchführbar sein.

Das ausführliche Testprotokoll ist im Konzept im Anhang [H.5.5](#) zu finden.

## 2.5 Fazit

### 2.5.1 Abweichungen

Das erstellte Datenbankschema weicht vom Anforderungskatalog leicht ab, so wird vorgesehen, dass für die Zuweisung von Musik-Genres nicht der Künstler sondern das Konzert/der Gig verwendet wird. Dies soll das Erfassen eines Gigs vereinfachen, da so nicht der Prozess des Erfassens unterbrochen werden muss, nur um einem Künstler das entsprechende Genre hinzuzufügen.

Ausserdem wurde Entschieden, dass für Künstler und Locations keine eigenen Verwaltungen implementiert werden. Die Daten werden von externen Quellen wie z.B. der Google Places API bezogen.

### 2.5.2 Probleme

Für die Location Autocomplete Funktion, ist es nicht klar, ob die Google Places API die nötigen Daten zur Verfügung stellt. So kann es gut sein, dass Locations nicht gefunden werden oder zuviele nicht relevante Resultate zurück gibt.

### 2.5.3 Machbarkeit

Für die Machbarkeit während dem im Konzept ein Test mit der Google Places API gemacht. Die Google Places API liefert alle nötigen Daten, um die gewünschten Features umzusetzen.

### 2.5.4 Wirtschaftlichkeit

In der Konzeptphase hat es keine Anzeichen auf Veränderungen, die die Wirtschaftlichkeit beeinflussen sollte.

### 2.5.5 Erweiterbarkeit

Es bestehen einige Ideen, wie man die Applikation erweitern könnte. Ein Feature wäre zum Beispiel, dass beim Erfassen eines Gigs sogleich ein Facebook-Event angelegt würde. Weiter denkbar wäre eine Filter-Funktion, die die Gigs basierend auf einem Radius in Kilometer um eine Stadt einschränken würde.

### 2.5.6 Projektplan

Trotz einer Verschiebung im Projektplans während der Initialisierung, wird nach der Konzeptphase der Projektplan soweit eingehalten werden können. Da die Screen-designs sowie das Testkonzept weniger Zeit beansprucht haben, bin ich nach dem Zwischenmeeting wieder im geplanten Bereich.

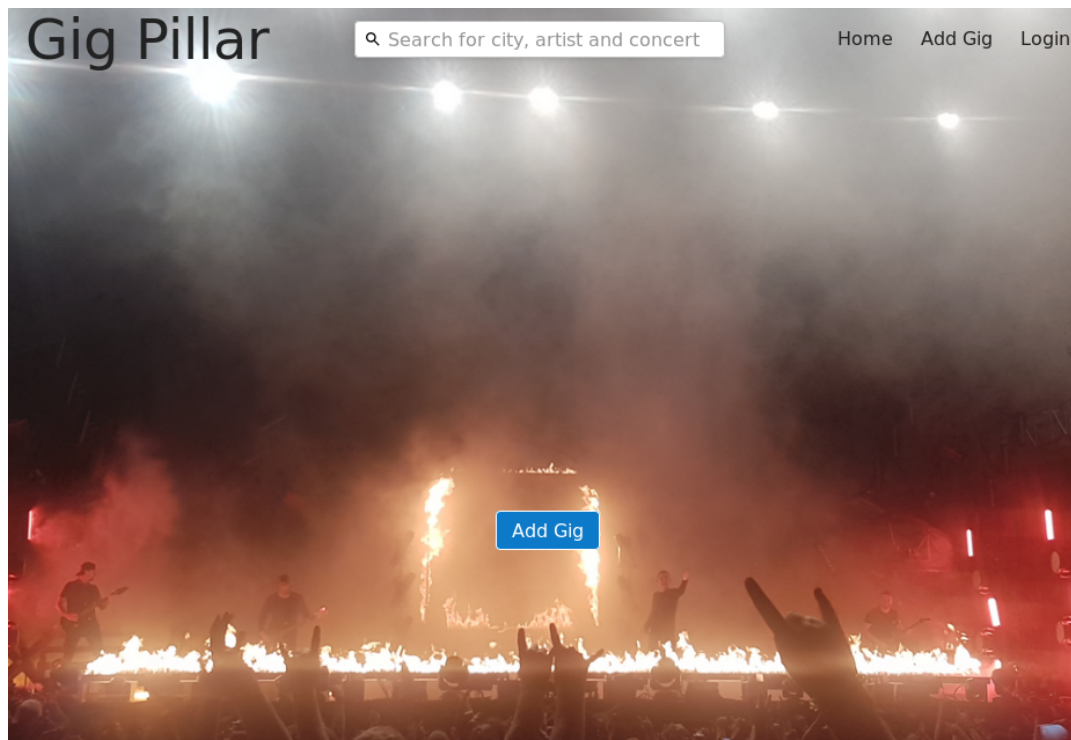
## Kapitel 3

# Realisierung

### 3.1 Umsetzung

#### 3.1.1 HTML-Prototyp

Im Rahmen der Umsetzung wurde ein HTML-Prototyp erstellt, dieser dient als Basis für die echte Applikation.



#### Gigs in Bern [change location](#)



24.12.2019  
Anti-Flag



24.12.2019  
Darkside



24.12.2019  
Liquid Session

ABBILDUNG 3.1: HTML Prototyp

### 3.1.2 Datenbankschema

Das finale Schema ist im Realisierungs Anhang J.4. Die folgenden Änderungen wurden am von der Konzeptphase vorgegebenen Schema vorgenommen.

#### Alle Entitäten

Alle «created\_at» Felder wurden nach «inserted\_at» umbenannt, da dies die Standardbenennung des Phoenix Frameworks ist.

#### User

Der Benutzer Entität wurde das Feld «password» nach «password\_hash» umbenannt, damit klar ist, dass nicht ein Passwort sondern nur ein Hash abgespeichert wird. Das Passwort wird mit Argon2 verschlüsselt und ist somit eine sichere Verschlüsselung.

#### Genre

Der Genre Entität sind im Konzept die Datumsfelder «update\_at» und «inserted\_at» vergessen gegangen und wurden in der Realisierung nachgeführt.

#### Gig

In der Gig Entität wurden drei weitere Felder hinzugefügt. Die Felder «uuid» und «picture» dienen dazu, die beim Erfassen sowie Bearbeiten eines Gigs hochgeladenen Bilder zu identifizieren. Das zusätzliche Feld «tickets» ermöglicht es, beim Erfassen eines Gigs einen Link zum Ticketvorverkauf zu hinterlegen.

#### Location

Die Location Entität erhielt bei der Realisierung zwei neue Felder, «address» für die Adresse der Location und «google\_place\_id» um die Referenz der Google API zu erhalten.

## 3.2 Tests

Bis auf fünf Tests im Testprotokoll konnten alle mit einem **OK** durchgeführt werden. Einige Features wurden aufgrund von mangelnder Zeit nicht umgesetzt. Genauere Details sind im Test Protokoll im Realisierungsanhang J.15.



## Kapitel 4

# Einführung

### 4.1 Projektcontrolling

#### 4.1.1 Erfüllung der Ziele

Mit der folgenden Tabelle wird aufgezeigt, welche Projektziele erfüllt, und welche Projektziele **nicht** erfüllt wurden. Projektziele die nicht erfüllt wurden, müssen genauer erläutert werden wieso diese nicht erfüllt wurden.

Nr.	Zielbeschreibung	Erfüllt
<b>Produktziele</b>		
1.1	Besucher können im Produkt nach Konzerten suchen	Ja
1.2	Suchresultate können nach Musik-Genre und Ort gefiltert werden	Ja
1.3	Besucher können Details zu einem Konzert ansehen	Ja
1.4	Das Produkt soll ein modernes Responsive Design vorweisen	Ja
1.5	Konzerte sollen von Suchmaschinen indexiert werden können	Ja
1.6	Benutzer können sich im Produkt registrieren	Ja
1.7	Benutzer können ihr Passwort nach Verlust neu setzen	<b>NEIN</b>
1.8	Inhalte des Portals sind durch die Benutzer erfassbar und bearbeitbar	Ja
1.9	Kompatibilität mit aktuellem Google Chrome und Mozilla Firefox Browser	Ja
1.10	Konzerte können vom Produkt nach Facebook exportiert werden	Nein
1.11	Ein angemeldeter Benutzer kann vermerken ob er einem Konzert teilnimmt	Nein
1.12	Das Produkt soll sich an die Security Best-Practices von OWASP halten	Ja
<b>Abwicklungsziele</b>		
2.1	Das Projekt soll nach HERMES 5 unter Berücksichtigung der Richtlinien von der TSBE dokumentiert werden	Ja
2.2	Das Produkt muss bis Projektende fertiggestellt, getestet und bereit für die Einführung sein	Ja
2.3	Die Technische-Umsetzung wird durch Damian Senn erstellt	Ja

Nr.	Zielbeschreibung	Erfüllt
2.4	Die Kommunikation zwischen Experten und Diplomanden erfolgt wie im Projektauftrag <a href="#">E.7.2</a> beschrieben.	Ja
2.5	Das Projekt muss bis Ende Mai 2019 abgeschlossen sein	Ja

TABELLE 4.1: Controlling: Ziele

Das Ziel 1.7 konnte nicht erfüllt werden. In Retrospektive, hätte dieses Feature kein MUSS Kriterium sein müssen. Die verfügbare Zeit der Realisierung wurde in wichtigere Kernkomponenten des Produktes investiert.

## 4.1.2 Erfüllung der Anforderungen

Feature	Titel	Nr.	Kriterium	Erfüllt
Suche	Suche nach Konzertname	1.1	Listet alle Konzerte die Wörter der Suche im Konzertnamen beinhalten	Ja
	Suche nach Konzertlocation	1.2	Schränkt die Such-Resultate nach gegebener Konzertlocation ein	Ja
	Suche nach Ort	1.2	Schränkt die Such-Resultate nach gegebenem Ort ein	Ja
	Suche nach Genre	1.2	Schränkt die Such-Resultate nach gegebenem Musik-Genre ein	Ja
Design	Desktop	2.1	Alle Ansichten haben eine Desktop-Optimierte Variante	Ja
	Tablet	2.2	Alle Ansichten haben eine Tablet-Optimierte Variante	Ja
	Mobile	2.3	Alle Ansichten haben eine Mobile-Optimierte Variante	Ja
	Browser Kompatibilität	2.4	Alle Ansichten müssen in aktuellem Google Chrome und Mozilla Firefox dem Grundlayout folgen	Ja
SEO	Indexierbarkeit	3.1	Das Produkt ist von Suchmaschinen indexierbar	Ja
	Linked Data	3.2	Konzert Detailseiten sind mit dem Event-Schema <sup>1</sup> ausgestattet	Ja
Benutzer	Registrierung	4.1	Besucher können sich einen Benutzer registrieren, Benutzernamen und E-Mail Adressen müssen einzigartig sein	Ja
	Passwort-Vergessen	4.2	Benutzer können sich einen Passwort-Reset Link anfordern	NEIN
	Social	4.3	Benutzer können auf Konzerten vermerken ob sie teilnehmen oder nicht	Nein

<sup>1</sup><https://schema.org/MusicEvent>

Feature	Titel	Nr.	Kriterium	Erfüllt
Erfassung	Artist	5.1	Benutzer können Artisten mit einem Genre erfassen	NEIN
	Location	5.2	Benutzer können eine Konzertlocation mit Ort/Strasse erfassen	NEIN
	Konzert	5.3	Benutzer können ein Konzert mit Konzertlocation und Artisten erfassen	Ja
	Facebook	5.4	Benutzer können ein Konzert in ein Facebook-Event exportieren	Nein
Security	SQL-Injection	6.1	Das Produkt soll resistent gegen SQL-Injection sein	Ja
	HTML-Injection	6.2	Das Produkt soll resistent gegen HTML-Injection / XSS sein	Ja
	Passwort encryption	6.3	Passwörter von Benutzer müssen mit einem sicheren Verfahren gespeichert werden	Ja
	Session	6.4	Session-Cookies dürfen nicht durch JavaScript ausgelesen werden	Ja
Performance	Ladezeit	7.1	Die Seitenansichten dürfen nicht länger als 6 Sekunden auf einem 3G Netz laden	Ja
Sonstiges	User Tracking	8.1	Benutzerverhalten soll analysiert und nachvollziehbar sein.	Nein

TABELLE 4.2: Anforderungskatalog

### **Nicht erfüllte Anforderungen**

Wie bereits bei den Zielen ist auch bei den Anforderungen die Passwort-Reset (4.2) Funktionalität nicht erfüllt. Zusätzlich sind die Anforderungen 5.1 und 5.2 nicht erfüllt. Die beiden Anforderungen wurden nicht umgesetzt, da in der Konzeptphase entschieden wurde, dass die Konzertlocations nur über die Google Places API ausgewählt werden können. Somit gibt es für die Anforderung 5.2 keine Umsetzung. Auch die Anforderung 5.1 wurde in der Konzeptphase angepasst, so wird nicht einem Artisten das Genre sondern einem Gig die Genres angehängt. Allerdings wurde das Feature aus Zeitgründen noch nicht umgesetzt.

## **4.2 Wirtschaftlichkeit**

Da keine Hardware oder Ähnliches involviert ist, hat sich an der Wirtschaftlichkeit des Projektes kaum etwas verändert. Einige Features konnten noch nicht implementiert werden, allerdings konnte ich aus gesundheitlichen sowie privaten Gründen weniger als erwartet am Projekt arbeiten. Die restlichen Features sollten in den noch übrigen 42 geplanten Stunden implementierbar sein.



## Kapitel 5

# Schlussbetrachtung

### 5.1 Planung

In dieser Projektarbeit habe ich gelernt, dass die Planung das absolut Wichtigste für ein Projekt ist.

Die Initialisierungsphase dauerte ein wenig länger als geplant, dies ist vor allem auf die Erstellung des Projektplans zurückzuführen. So habe ich den Plan zuerst mit der Projektmanagement Software «GanttProject»<sup>1</sup> versucht umzusetzen. Mit der Software bin ich jedoch schnell an ihre Grenzen gestossen und konnte nur Stundenweise und nicht Wochenweise Planen, was für eine Projektarbeit die in der Freizeit neben der Arbeit und Schule getätigt wird eher ungünstig ist.

Glücklicherweise haben wir im Projektmanagement Unterricht bereits einen Projektplan für eine in der Schule durchgeführte Fallstudie erstellt. Diesen konnte ich kopieren und entsprechend diesem Projekt anpassen.

Während der Konzeptphase musste ich mit gesundheitlichen und privaten Problemen kämpfen. Das Konzept konnte innerhalb der geplanten Zeit jedoch trotzdem durchgeführt werden und benötigte weniger Stunden als angenommen.

Es wurde viel Zeit für die Umsetzung geplant, jedoch konnte diese Zeit gar nicht voll und ganz in die vorhandenen Wochen investiert werden. So sind für die Umsetzung noch rund 20% der geplanten Zeit übrig.

Dies hat sich vor allem in den letzten zwei Wochen stark bemerkbar gemacht und so konnten einige kleinere Muss-Kriterien nicht umgesetzt werden.

### 5.2 Ziele und Anforderungen

Bei der Zielsetzung und Erstellung des Anforderungskatalogs habe ich gelernt, dass in der Phase Initialisierung die Ziele noch ein bisschen weniger detailreich beschrieben werden können.

Einige Details wurden bei der Konkretisierung in der Konzeptphase anders als in der Initialisierung definiert, lösen aber grundsätzlich die selben Bedürfnisse.

### 5.3 Realisierung

Die Phase Realisierung war eine intensive Zeit, so musste einige Zeit aufgeholt werden und mit Problemen mit fehlendem Know-How für das Phoenix Framework gekämpft werden.

Das Projekt hat mir gezeigt, dass selbst wenn man ein Framework bereits für einige andere Projekte eingesetzt hat, trotzdem noch grössere unbekannte Bereiche auftauchen können.

---

<sup>1</sup><https://www.ganttproject.biz/>

# PROJEKTINITIALISIERUNGSAUFTAG

## **WEBBASIERTER KONZERTKALENDER**

**Auftraggeber:** Damian Senn

**Projektleiter:** Damian Senn

**Autor:** Damian Senn

<b>1</b>	Ausgangslage	2
<b>2</b>	Ziele	3
<b>3</b>	Rahmenbedingungen	3
<b>4</b>	Ergebnisse und Termine	3
<b>5</b>	Aufwand	3
<b>6</b>	Kosten	4
<b>7</b>	Ressourcen	4
<b>8</b>	Kommunikation	4
<b>9</b>	Risiken	4





## AUSGANGSLAGE

Als regelmässiger Konzertbesucher wünsche ich mir eine Plattform im Internet, auf welcher ich eine zuverlässige Übersicht an Konzerten in meiner Umgebung vorfinde. Heute sind die Events nur verteilt auf verschiedenen Seiten wie die der Venues, des Konzertveranstalters, des Künstlers oder auf Facebook publiziert.

Ich möchte deshalb eine zentrale Plattform entwickeln, die es Benutzern einfach macht, Konzerte für ihren Geschmack zu finden.

Die Plattform soll Genre unabhängig sein und entsprechende Filter anbieten.

Um einen zusätzlichen Service für den User zur Verfügung zu stellen, ist es auch denkbar, eine Art Notifikationssystem zu bauen um Benutzer über Handy-Notifications oder per Email an Konzerte oder Künstler zu erinnern.

Konzertveranstaltern kann das Erfassen ihrer Events vereinfacht werden, indem auf der Plattform erfasste Veranstaltungen direkt auf den Sozialen Medien wie Facebook, Twitter oder Instagram geteilt werden können.

## ZIELE

- Definition der funktionalen Anforderungen
- Definition der nicht funktionalen Anforderungen
- Definition Projektumfang
- Projektplanung
- Aufwandschätzung
- Technologie Evaluierungen
- Lösungsvarianten

## RAHMENBEDINGUNGEN

- Das Projekt wird im Rahmen der Diplomarbeit durchgeführt
- Richtlinien zum Erstellen des Diplomberichtes
- Anwendung von HERMES, angepasst auf das Projekt

## ERGEBNISSE UND TERMINE

- Studie
- Projektauftrag
- Projektplan
- Evaluation
- Festgelegter Scope

## AUFWAND

Der Aufwand der Diplomarbeit wird auf ca. 300 Stunden geschätzt. Für die Initialisierungsphase wird mit ca. einer Woche gerechnet.

Initialisierung: 42h

## KOSTEN

Die Kosten werden mit einem durchschnittlichen Stundensatz von CHF 150.– gerechnet:

Initialisierung: CHF 6300.–

## RESSOURCEN

### **Personal**

Damian Senn (ca. 300 Stunden)

Da das Projekt durch Damian Senn alleine durchgeführt wird, ist keine Ressourcen aufteilung nötig.

### **Sachmittel**

Es werden keine Sachmittel wie Räume, IT-Infrastruktur, Spezifische Software, etc. benötigt die externe Kosten verursachen.

## KOMMUNIKATION

Da das Projekt von Damian Senn alleine durchgeführt wird, gibt es keine zu definierende Kommunikationswege.

## RISIKEN

Es sind keine Risiken für die Initialisierungsphase bekannt.

## Anhang B

# Sitzungsprotokoll — Kickoff

Hallo Sandro, hallo Severin

Folgende Themen/ Aktionen haben wir am Kickoff Meeting besprochen:

- Der Diplombericht ist zwingend eine Woche vor dem Abschlussgespräch ausgedruckt abzugeben.
- Ich frage Marc Aeby betreffend der Phasenfreigaben, wie das funktioniert wenn ich Auftragsgeber sowie Projektleiter bin.
- Für die Wirtschaftlichkeit werde ich aufzeigen was das Projekt gekostet hat und zeige mit einer Break-Even Analyse auf, ab wann das Projekt potentiell Gewinn machen könnte.
- Ich werde mir noch einmal den Bewertungsbogen genau durchlesen
- Die Abgrenzungen werde ich noch einmal Anpassen und das Monitoring sowie Deployment aus dem Projekt abgrenzen.
- Die Meilensteine werden noch von mir definiert.
- Sandro hat in KW15 und KW16 viel los und ist anfangs Mai im Ausland.

Bei Fragen oder Anmerkungen stehe ich euch gerne zur Verfügung.

Gruss  
Damian



## Anhang C

# Terminplan

Projektplan: Konzertkalender

Aktivität	Dauer [h]			Status	Wer																												
						Februar				März			April						Mai				Juni										
	Soll	Ist	Abw.			06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27						
Initialisierung	60	0	-60																														
1.1 Projektinitialisierung erstellen	4		-4	erledigt	DS																												
1.2 Projektorganisation	2		-2	erledigt	DS																												
1.3 Projektziele und Abgrenzungen	4		-4	erledigt	DS																												
1.4 Vorbereitung Kick Off & Meeting	8		-8	erledigt	DS,SB,SR																												
1.5 Projektplan	12		-12	erledigt	DS																												
1.6 Anforderungskatalog	4		-4	erledigt	DS																												
1.7 Risikoanalyse	4		-4	geplant	DS																												
1.8 Varianten beschreiben	8		-8	geplant	DS																												
1.9 Varianten evaluieren & auswählen	2		-2	geplant	DS																												
1.10 Projektauftrag erstellen	12		-12	geplant	DS																												
Konzept	66	0	-66																														
3.1 Portalnamen finden	2		-2	geplant	DS																												
3.2 Screens definieren	8		-8	geplant	DS																												
3.3 Screens designen	24		-24	geplant	DS,JS																												
3.4 Software Architektur	12		-12	geplant	DS																												
3.5 Test Konzept	12		-12	geplant	DS																												
3.6 Zwischen-Meeting	8		-8	geplant	DS,SB,SR																												
Realisierung	136	0	-136																														
4.1 Screens in HTML/CSS umsetzen	24		-24	geplant	DS																												
4.2 Initialisierung Backend	8		-8	geplant	DS																												
4.3 Implementation Registrierung/Login	8		-8	geplant	DS																												
4.4 Implementation Passwort Reset	8		-8	geplant	DS																												
4.5 Implementation der Screens	24		-24	geplant	DS																												
4.6 Implementation Suche	16		-16	geplant	DS																												
4.7 Tests erstellen	48		-48	geplant	DS																												
Abschluss	36	0	-36																														
5.1 Management Summary	4		-4	geplant	DS																												
5.2 Bericht ausdrucken, binden & senden	8		-8	geplant	DS																												
5.3 Diplomarbeit bewerten	16		-16	geplant	SB,SR																												
5.4 Abschluss Meeting	8		-8	geplant	DS,SB,SR																												
Total / bereits benötigt / Restliche Stunden:	298	0	-286																														

Total Soll:	298
Bereits benötigt:	0
Restliche Stunden:	286

### Legende

Name	Abk.
Damian Senn	DS
Sandro Bertolino	SB
Severin Rätz	SR
Joshua Schär	JS

planung-empty.ods

Damian Senn





## Anhang D

# Studie

### D.1 Zweck des Dokuments

In der Studie werden die Anforderungen aufgenommen, sowie Variantenbeschriebe für die Projektrealisierung erstellt. Die Varianten werden miteinander verglichen und durch den Variantenentscheid wird das weitere Vorgehen definiert. Ausserdem werden in der Studie die Risiken und Wirtschaftlichkeit des Projekts analysiert.

Folgende Arbeiten werden in dieser Studie abgehandelt:

- der Anforderungskatalog wird definiert
- die Evaluation der Browser Software-Technologien
- die Evaluation der Server Software-Technologien
- die Evaluation der Testing Software-Technologien
- eine Kostenschätzung und mögliche Wirtschaftlichkeit ausgerechnet

### D.2 Informationsbeschaffung

Folgende Quellen werden in diesem Projekt für die Informationsbeschaffung genutzt:

Quelle	Beschreibung
Schulwissen / Berufserfahrung	Die Grundlage für die Umsetzung dieses Projekts wird durch mein existierendes Schulwissen sowie meine langjährige Berufserfahrung in der Software-Entwicklung gesetzt.
Internet	Ein Grossteil der Informationen werden heute über das Internet bezogen, für die Evaluation von Technologien und Lösungsansätzen wird einiges über das Internet recherchiert werden müssen.
Externer Experte	Bei konzeptionellen sowie technischen Fragen kann der externe Experte um Rat gefragt werden.

TABELLE D.1: Informationsbeschaffung

### D.3 Anforderungskatalog

Im Anforderungskatalog werden die Muss- und Kann-Kriterien definiert. Muss-Kriterien sind zwingend zu erfüllen, Kann-Kriterien sind als optionale Erweiterung zu verstehen.

Feature	Titel	Nr.	Kriterium	Ziel	Muss
Suche	Suche nach Konzertname	1.1	Listet alle Konzerte die Wörter der Suche im Konzertnamen beinhalten	1.1	<b>Muss</b>
	Suche nach Konzertlocation	1.2	Schränkt die Such-Resultate nach gegebener Konzertlocation ein	1.2	<b>Muss</b>
	Suche nach Ort	1.2	Schränkt die Such-Resultate nach gegebenem Ort ein	1.2	<b>Muss</b>
	Suche nach Genre	1.2	Schränkt die Such-Resultate nach gegebenem Musik-Genre ein	1.2	<b>Muss</b>
Design	Desktop	2.1	Alle Ansichten haben eine Desktop-Optimierte Variante	1.4	<b>Muss</b>
	Tablet	2.2	Alle Ansichten haben eine Tablet-Optimierte Variante	1.4	<b>Muss</b>
	Mobile	2.3	Alle Ansichten haben eine Mobile-Optimierte Variante	1.4	<b>Muss</b>
	Browser Kompatibilität	2.4	Alle Ansichten müssen in aktuellem Google Chrome und Mozilla Firefox dem Grundlayout folgen	1.9	<b>Muss</b>
SEO	Indexierbarkeit	3.1	Das Produkt ist von Suchmaschinen indexierbar	1.5	<b>Muss</b>
	Linked Data	3.2	Konzert Detailseiten sind mit dem Event-Schema <sup>1</sup> ausgestattet	1.5	<b>Muss</b>
Benutzer	Registrierung	4.1	Besucher können sich einen Benutzer registrieren, Benutzernamen und E-Mail Adressen müssen einzigartig sein	1.6	<b>Muss</b>
	Passwort-Vergessen	4.2	Benutzer können sich einen Passwort-Reset Link anfordern	1.7	<b>Muss</b>
	Social	4.3	Benutzer können auf Konzerten vermerken ob sie teilnehmen oder nicht	1.11	Kann

<sup>1</sup><https://schema.org/MusicEvent>

Feature	Titel	Nr.	Kriterium	Ziel	Muss
Erfassung	Artist	5.1	Benutzer können Artisten mit einem Genre erfassen	1.8	<b>Muss</b>
	Location	5.2	Benutzer können eine Konzertlocation mit Ort/Strasse erfassen	1.8	<b>Muss</b>
	Konzert	5.3	Benutzer können ein Konzert mit Konzertlocation und Artisten erfassen	1.8	<b>Muss</b>
	Facebook	5.4	Benutzer können ein Konzert in ein Facebook-Event exportieren	1.10	Kann
Security	SQL-Injection	6.1	Das Produkt soll resistent gegen SQL-Injection sein	1.12	<b>Muss</b>
	HTML-Injection	6.2	Das Produkt soll resistent gegen HTML-Injection / XSS sein	1.12	<b>Muss</b>
	Passwort encryption	6.3	Passwörter von Benutzer müssen mit einem sicheren Verfahren gespeichert werden	1.12	<b>Muss</b>
	Session	6.4	Session-Cookies dürfen nicht durch JavaScript ausgelesen werden	1.12	Kann
Performance	Ladezeit	7.1	Die Seitenansichten dürfen nicht länger als 6 Sekunden auf einem 3G Netz laden		<b>Muss</b>
Sonstiges	User Tracking	8.1	Benutzerverhalten soll analysiert und nachvollziehbar sein.		Kann

TABELLE D.2: Anforderungskatalog

## D.4 Evaluation Browser-Technologie

### Gewichtung:

5 = Unverzichtbar, 4 = Sehr wichtig, 3 = Erleichtert die Arbeit, 2 = Weniger wichtig, 1 = unwichtig

Kriterium	Gewicht	Abnahmekriterium
Komplexität	3	Die Technologie sollte im Rahmen der Diplomarbeit nicht eine zu hohe Komplexität vorweisen. Durch eine niedrigere Komplexität bestehen weniger Risiken dass technische Probleme auftreten werden.
Performance	4	In den Projektzielen wurde definiert, dass die Applikation in maximal 6 Sekunden im Browser geladen sein muss. Daher ist es wichtig, dass die Technologie gute Performance Charakteristiken vorweist.
SEO	5	Für eine öffentliche Applikation ist es unentbehrlich, dass sie indexierbar durch Suchmaschinen ist.
Interaktivität	4	Applikationen im Browser werden immer interaktiver, daher ist es wichtig, dass die Technologie anspruchsvolle Abläufe implementieren kann.
Stabilität	3	Für das Projekt ist es wichtig, dass auf eine stabile Technologie gesetzt wird, welche den Projektablauf so wenig wie möglich beeinträchtigt.
Testing	3	Durch einfaches Testing, kann sichergestellt werden, dass die Applikation wie gewünscht umgesetzt wurde und auch beim Weiterentwickeln nicht existierende Funktionalitäten beeinträchtigt werden.

TABELLE D.3: Browser-Technologie Kriterien

### D.4.1 Variante: React

Die JavaScript Library **React** ist heute die wohl beliebteste Technologie um interaktive Applikationen im Web zu bauen.

React ermöglicht es den Entwicklern Screens auf eine deklarative Weise zu definieren, so dass sich Elemente jeweils dem Zustand der Applikation anpassen.

Die Features von React sind minimal gehalten und sehen auf den ersten Blick einfach aus, jedoch wird für Anwendungen schnell klar, dass zusätzliche Software-Libraries benötigt werden um eine grössere Applikation zu entwickeln.

Durch das Hinzufügen von weiteren Libraries steigt die Komplexität dieser Variante stark an, da es im React Ökosystem für viele Lösungen diverse verschiedene Lösungsansätze gibt.

### D.4.2 Variante: Next.js

**Next.js** ist ein JavaScript Framework, das auf der **React** Library aufbaut und zusätzliche Features sowie gängige Konventionen mitbringt.

Dadurch dass Next.js ein komplettes Framework ist und nicht nur eine Library, leidet diese Variante weniger der Komplexität eines kompletten Eigenbaus mit React. Features wie die Navigation von Seite zu Seite bringt Next.js bereits mit.

### D.4.3 Variante: SSR

**SSR** steht für **Serverside Rendering** und beschreibt die klassische Methode vom Erstellen von Webseiten, indem man HTML auf dem Server generiert und zum Browser schickt.

Dies hat nach wie vor seine Daseinsberechtigung, da dies wenig Komplexität mit sich bringt, einen schnelleren Seitenaufbau garantiert und ohne zusätzlichen Aufwand von Suchmaschinen indexiert werden kann.

## D.5 Bewertungen Browser-Technologie

### Bewertung:

4 = Sehr gut, 3 = Gut, 2 = Ungenügend, 1 = Schlecht

### Gewichtung:

5 = Unverzichtbar, 4 = Sehr wichtig, 3 = Erleichtert die Arbeit, 2 = Weniger wichtig, 1 = unwichtig

$$\text{Bewertung} \times \text{Gewichtung} = \text{Punktzahl}$$

Kriterium	Gewicht	Variante: React		Variante: Next.js		Variante: SSR	
		Wertung	P.	Wertung	P.	Wertung	P.
Komplexität	3	2	6	3	9	4	12
Performance	4	3	12	3	12	4	16
SEO	5	2	10	4	20	4	20
Interaktivität	4	4	16	4	16	3	12
Stabilität	3	2	6	3	9	4	12
Testing	4	4	16	4	16	4	16
<b>Total:</b>		React:	66	Next.js:	82	SSR:	88

TABELLE D.4: Browser-Technologie Bewertung

## D.6 Entscheid Browser-Technologie

Durch die Evaluierung wurde klar, dass das Einsetzen eines JavaScript-Frameworks zuviel zusätzliche Komplexität und gewisse Einbussungen in Performance und Stabilität unvermeidbar ist. Somit ist ein die Wahl für eine klassische SSR Webseite favorisierend.

Es ist durchaus vorstellbar, dass in einem zweiten Schritt, nach diesem Projekt, die SSR Applikation durch eine Next.js Applikation ersetzt werden könnte.

### Kosten / Wirtschaftlichkeit

Da die Programmiersprache Elixir sowie das Framework Phoenix unter einer Open Source Lizenz verfügbar sind, fallen bei dieser Lösung keine zusätzlichen Kosten an.

## D.7 Evaluation Server-Technologie

### Gewichtung:

5 = Unverzichtbar, 4 = Sehr wichtig, 3 = Erleichtert die Arbeit, 2 = Weniger wichtig, 1 = unwichtig

Kriterium	Gewicht	Abnahmekriterium
Komplexität	3	Die Technologie sollte im Rahmen der Diplomarbeit nicht eine zu hohe Komplexität vorweisen. Durch eine niedrigere Komplexität bestehen weniger Risiken dass technische Probleme auftreten werden.
Performance	4	In den Projektzielen wurde definiert, dass die Applikation in maximal 6 Sekunden im Browser geladen sein muss. Daher ist es wichtig, dass die Technologie gute Performance Charakteristiken vorweist.
Stabilität	5	Während es für die Browser-Technologie vorstellbar ist, die Technologie auszuwechseln, ist es für den Server wichtig auf eine stabile und zukunftssichere Technologie zu setzen.
Testing	5	Durch einfaches Testing, kann sichergestellt werden, dass die Applikation wie gewünscht umgesetzt wurde und auch beim Weiterentwickeln nicht existierende Funktionalitäten beeinträchtigt werden. Vorallem auf dem Server ist wichtig, dass die Businesslogik gut abdeckend getestet werden kann.

TABELLE D.5: Server-Technologie Kriterien

### D.7.1 Variante: Node.js / koa.js

Auch auf dem Server gewinnt JavaScript immer mehr an Beliebtheit. Mit Node.js und koa.js können schnell kleinere und simplere Applikationen erstellt werden, die dennoch sehr performant sind.

koa.js ist eine Library für Server-Applikationen und bietet nur eine einfache Basis und bringt keine Features für Datenpersistenz oder HTML Templates mit sich. Diese Features müssen durch zusätzliche Module installiert werden.

### D.7.2 Variante: Elixir / Phoenix

Elixir ist eine Programmiersprache die eine sehr stabile und performante Grundlage bietet. Durch das Framework Phoenix, wird im Elixir Ökosystem ein starkes feature umfangreiches Web-Framework angeboten. Phoenix beinhaltet eine grosse Menge an Features mit sich und gibt dem Entwickler direkt Funktionalitäten wie HTML Templates, Datenpersistenz und ein einfaches Test-Framework.

**D.7.3 Variante: Next.js**

Next.js wurde bereits als Variante für die Browser-Technologie in Betracht gezogen. Ein zusätzliches Feature von Next.js ist, dass die Applikation auch auf dem Server betrieben werden kann. Das Einsetzen der selben Technologie kann bedeutende Vorteile mit sich bringen, so muss man nur ein Framework lernen und kann Programmcode auf dem Server mit der Applikation im Browser geteilt werden.



## D.8 Bewertungen Server-Technologie

### Bewertung:

4 = Sehr gut, 3 = Gut, 2 = Ungenügend, 1 = Schlecht

### Gewichtung:

5 = Unverzichtbar, 4 = Sehr wichtig, 3 = Erleichtert die Arbeit, 2 = Weniger wichtig, 1 = unwichtig

$$\text{Bewertung} \times \text{Gewichtung} = \text{Punktzahl}$$

Kriterium	Gewicht	Variante: koa.js		Variante: Phoenix		Variante: Next.js	
		Wertung	P.	Wertung	P.	Wertung	P.
Komplexität	3	2	6	4	12	3	9
Performance	4	4	16	4	16	3	12
Stabilität	5	3	15	4	20	3	15
Testing	5	3	15	4	20	4	20
<b>Total:</b>		koa.js:	55	Phoenix:	68	Next.js:	56

TABELLE D.6: Server-Technologie Bewertung

## D.9 Entscheid Server-Technologie

Durch das grosse Featurset von Phoenix sowie tieferer Komplexität gegenüber den beiden anderen Varianten hat sich Phoenix für die Server-Technologie klar durchgesetzt.

### Kosten / Wirtschaftlichkeit

Da die Programmiersprache Elixir sowie das Framework Phoenix unter einer Open Source Lizenz verfügbar sind, fallen bei dieser Lösung keine zusätzlichen Kosten an.

## D.10 Evaluation Testing-Technologie

### Gewichtung:

5 = Unverzichtbar, 4 = Sehr wichtig, 3 = Erleichtert die Arbeit, 2 = Weniger wichtig, 1 = unwichtig

Kriterium	Gewicht	Abnahmekriterium
Performance	3	Bei wachsender Anzahl von Tests ist es wichtig, dass die Test-Software genug skalierbar ist um Tests in parallel auszuführen.
Stabilität	5	
Backend-Integration	4	Es ist sehr hilfreich, wenn die End-to-End Test-Software vom Server direkt ausgeführt werden. So kann gleichzeitig zum Browser-Test auch die Businesslogik getestet werden.
Visualtesting	5	Die Technologie soll mit dem Service percy.io integrierbar sein.

TABELLE D.7: Testing-Technologie Kriterien

### D.10.1 Jest + Puppeteer

Jest ist ein JavaScript-Test Framework von Facebook. Durch die Kombination der Puppeteer Library von Google ist es möglich, automatisierte Browser-Tests durchzuführen.

### D.10.2 Wallaby

Wallaby ist ein Elixir Browser-Test Framework, welches sich nahtlos mit Phoenix integrieren lässt. Wallaby unterstützt parallelisierung von Tests und ist daher ein guter Kandidat für eine hohe Anzahl von automatisierten Tests.

## D.11 Bewertungen Testing-Technologie

### Bewertung:

4 = Sehr gut, 3 = Gut, 2 = Ungenügend, 1 = Schlecht

### Gewichtung:

5 = Unverzichtbar, 4 = Sehr wichtig, 3 = Erleichtert die Arbeit, 2 = Weniger wichtig, 1 = unwichtig

$$\text{Bewertung} \times \text{Gewichtung} = \text{Punktzahl}$$

Kriterium	Gewichtung	Variante: Jest		Variante: Wallaby	
		Bewertung	Punkte	Bewertung	Punkte
Performance	3	4	12	4	12
Stabilität	5	3	15	4	20
Backend-Integration	4	2	8	4	16
Visualtesting	4	4	16	1	4
<b>Total:</b>		Jest:	51	Wallaby:	52

TABELLE D.8: Testing-Technologie Bewertung

## D.12 Entscheid Testing-Technologie

Dadurch dass sich Wallaby einfach mit der ausgewählten Server-Technologie verwenden lässt, hohe Performance und Stabilität aufweist, ist Wallaby die knapp bessere Variante als eine Jest + Puppeteer kombination.

Leider hat Wallaby keine Visualtesting Integration mit dem Dienst percy.io, dies könnte aber im verlaufe der Umsetzung eventuell im Rahmen dieser Arbeit umgesetzt werden.

### Kosten / Wirtschaftlichkeit

Auch Wallaby ist unter eines Open Source Lizenz veröffentlicht und erzeugt somit im Projekt keine direkten zusätzlichen Kosten. Durch fehlende Unterstützung von Visualtesting, wird im Verlauf der Realisierung zusätzlicher Aufwand erzeugt, entweder durch eigene Implementation solcher Tests oder durch unbemerkte Fehler die sich bei Änderungen eingeschlichen haben.

## D.13 Wirtschaftlichkeit

### D.13.1 Projektkosten

Für die Berechnung der Projektkosten wird ein Stundensatz von 150.- CHF angenommen.

Phase	Geplante Stunden	Kosten
Initialisierung	64	9'600.- CHF
Konzept	66	9'900.- CHF
Realisierung	136	20'400.- CHF
Abschluss	36	5'400.- CHF
<b>Total:</b>	302	42'900.- CHF

TABELLE D.9: Projektkosten

Die geplanten Projektkosten betragen somit **42'900.- CHF**.

Kostenstelle	Jährliche Kosten
Software	Keine
.com Domain	20.- CHF
Hosting	1'800.- CHF
<b>Total:</b>	1'820.- CHF

TABELLE D.10: Betriebskosten

Für die Betriebskosten eines Hostings wird einen durchschnittlichen monatlichen Preis von 150.- CHF angenommen, da das Deployment für dieses nicht vorgesehen ist, ist dies eine von Damian Senn geschätzte Zahl.

### D.13.2 Break Even Analyse

#### Gigboost

Beim Modell «Gigboost» wird Benutzern eine Option angeboten bei der ihre publizierten Gigs auf der Startseite sowie in Suchresultaten anderen Einträgen bevorzugt dargestellt werden. Für einen Gegenpreis von 10.- CHF kann ein Benutzer seinen Gig «boosten».

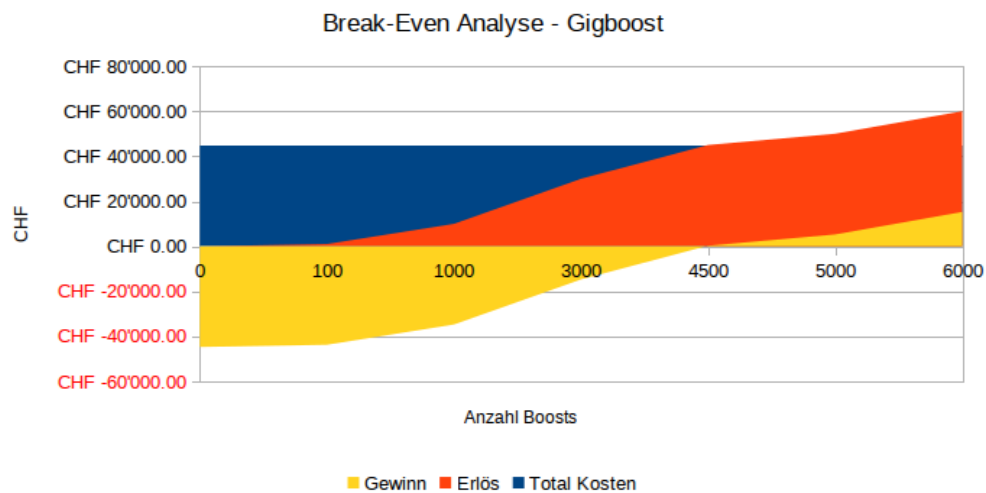


ABBILDUNG D.1: Break-Even Analyse - Gigboost

## Werbung

Im Modell «Werbung» wird ausgerechnet wieviele aktive Benutzer das Produkt benötigt um in den nächsten Jahren Gewinn zu erzielen.

Durch Annahme von einem Erlös von **140.- CHF** pro **40'000 Besucher<sup>2</sup>** erhalten wir folgendes Bild:

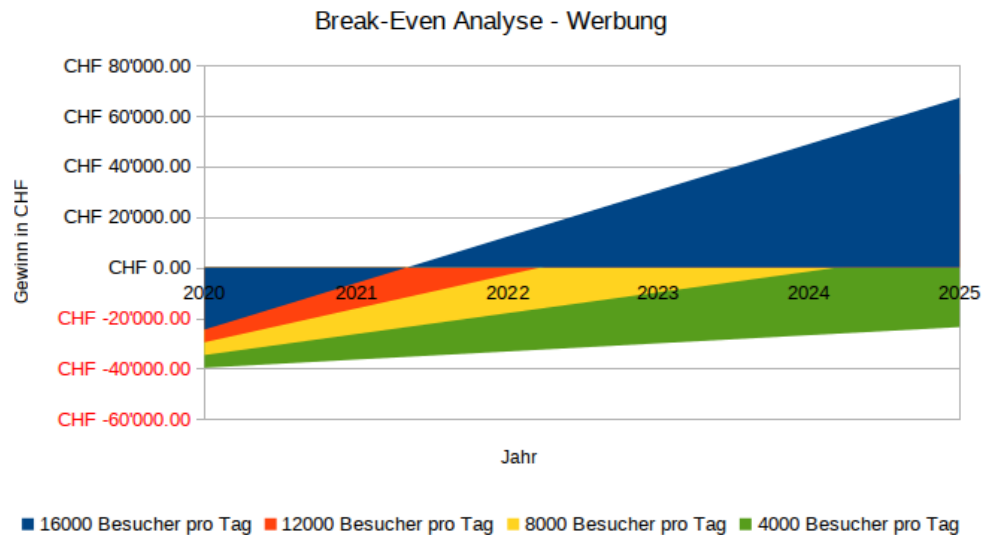


ABBILDUNG D.2: Break-Even Analyse - Werbung

Besucher pro Tag	Erlös pro Tag	Erlös pro Monat
2'000	7.- CHF	210.- CHF
4'000	14.- CHF	420.- CHF
8'000	28.- CHF	840.- CHF
12'000	42.- CHF	1'260.- CHF
16'000	46.- CHF	1'680.- CHF

TABELLE D.11: Werbeeinnahmen pro Besucher

Der Grafik ist zu entnehmen, dass das Produkt bei 8'000 Besucher pro Tag nach ca. 6 Jahren Gewinn erzielt. Bei 12'000 Besucher pro Tag erzielt das Produkt nach bereits 4 Jahren Gewinn und mit 16'000 Besucher pro Tag schon im dritten Jahr.

<sup>2</sup><https://www.quora.com/How-much-does-Google-AdSense-pay-for-3-banners-on-a-webpage-per-1-000-views/answer/Manas-Sahu-59>

## Anhang E

# Projektauftrag

### E.1 Zweck des Dokuments

Der Projektauftrag ist die verbindliche Vereinbarung zwischen Auftraggeber und Projektleiter, und bildet die Grundlage für den Projektstart sowie die Phasenfreigaben.

Folgend sind alle wichtigen Informationen die in der Phase Initialisierung erarbeitet wurden. Alle weiteren Details die in der Initialisierung ausgearbeitet wurden sind in der Studie im Anhang **D** zu finden.

### E.2 Ausgangslage

Als regelmässiger Konzertbesucher wünsche ich mir eine Plattform im Internet, auf welcher ich eine zuverlässige Übersicht an Konzerten in meiner Umgebung vorfinde. Heute sind die Events nur verteilt auf verschiedenen Seiten wie die der Venues, des Konzertveranstalters, des Künstlers oder auf Facebook publiziert.

Ich möchte deshalb eine zentrale Plattform entwickeln, die es Benutzern einfach macht, Konzerte für ihren Geschmack zu finden. Die Plattform soll Genre unabhängig sein und entsprechende Filter anbieten. Den Benutzern der Plattform soll es möglich sein, Konzerte selber zu erfassen und pflegen.

Um einen zusätzlichen Service für den Benutzer zur Verfügung zu stellen, ist es auch denkbar, eine Art Notifikationssystem zu bauen um Benutzer über Handy-Notifications oder per Email an Konzerte oder Künstler zu erinnern.

Konzertveranstaltern kann das Erfassen ihrer Events vereinfacht werden, indem auf der Plattform erfasste Veranstaltungen direkt auf den Sozialen Medien wie Facebook, Twitter oder Instagram geteilt werden können.

### E.3 Projektziele

Folgende Ziele sind in der Initialisierungsphase definiert worden:

Nr.	Zielbeschreibung	Muss/Kann
<b>Produktziele</b>		
1.1	Besucher können im Produkt nach Konzerten suchen	<b>Muss</b>
1.2	Suchresultate können nach Musik-Genre und Ort gefiltert werden	<b>Muss</b>
1.3	Besucher können Details zu einem Konzert ansehen	<b>Muss</b>
1.4	Das Produkt soll ein modernes responsives Design vorweisen	<b>Muss</b>
1.5	Konzerte sollen von Suchmaschinen indexiert werden können	<b>Muss</b>
1.6	Benutzer können sich im Produkt registrieren	<b>Muss</b>
1.7	Benutzer können ihr Passwort nach Verlust neu setzen	<b>Muss</b>
1.8	Inhalte des Portals sind durch die Benutzer erfassbar und bearbeitbar	<b>Muss</b>
1.9	Kompatibilität mit aktuellem Google Chrome und Mozilla Firefox Browser	<b>Muss</b>
1.10	Konzerte können vom Produkt nach Facebook exportiert werden	Kann
1.11	Ein angemeldeter Benutzer kann vermerken ob er einem Konzert teilnimmt	Kann
1.12	Das Produkt soll sich an die Security Best-Practices von OWASP halten	<b>Muss</b>
<b>Abwicklungsziele</b>		
2.1	Das Projekt soll nach HERMES 5 unter Berücksichtigung der Richtlinien von der TSBE dokumentiert werden	<b>Muss</b>
2.2	Das Produkt muss bis Projektende fertiggestellt, getestet und bereit für die Einführung sein	<b>Muss</b>
2.3	Die Technische-Umsetzung wird durch Damian Senn erstellt	<b>Muss</b>
2.4	Die Kommunikation zwischen Experten und Diplomanden erfolgt wie im Projektauftrag E.7.2 beschrieben.	<b>Muss</b>
2.5	Das Projekt muss bis Ende Mai 2019 abgeschlossen sein	<b>Muss</b>

TABELLE E.1: Ziele

### E.4 Rahmenbedingungen

- Das Projekt wird im Rahmen der Diplomarbeit durchgeführt.
- Die Richtlinien zum Erstellen des Diplomberichtes der TSBE. müssen eingehalten werden.
- Als Projektmethodik wird HERMES 5 verwendet, angepasst auf das Projekt.
- Sämtliche Projekt-Dokumente sowie Programmcode wird regelmässig ins private Github Repository<sup>1</sup> geladen.

<sup>1</sup><https://github.com/topaxi/diplomarbeit-tsbe>



## E.5 Terminplan

Nachfolgend ist der grobe Terminplan für die geplanten Phasen. Im Anhang C ist der detaillierte Terminplan abgelegt.

Phase	Datum	Stunden
Initialisierung	06.03.2019 - 31.03.2019	64
Konzept	01.04.2019 - 21.04.2019	66
Realisierung	22.04.2019 - 19.05.2019	136
Abschluss	20.05.2019 - 26.05.2019	36
Total:		302

TABELLE E.2: Terminplan

## E.6 Meilensteine

Im Projektplan wurden folgende Meilensteine und Termine festgelegt:

Nr.	Meilenstein	KW	Datum
1	Kickoff-Meeting	10	06.03.2019
2	Abschluss Phase Initialisierung	13	31.03.2019
3	Zwischen-Meeting	18	24.04.2019
4	Abschluss Phase Konzept	16	21.04.2019
5	Abschluss Phase Realisierung	20	19.05.2019
6	Abschluss Phase Abschluss	21	
7	Abschluss-Meeting	22	

TABELLE E.3: Meilensteine

Das Datum für das Abschluss-Meeting wird im Zwischen-Meeting mit den Experten, Sandro Bertolino und Severin Rätz, festgelegt. Der Abschluss der Phase Abschluss ist Abhängig vom Abschluss-Meeting und wird mindestens eine Woche vor dem Meeting stattfinden.

## E.7 Organigramm

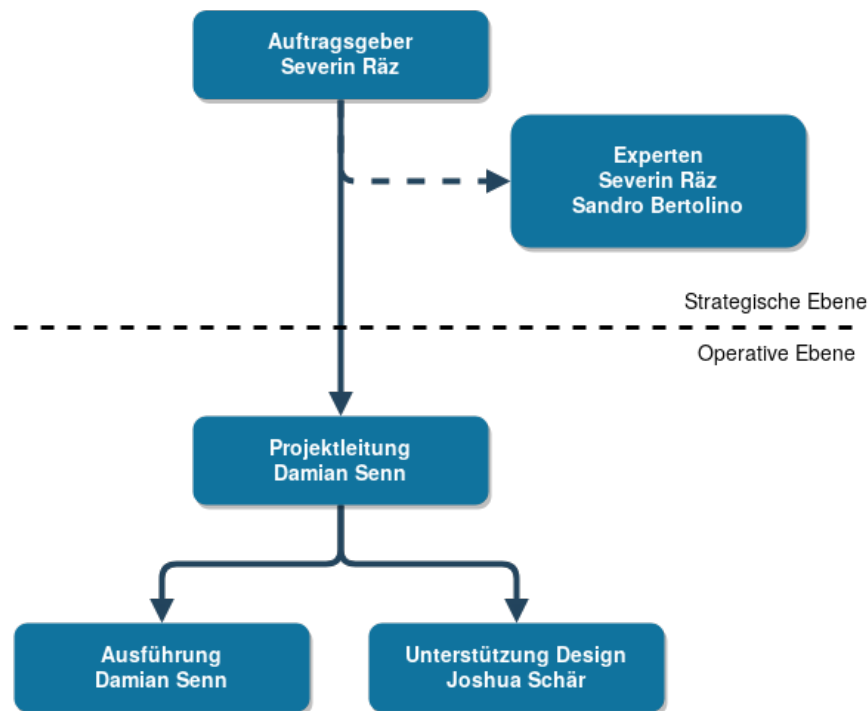


ABBILDUNG E.1: Organigramm

### E.7.1 Tätigkeiten im Projekt

Für die Freigaben der Phasen ist nach Absprache mit Severin Rätz Damian Senn selbstständig verantwortlich.

Name	Funktions- und Tätigkeitsbereich
Severin Rätz	Auftraggeber, externer Experte
Sandro Bertolino	Interner Experte
Damian Senn	Projektleiter, Ausführung
Joshua Schär	Unterstützung Design

TABELLE E.4: Tätigkeiten Verteilung

### E.7.2 Kommunikation

Wie im Kickoff-Meeting besprochen, wird Damian Senn alle zwei Wochen einen kurzen Bericht an Sandro Bertolino und Severin Rätz per E-Mail schicken. Im Bericht wird erläutert was in der Zwischenzeit erledigt wurde und was die nächsten Schritte im Projekt sind.

## E.8 Abgrenzungen

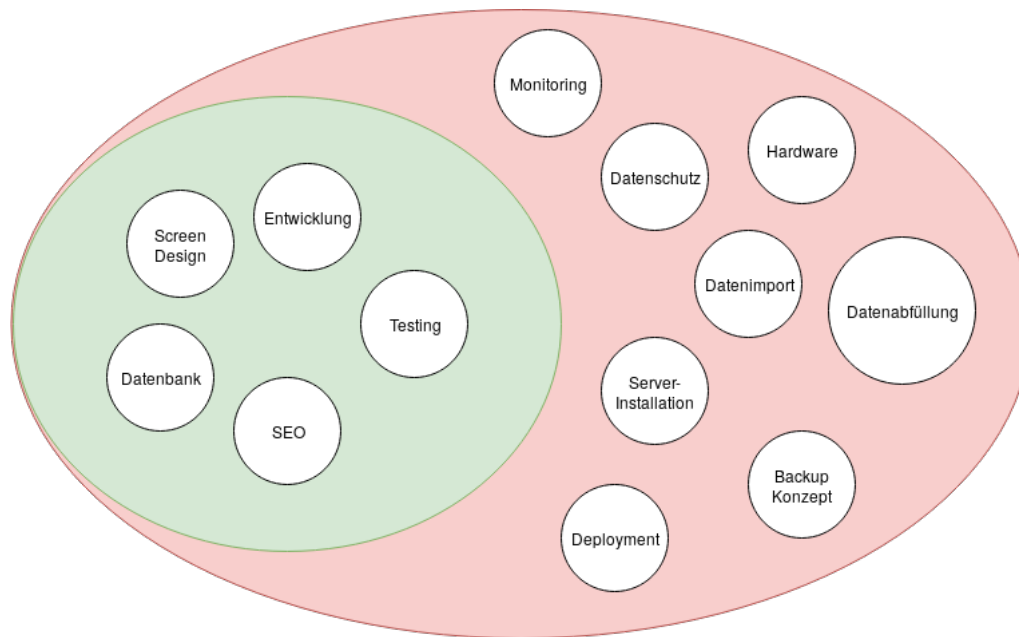


ABBILDUNG E.2: Abgrenzungen

### Hardware, Server-Installation, Deployment und Monitoring

Da das Projekt ein reines Software-Entwicklungs Projekt ist, werden keine Operativen tätigkeiten wie Hardwarebeschaffung, Server-Installation, Deployment und das einrichten eines Monitoring-Systems vorgenommen.

### Datenschutz

Da das Projekt nicht deployed wird und somit nicht produktiv /online gestellt wird, müssen im Rahmen dieser Projektarbeit noch keine Gedanken über den Datenschutz gemacht werden.

### Datenimport

Da wir bisher keine existierenden Konzertdaten besitzen, ist es nicht nötig, einen Datenimport zu implementieren.

### Datenabfüllung

Die Projektarbeit beinhaltet kein Datenset, Tests werden mit Testdaten abgewickelt. Es liegt nicht in der Verantwortung des Projektleiters, dass Daten in die Applikation abgefüllt werden.

### Backup Konzept

Es wird kein Backup Konzept benötigt, da die Applikation im Rahmen dieses Projektes nicht produktiv geschaltet wird.

## E.9 Anforderungskatalog

Der Anforderungskatalog wurde in der Studie erarbeitet. Es wurden Kann und Muss Kriterien definiert, wobei ein Muss-Kriterium zwingend erfüllt werden muss und ein Kann-Kriterium als Erweiterung angesehen wird.

Feature	Titel	Nr.	Kriterium	Ziel	Muss
Suche	Suche nach Konzertname	1.1	Listet alle Konzerte die Wörter der Suche im Konzertnamen beinhalten	1.1	<b>Muss</b>
	Suche nach Konzertlocation	1.2	Schränkt die Such-Resultate nach gegebener Konzertlocation ein	1.2	<b>Muss</b>
	Suche nach Ort	1.2	Schränkt die Such-Resultate nach gegebenem Ort ein	1.2	<b>Muss</b>
	Suche nach Genre	1.2	Schränkt die Such-Resultate nach gegebenem Musik-Genre ein	1.2	<b>Muss</b>
Design	Desktop	2.1	Alle Ansichten haben eine Desktop-Optimierte Variante	1.4	<b>Muss</b>
	Tablet	2.2	Alle Ansichten haben eine Tablet-Optimierte Variante	1.4	<b>Muss</b>
	Mobile	2.3	Alle Ansichten haben eine Mobile-Optimierte Variante	1.4	<b>Muss</b>
	Browser Kompatibilität	2.4	Alle Ansichten müssen in aktuellem Google Chrome und Mozilla Firefox dem Grundlayout folgen	1.9	<b>Muss</b>
SEO	Indexierbarkeit	3.1	Das Produkt ist von Suchmaschinen indexierbar	1.5	<b>Muss</b>
	Linked Data	3.2	Konzert Detailseiten sind mit dem Event-Schema <sup>2</sup> ausgestattet	1.5	<b>Muss</b>
Benutzer	Registrierung	4.1	Besucher können sich einen Benutzer registrieren, Benutzernamen und E-Mail Adressen müssen einzigartig sein	1.6	<b>Muss</b>
	Passwort-Vergessen	4.2	Benutzer können sich einen Passwort-Reset Link anfordern	1.7	<b>Muss</b>
	Social	4.3	Benutzer können auf Konzerten vermerken ob sie teilnehmen oder nicht	1.11	Kann

<sup>2</sup><https://schema.org/MusicEvent>

Feature	Titel	Nr.	Kriterium	Ziel	Muss
Erfassung	Artist	5.1	Benutzer können Artisten mit einem Genre erfassen	1.8	<b>Muss</b>
	Location	5.2	Benutzer können eine Konzertlocation mit Ort/Strasse erfassen	1.8	<b>Muss</b>
	Konzert	5.3	Benutzer können ein Konzert mit Konzertlocation und Artisten erfassen	1.8	<b>Muss</b>
	Facebook	5.4	Benutzer können ein Konzert in ein Facebook-Event exportieren	1.10	Kann
Security	SQL-Injection	6.1	Das Produkt soll resistent gegen SQL-Injection sein	1.12	<b>Muss</b>
	HTML-Injection	6.2	Das Produkt soll resistent gegen HTML-Injection / XSS sein	1.12	<b>Muss</b>
	Passwort encryption	6.3	Passwörter von Benutzer müssen mit einem sicheren Verfahren gespeichert werden	1.12	<b>Muss</b>
	Session	6.4	Session-Cookies dürfen nicht durch JavaScript ausgelesen werden	1.12	Kann
Performance	Ladezeit	7.1	Die Seitenansichten dürfen nicht länger als 6 Sekunden auf einem 3G Netz laden		<b>Muss</b>
Sonstiges	User Tracking	8.1	Benutzerverhalten soll analysiert und nachvollziehbar sein.		Kann

TABELLE E.5: Anforderungskatalog

## E.10 Lösungsbeschreibung

In der Studie (Anhang D) wurden Technologien gegenüber gestellt und für die Umsetzung mittels Nutzwertanalysen ausgewählt.

Folgende Technologien wurden ausgewählt:

**Browser sowie Server Technologie:**



ABBILDUNG E.3: Phoenix Framework Logo

Quelle: <https://github.com/phoenixframework/phoenix>

Die Nutzwertanalyse hat ergeben, dass es sinnvoller ist, das Projekt mit einer klassischen SSR Applikation zu starten. Das Phoenix Framework bietet alle benötigten Features an und kann durch zusätzliche Module einfach erweitert werden.

Für dynamische Interaktionen wie Formular-Validierungen wird zu einfachem JavaScript gegriffen. Ist ein Screen besonders interaktiv, kann gegebenenfalls eine kleinere JavaScript-Library verwendet werden um die Problemlösung zu vereinfachen.

**Testing Technologie:**



ABBILDUNG E.4: Wallaby Logo

Quelle: <https://github.com/keathley/wallaby>

Getestet wird die Applikation durch die von Phoenix gegebenen Testing-Tools sowie mit der Browser-Testing Library «Wallaby».

## E.11 Kosten

In der Studie wurden die Projekt- sowie Betriebskosten ausgerechnet.

Der gesamte Personalaufwand beträgt **42'900** für die geplanten Stunden.

Phase	Geplante Stunden	Kosten
Initialisierung	64	9'600.- CHF
Konzept	66	9'900.- CHF
Realisierung	136	20'400.- CHF
Abschluss	36	5'400.- CHF
<b>Total:</b>	302	42'900.- CHF

TABELLE E.6: Projektkosten

Für die Betriebskosten wurde angenommen, dass das Produkt in der Cloud auf einer mittelgrossen Umgebung betrieben wird. Die Kosten dieser Umgebung wurde auf 150.- CHF pro Monat geschätzt.

Neben der Umgebung muss mindestens eine Domain gekauft und jährlich bezahlt werden. Die Kosten einer Domain sind rund 20.- CHF pro Jahr.

Da jediglich Open Source Software eingesetzt wird, gibt es keine Software-Lizenzen zu bezahlen.

Kostenstelle	Jährliche Kosten
Software	Keine
.com Domain	20.- CHF
Hosting	1'800.- CHF
<b>Total:</b>	1'820.- CHF

TABELLE E.7: Betriebskosten

## E.12 Risiken

Die Risikobewertung erfolgt mit folgender Formel:

$$\text{Bewertung} = \text{Schaden} \times \text{Eintrittswahrscheinlichkeit}$$

Schadensskala:

Gewichtung	Beschreibung
Gering (1-2)	Kleiner Schaden, hat kaum Auswirkungen auf das Projekt.
Mittel (3-4)	Mittlerer Schaden, Zeitverzögerungen oder Qualitätsverluste.
Hoch (5-6)	Hoher Schaden, wichtige Arbeiten oder Phasen können nicht abgeschlossen werden, schlimmstenfalls ein Abbruch des Projekts.

TABELLE E.8: Risiken - Schadensskala

Eintrittswahrscheinlichkeitsskala:

Gewichtung	Beschreibung
Gering (1-2)	Kleine Eintrittswahrscheinlichkeit.
Mittel (3-4)	Mittlere Eintrittswahrscheinlichkeit.
Hoch (5-6)	Hohe Eintrittswahrscheinlichkeit.

TABELLE E.9: Risiken - Eintrittswahrscheinlichkeit

Handlungen um Risikobewertungen zu senken:

Handlung	Beschreibung
Akzeptanz	Das Eintreten eines Risiko wird wissentlich angenommen.
Transfer	Die Verantwortung von Risiken können an Dritte abgegeben werden.
Verminderung	Der Schaden oder die Eintrittswahrscheinlichkeit kann begrenzt oder reduziert werden.
Vermeidung	Es kann jeglichen Schaden vermieden werden.

TABELLE E.10: Risiken - Handlungen zur Senkung der Bewertung



**E.12.1 Projektrisiken**

<b>Nr.</b>	<b>Risiko</b>	<b>Auswirkung</b>	<b>Schaden</b>	<b>Wahrsch.</b>	<b>Bewertung</b>
1	Ausfall des Entwicklers oder Projektleiters	Verzögerungen von Arbeiten	4	3	Mittel
2	Unvollständige Projektdokumentation	Schlechtere Diplomarbeit Bewertung	4	2	Mittel
3	Schlechter Projektplan	Verzögerungen und eventuelle Qualitätsverluste	4	3	Mittel
4	Keine Benutzer	Das Produkt wird nicht von Benutzern eingesetzt	3	4	Mittel
5	Technisch nicht umsetzbare Features	Das Produkt kann nicht wie angedacht benutzt werden	4	3	Mittel

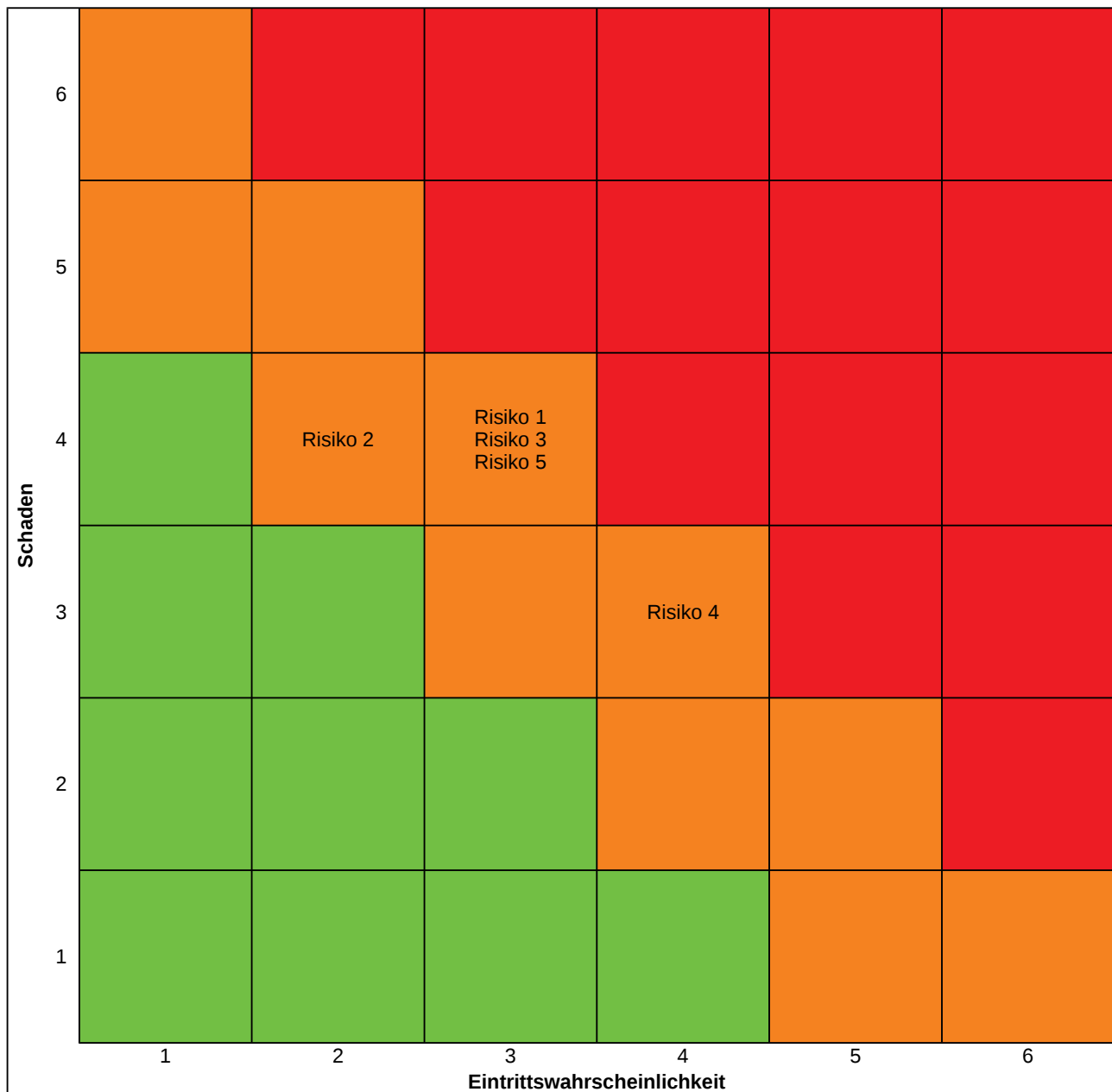
TABELLE E.11: Projektrisiken

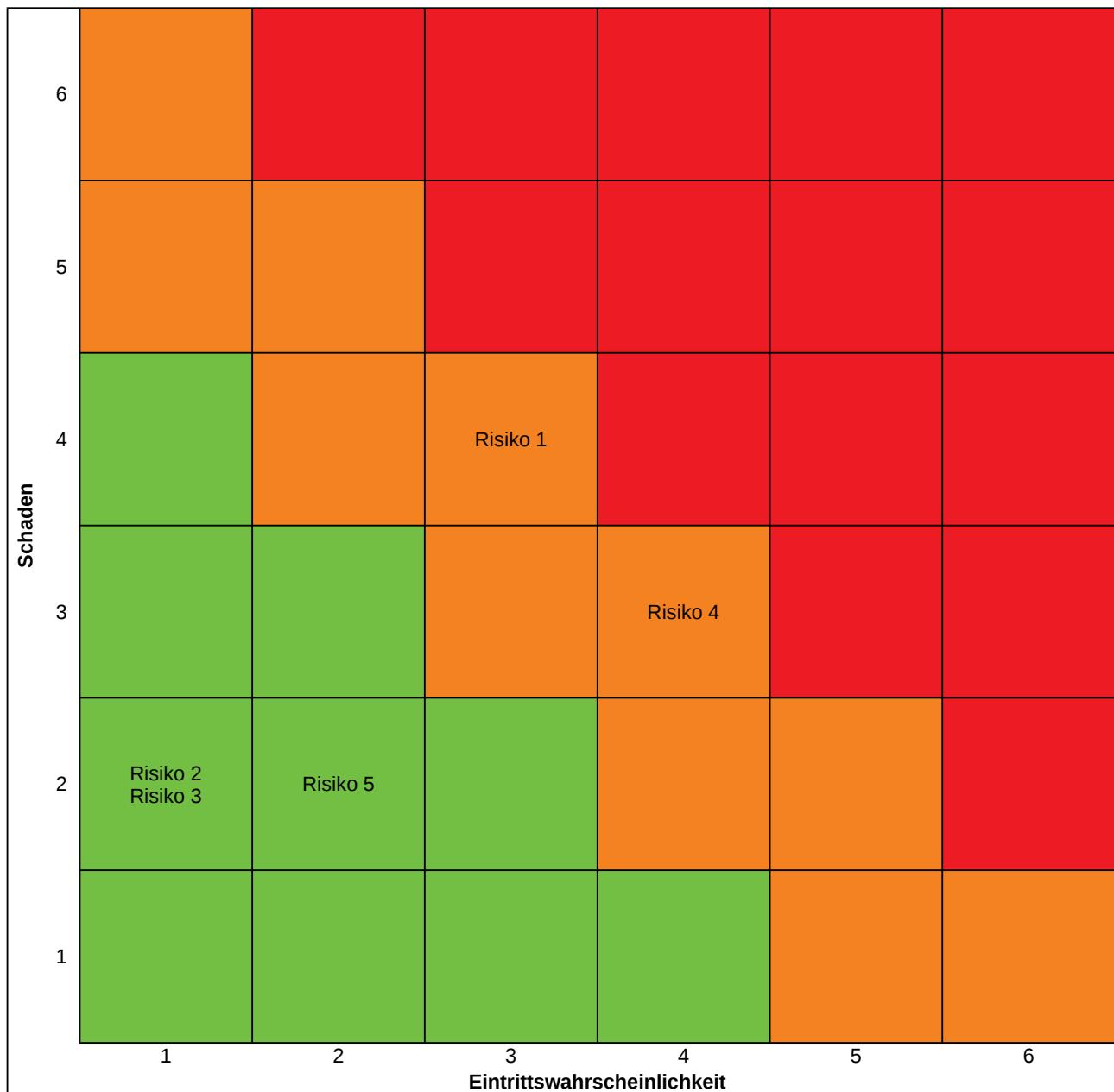
**E.12.2 Massnahmen**

Nr.	Massnahme	Handlung	Bewertung nach Massnahme		
			Schaden	Wahrsch.	Bewertung
1	Arzt aufsuchen, ggf. Projekt-Pause oder Abbruch	Akzeptanz	4	3	Mittel
2	Statusbericht alle zwei Wochen, bei Fragen sofort Hilfe suchen	Verminderung	2	1	Gering
3	Genügend Buffer-Zeit einplanen, ggf. Ferientage für Projekt einsetzen	Verminderung	2	1	Gering
4	Das Produkt löst vor allem ein persönliches Interesse	Akzeptanz	3	4	Mittel
5	Vereinfachte Alternativen in Konzept-Phase untersuchen	Verminderung	2	2	Gering

TABELLE E.12: Projektrisiken - Massnahmen

## E.12.3 Risikodiagramm ohne Massnahmen



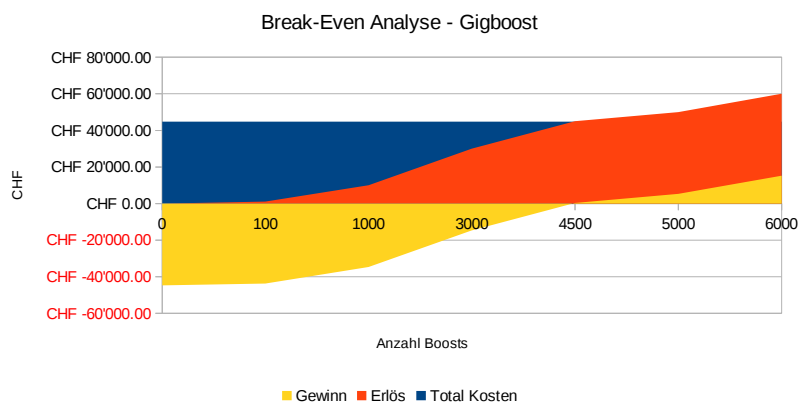
**E.12.4 Risikodiagramm mit Massnahmen**

## Anhang F

# Wirtschaftlichkeit - Gigboost

### Analyse Modell – Gigboost

Anzahl Boosts	0	100	1000	3000	4500	5000	6000
Projektkosten	CHF 42'900.00	CHF 42'900.00	CHF 42'900.00	CHF 42'900.00	CHF 42'900.00	CHF 42'900.00	CHF 42'900.00
Betriebskosten	CHF 1'820.00	CHF 1'820.00	CHF 1'820.00	CHF 1'820.00	CHF 1'820.00	CHF 1'820.00	CHF 1'820.00
Total Kosten	CHF 44'720.00	CHF 44'720.00	CHF 44'720.00	CHF 44'720.00	CHF 44'720.00	CHF 44'720.00	CHF 44'720.00
„Gigboost“	CHF 10.00	CHF 10.00	CHF 10.00	CHF 10.00	CHF 10.00	CHF 10.00	CHF 10.00
Erlös	CHF 0.00	CHF 1'000.00	CHF 10'000.00	CHF 30'000.00	CHF 45'000.00	CHF 50'000.00	CHF 60'000.00
Gewinn	CHF -44'720.00	CHF -43'720.00	CHF -34'720.00	CHF -14'720.00	CHF 280.00	CHF 5'280.00	CHF 15'280.00





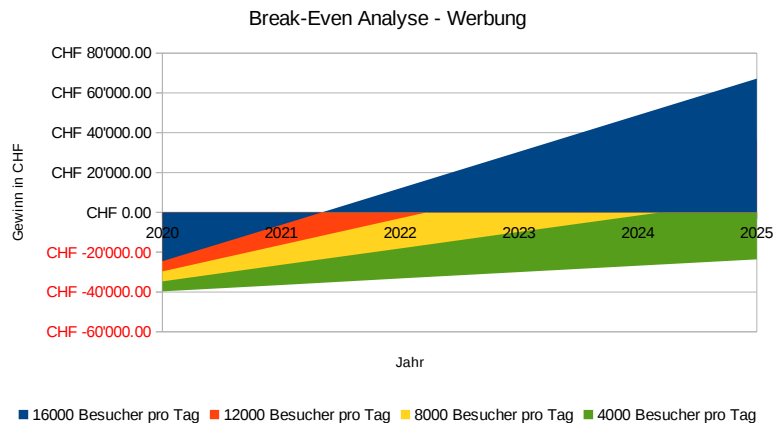
## Anhang G

# Wirtschaftlichkeit - Werbung

### Analyse Modell – Werbung

Besucher pro Tag	2000	4000	8000	12000	16000
Besucher pro Jahr	720000	1440000	2880000	4320000	5760000
Projektkosten	CHF 42'900.00	CHF 42'900.00	CHF 42'900.00	CHF 42'900.00	CHF 42'900.00
Betriebskosten	CHF 1'820.00	CHF 1'820.00	CHF 1'820.00	CHF 1'820.00	CHF 1'820.00
Total Kosten	CHF 44'720.00	CHF 44'720.00	CHF 44'720.00	CHF 44'720.00	CHF 44'720.00
CPC	CHF 5.00	CHF 10.00	CHF 20.00	CHF 30.00	CHF 40.00
CPM	CHF 2.00	CHF 4.00	CHF 8.00	CHF 12.00	CHF 16.00
Erlös pro Monat	CHF 210.00	CHF 420.00	CHF 840.00	CHF 1'260.00	CHF 1'680.00
Erlös pro Jahr	CHF 2'520.00	CHF 5'040.00	CHF 10'080.00	CHF 15'120.00	CHF 20'160.00
Gewinn	CHF -42'200.00	CHF -39'680.00	CHF -34'640.00	CHF -29'600.00	CHF -24'560.00

	4000 Besucher pro Tag	8000 Besucher pro Tag	12000 Besucher pro Tag	16000 Besucher pro Tag
2020	CHF -39'680.00	CHF -34'640.00	CHF -29'600.00	CHF -24'560.00
2021	CHF -36'460.00	CHF -26'380.00	CHF -16'300.00	CHF -6'220.00
2022	CHF -33'240.00	CHF -18'120.00	CHF -3'000.00	CHF 12'120.00
2023	CHF -30'020.00	CHF -9'860.00	CHF 10'300.00	CHF 30'460.00
2024	CHF -26'800.00	CHF -1'600.00	CHF 23'600.00	CHF 48'800.00
2025	CHF -23'580.00	CHF 6'660.00	CHF 36'900.00	CHF 67'140.00







## Anhang H

# Konzept

### H.1 Zweck des Dokuments

Das Konzept dient als Anleitung für die Realisierungsphase. Die in der Konzept erarbeiteten Details müssen in der Realisierung eingehalten und umgesetzt werden.

#### H.1.1 Teilkonzepte

Durch die in der Studie gewonnenen Erkenntnissen, werden in der Phase Konzept verschiedene Teilkonzepte erstellt.

Im Teilkonzept «Portalname» wird der Name des Produktes erarbeitet.

Im Teilkonzept «Design- und Bedienkonzept» werden die Ansichten der Applikation in Mockups umgesetzt. Es werden die Benutzer Use-Cases vom Besucher sowie der Konzert-Erfasser aufgezeigt.

Im Teilkonzept «Softwarekonzept» werden die Datenflüsse hinter den Mockups aufgezeigt, sowie die Datenbankstruktur aufgebaut.

Im Teilkonzept «Testkonzept» werden die einzelnen Systemtests aufgelistet sowie ausgearbeitet wie granular welche Teile der Software getestet werden sollen.

Im letzten Teil des Konzept-Dokuments wird im Fazit dokumentiert, wie und warum das Konzept von den vorhergehenden Phasen des Projekts abweicht.

## H.2 Portalname

Der Portalname wurde in einer Brainstorming-Session von Damian Senn auf den Namen «**Gigpillar**» festgelegt. Der Name ist angelehnt an die Werbepfeiler in Städten, wo oft Werbeplakate für Konzerte hängen.

Die folgenden Ideen wurden in Betracht gezogen, jedoch war keine Domain mehr verfügbar oder der Name überzeugte nicht:

- upto.com («What are you up to?»)
- up-to.com
- uptoin.com
- gigup.com
- gigsta.com («Gigs to attend»)
- gigin.com
- gigin.com
- gixin.com («Gigs in»)
- dualact.com («Loud act»)
- trecnoc.com («Concert» rückwärts)

## H.3 Design- und Bedienkonzept

### H.3.1 Mockups

#### Homepage

Die Homepage ist die erste Seite, die der Besucher sieht, wenn er/sie die Applikation direkt über [gigpillar.com](https://gigpillar.com) aufruft. Auf den ersten Blick ist die Suche sowie ein grosses Bild (Banner) eines Gigs zu erblicken. Weiter sind Links zu gängigen Funktionalitäten wie Gig hinzufügen sowie das Login in einer Navigation erreichbar.

Unter dem Banner werden Gigs in nächster Nähe des Besuchers aufgelistet, der Link «change location» führt weiter zur Suchresultate Seite um den entsprechenden Filter anzupassen.

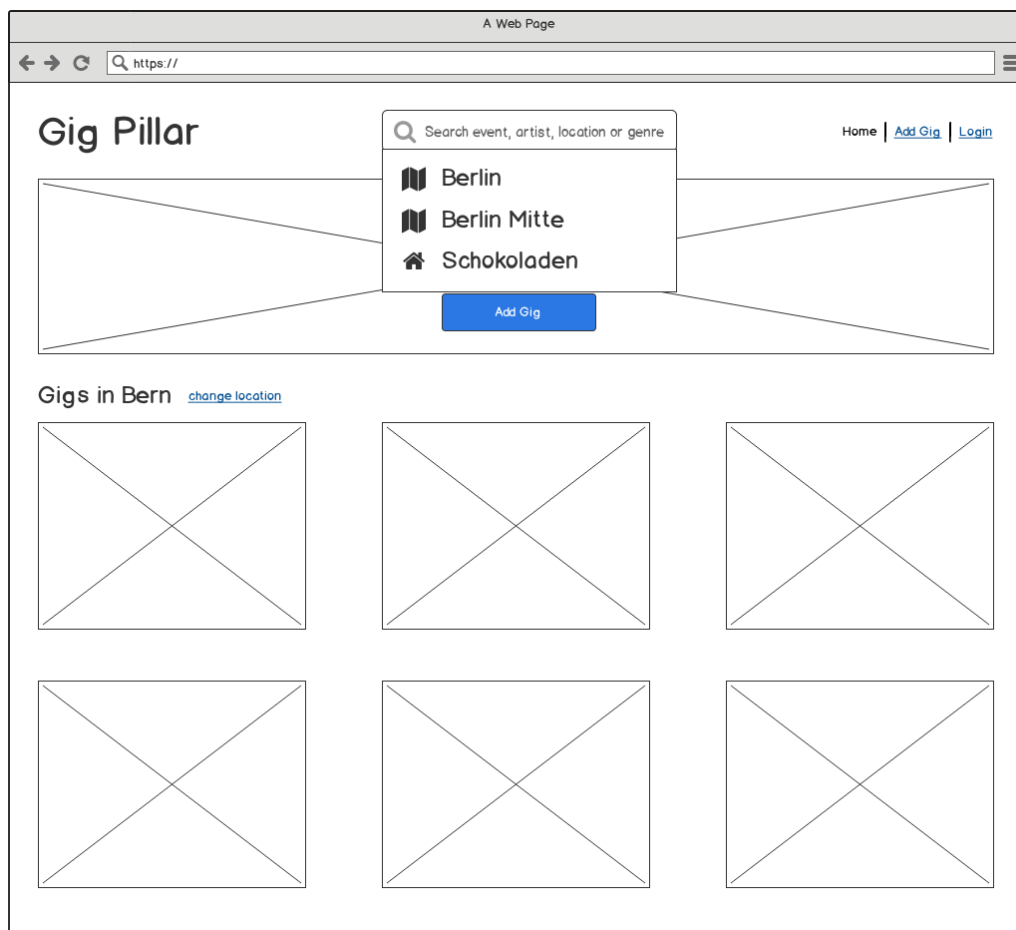


ABBILDUNG H.1: Mockup: Homepage

## Suchresultate

Auf der Suchresultate Seite sieht der Benutzer seine Suchresultate der von der globalen Suchbox ausgelösten Suche. Die Seite bietet weitere Filter an um die Resultate weiter einzugrenzen.

Folgende Filter stehen den Benutzern zur Verfügung:

- Ort
- Datum von
- Datum bis
- Musik Genre

Das Anwählen eines Suchresultates führt den Benutzer weiter zur detaillierten Gig Ansicht.

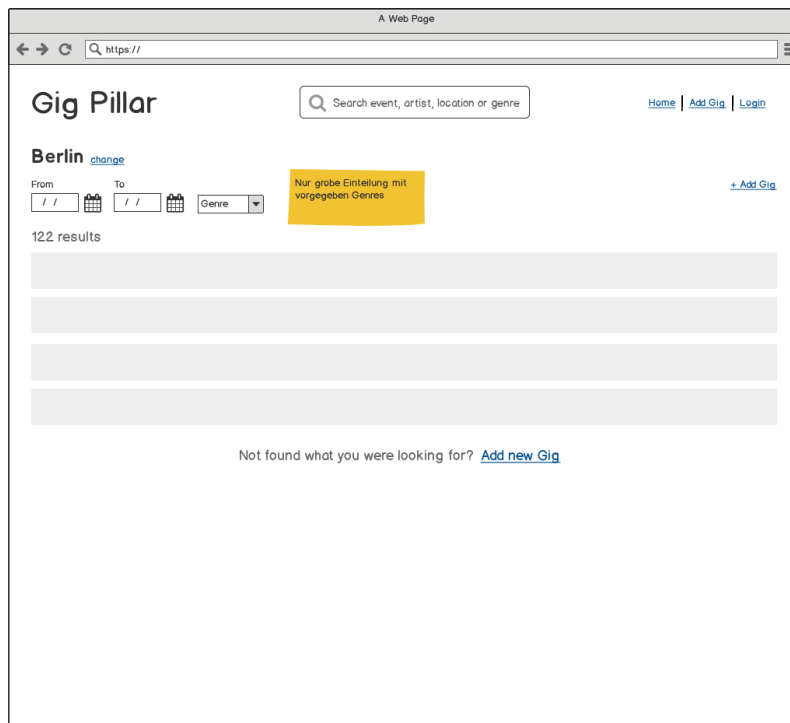


ABBILDUNG H.2: Mockup: Suchresultate

## Gig Ansicht

In der Gig Ansicht werden alle Details zu einem Event aufgelistet.

- Datum des Events
- Zeit wann das Event beginnt, bzw die Location die Türen öffnet
- Liste aller Künstler mit optionaler Startzeit
- Eine Beschreibung des Events
- Die Adresse der Location mit Link auf Google Maps

Ausserdem soll es den Benutzern möglich sein, über einen «Add to my calendar» Link das Event zu seiner Kalender-Applikation zu importieren.

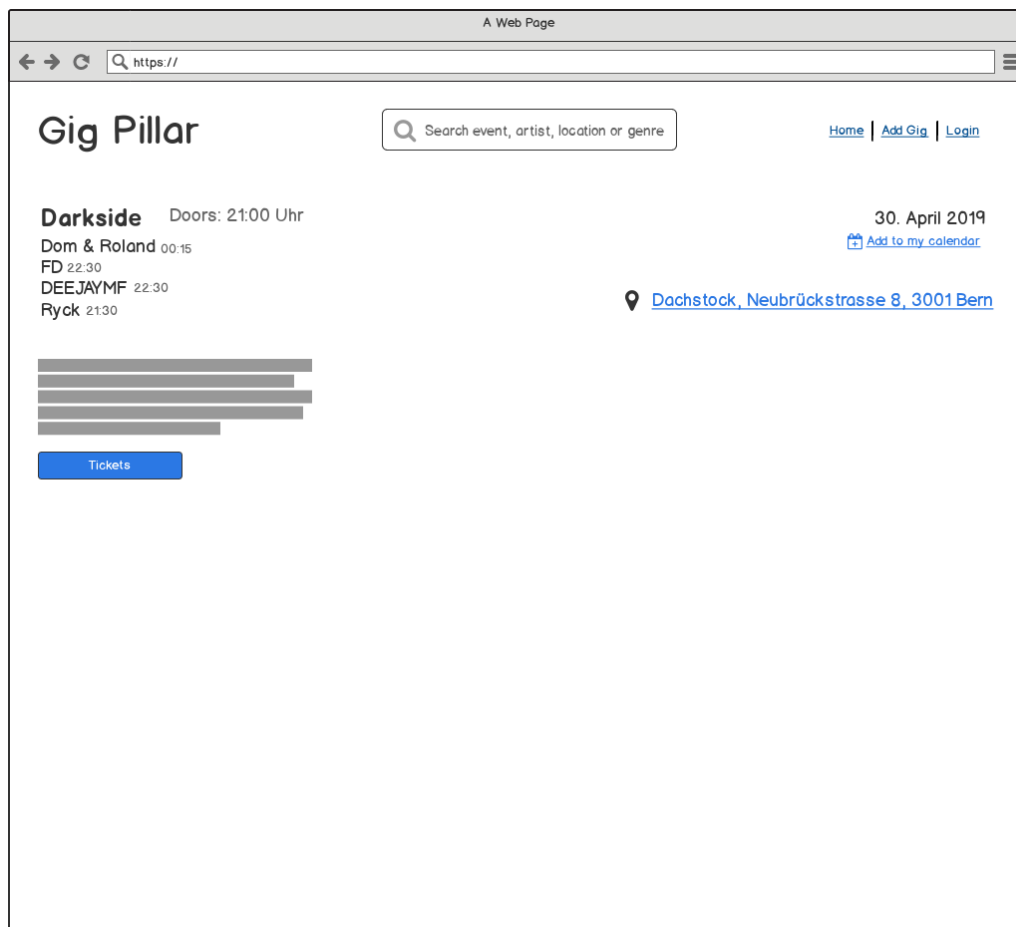


ABBILDUNG H.3: Mockup: Gig Ansicht

## Gig erfassen

Benutzer können Gigs erfassen.

Folgende Daten sind für einen Gig zu erfassen:

- Name
- Bild (*optional*)
- Location
- Datum
- Zeit
- Eine Liste von Artists mit optionaler Startzeit
- Beschreibung
- Link zum Ticketvertreiber

The mockup shows a web browser window titled 'A Web Page' with the URL 'https://'. The page header for 'Gig Pillar' includes a search bar 'Search event, artist, location or genre', links for 'Home' and 'Add Gig', and a user profile icon. The main section is titled 'Add new Gig' and contains the following form elements:

- Gig name:** A text input field.
- Image:** A circular placeholder with an 'Add image' button.
- Where:** A text input field with a search icon and the placeholder text 'search location'.
- When:** A date and time selector with a calendar icon. The time is set to 'Doors: 20.00'.
- Who:** A text input field with a search icon and the placeholder text 'Choose artist'. Below it, an autocomplete dropdown is open, showing 'Pennywise' as a suggestion. A green arrow points from the input field to the suggestion.
- Describe Gig:** A large text area for the gig description.
- Tickets:** A text input field with the placeholder text 'https://whosellsthesetickets.example.com'.
- Save Gig:** A blue button at the bottom of the form.

ABBILDUNG H.4: Mockup: Gig erfassen

## Benutzerprofil

Benutzer können ihr eigenes Profil verwalten und folgende Tätigkeiten verrichten:

- Anzeigenname ändern
- E-Mail Adresse ändern (*mit E-Mail Bestätigung*)
- Passwort ändern (*muss vorher altes Passwort bestätigen*)
- Account löschen (*muss doppelt bestätigt werden!*)

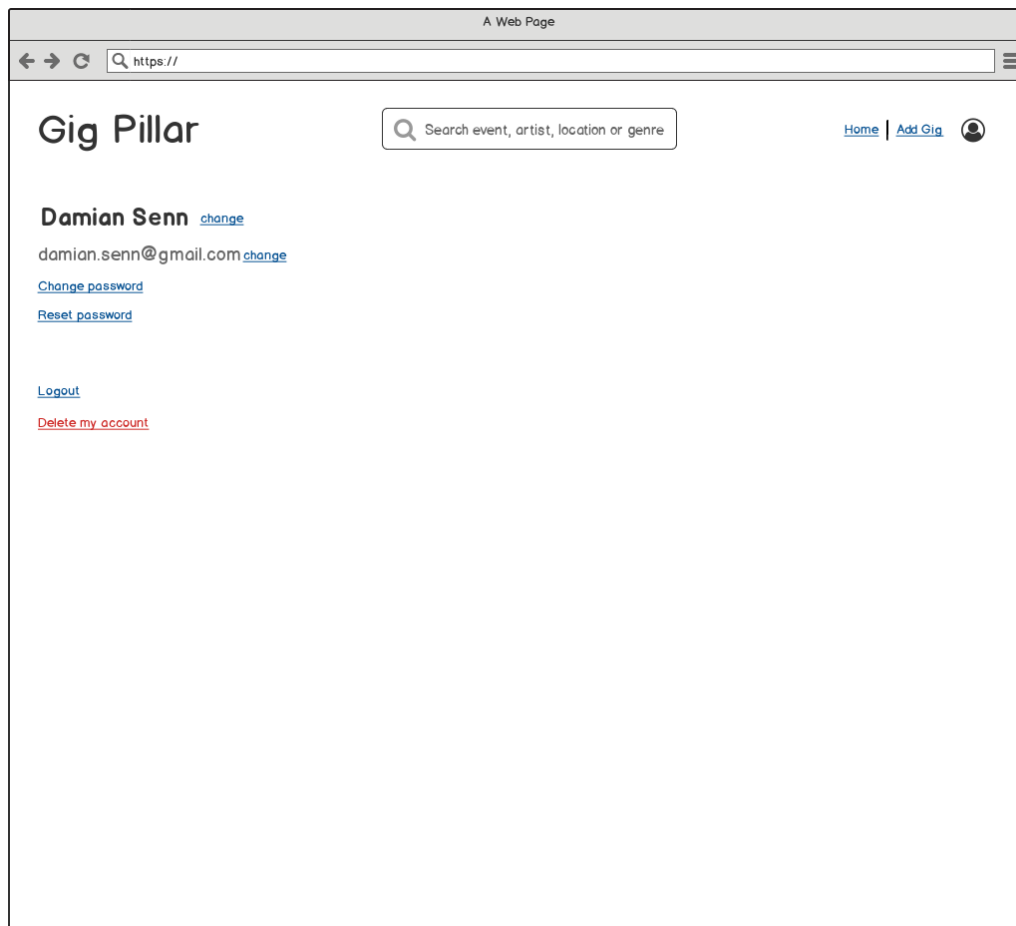


ABBILDUNG H.5: Mockup: Benutzerprofil

### **H.3.2 Genre Filter**

Der Genre Filter soll folgende Werte zur Verfügung stellen.

- Alternative
- Blues
- Classical
- EDM
- Hip-Hop
- Jazz
- Metal
- Pop
- Punk
- Reggae
- Rock



## H.4 Softwarekonzept

### H.4.1 Datenfluss

#### Homepage

Die Homepage zeigt den Besuchern Gigs in ihrer Nähe an, dazu muss über eine GeoIP API die IP-Adresse des Besuchers auf ein Land zurückverfolgt werden. Dazu wird beim ersten Besuch die GeoIP API abgefragt und das Land des Benutzers in eine Session geschrieben. Bei weiteren Aufrufen wird das Land direkt aus der Session bezogen.

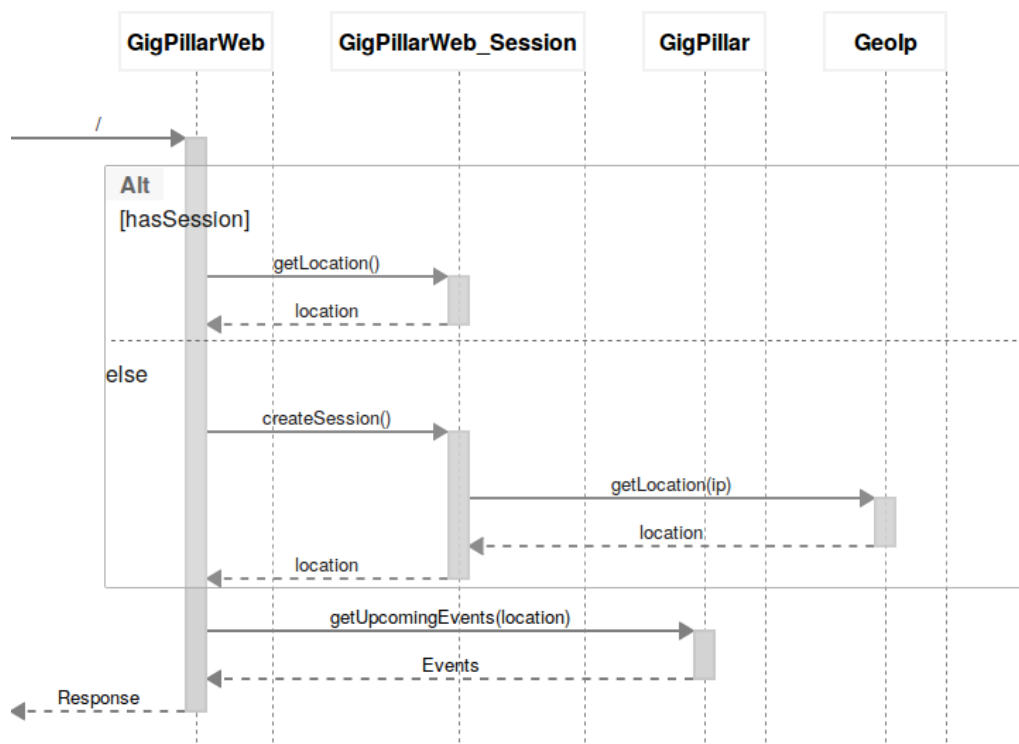


ABBILDUNG H.6: Datenfluss: Homepage

## Suchfeld

Das globale Suchfeld hat eine Autocompletion, welche Daten direkt von der GigPillar Applikation bezieht. Die Daten für Städtenamen wird jedoch von einer externen Datenquelle, z.B. Google Maps, bezogen.

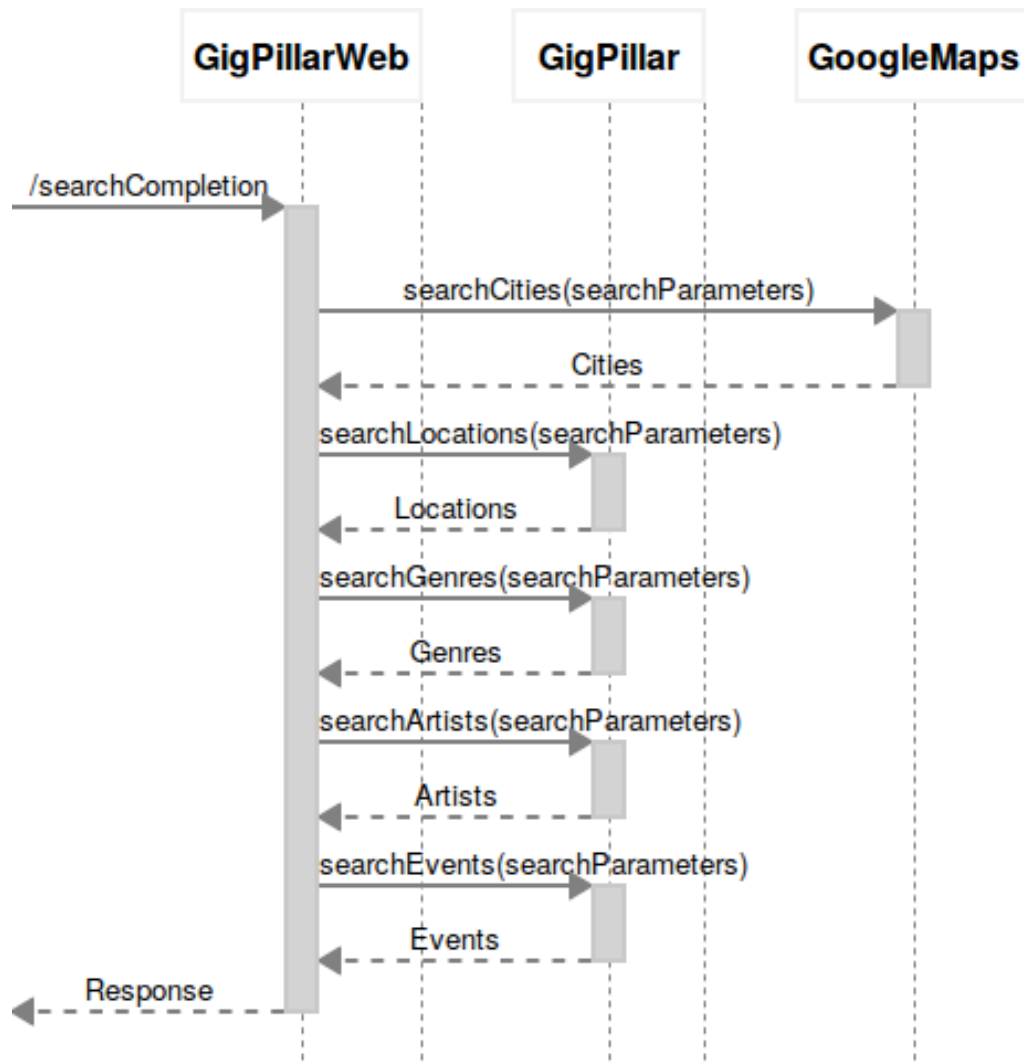


ABBILDUNG H.7: Datenfluss: Suchfeld

### Gig erstellen - Locationfeld

Beim Erstellen eines neuen Gigs, muss eine Location zugewiesen werden. Die Locations werden über die bereits in GigPillar erfassten Locations sowie über eine externe Datenquelle, wie z.B. Google Maps, bezogen.

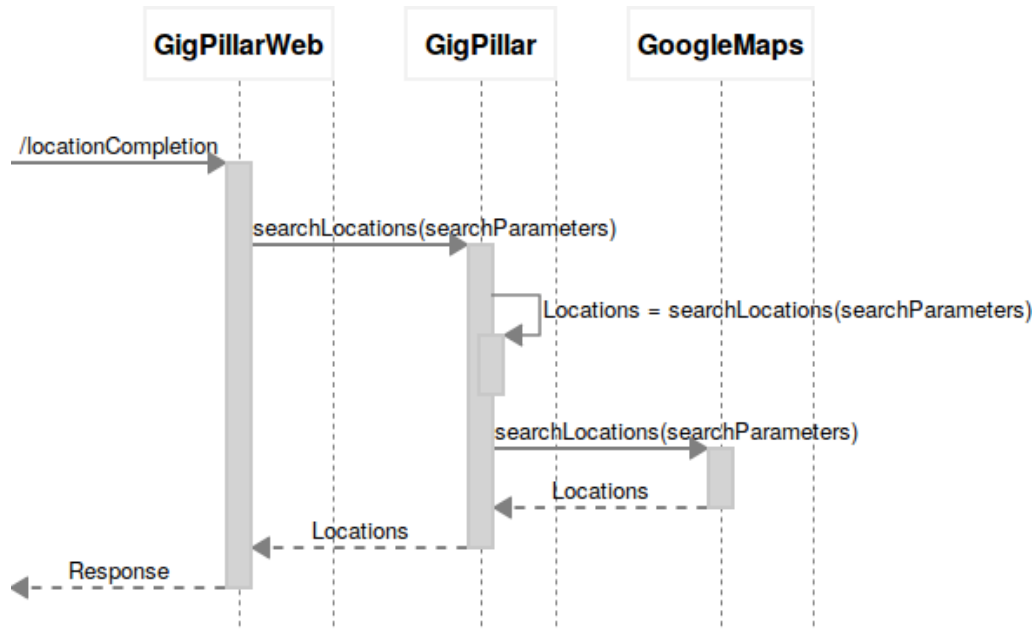


ABBILDUNG H.8: Datenfluss: Gig erstellen - Locationfeld

### Passwort-Reset

Falls ein Benutzer sein Passwort vergessen hat, kann dieser ein neues Passwort über die Passwort-Reset Funktion setzen. Beim Auslösen eines Passwort-Resets, wird dem Benutzer ein E-Mail mit einem Link zugeschickt. Der Passwort-Reset-Link führt den Benutzer auf ein Formular auf welchem er/sie die Möglichkeit hat, ein neues Passwort zu setzen.

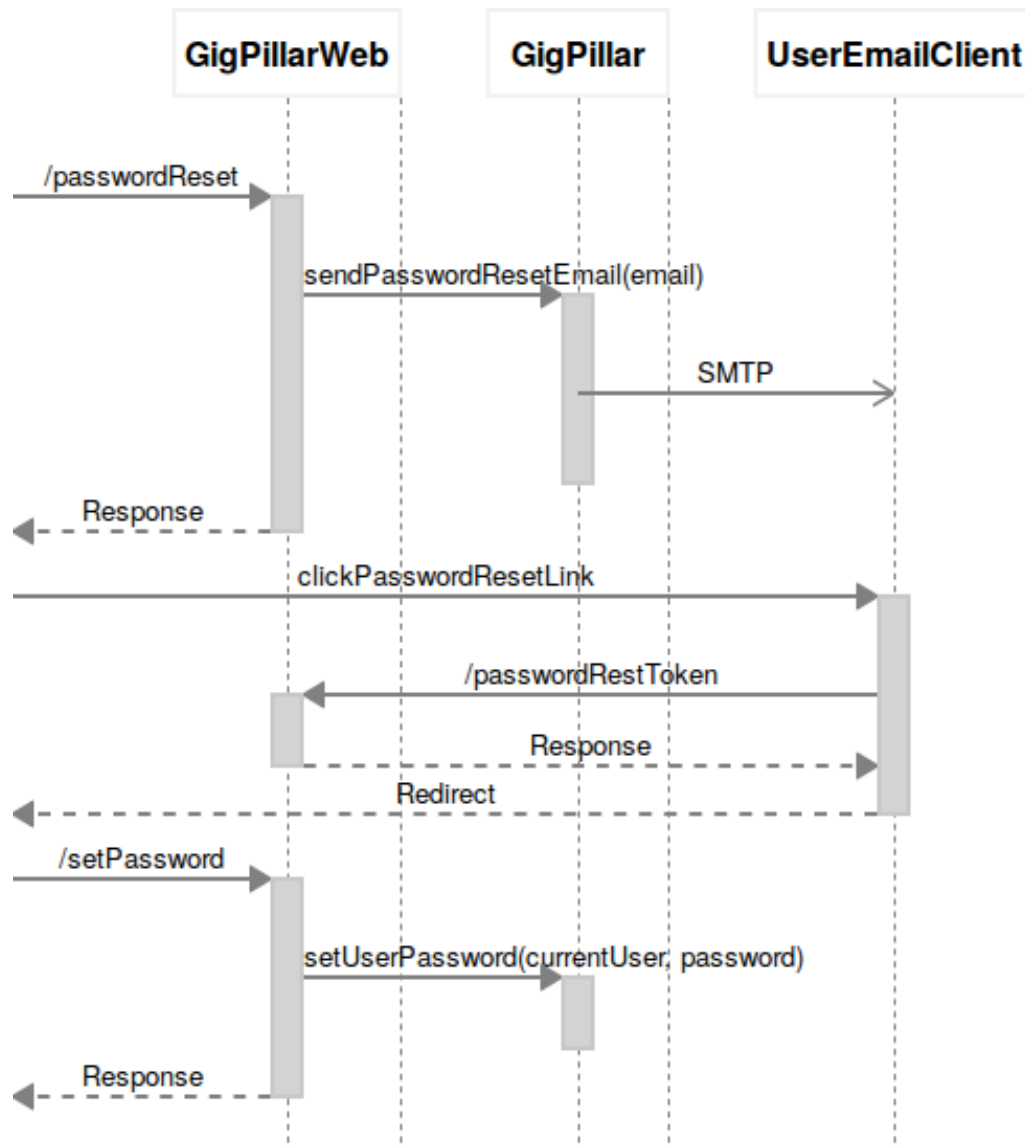


ABBILDUNG H.9: Datenfluss: Passwort-Reset

## H.4.2 Datenbankstruktur

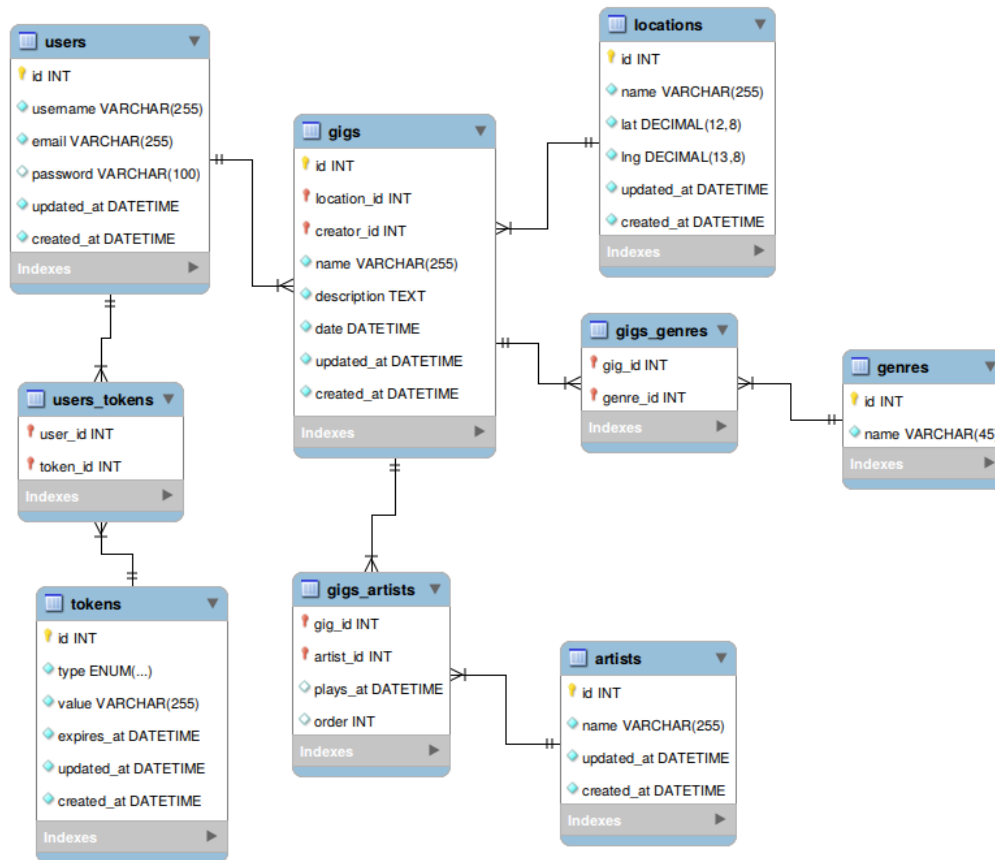


ABBILDUNG H.10: Konzept: Entity Relationship Diagram

## **H.5 Testkonzept**

### **H.5.1 Unit-Tests**

Der Applikationscode soll mit Unit-Tests getestet werden.

Unit-Tests testen einzelne Funktionen, d.h. hier sind Aufrufe über HTTP, wie sie durch den Browser ausgelöst würden, oder Datenbankabfragen ausgeschlossen.

### **H.5.2 Integration-Tests**

HTTP Requests sowie Datenabfragen sollen über Integration-Tests abgedeckt werden. In den Integration-Tests wird vor allem Business-Logik wie z.B. Validierungen von Daten getestet.

### **H.5.3 Browser-Tests**

Mit der in der Studie evaluierten Testing Library «Wallaby» sollen die gängigsten Benutzer Use-Cases getestet werden.

### **H.5.4 Visual-Tests**

Es soll für jede Ansicht während den Tests mindestens ein Screenshot für [percy.io](https://percy.io) erstellt werden.

### H.5.5 Akzeptanztests

Die Akzeptanztest sind vom Projektleiter vor Abschluss der Realisierung durchzuführen. Der Umfang der Akzeptanztests basiert auf den Kriterien die im Anforderungskatalog definiert wurden.

Technische Kriterien wie die Indexierbarkeit wurden in den Akzeptanztests bewusst ausgelassen, die Tests sollen möglichst unabhängig vom System durchführbar sein.

Test 01	Tester:	Datum:
<b>Kriterium</b>	Es ist möglich nach Konzerten in einem bestimmten Ort zu suchen.	
<b>Erwartetes Ergebnis</b>	Nach auswählen von «Berlin» in der Suche, werden nur noch Konzerte in Berlin aufgelistet.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.1: Akzeptanztest 01

Test 02	Tester:	Datum:
<b>Kriterium</b>	Es ist möglich eine Suche weiter nach Genre einzuschränken.	
<b>Erwartetes Ergebnis</b>	Nach auswählen von «Rock» in einem Suchresultat, werden nur noch Rock-Konzerte aufgelistet.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.2: Akzeptanztest 02

Test 03	Tester:	Datum:
<b>Kriterium</b>	Responsive - Homepage	
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.3: Akzeptanztest 03

Test 04	Tester:	Datum:
<b>Kriterium</b>	Browserkompatibilität - Homepage	
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.4: Akzeptanztest 04



<b>Test 05</b>	<b>Tester:</b>	<b>Datum:</b>
<b>Kriterium</b>	Responsive - Suche	
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.5: Akzeptanztest 05

<b>Test 06</b>	<b>Tester:</b>	<b>Datum:</b>
<b>Kriterium</b>	Browserkompatibilität - Suche	
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.6: Akzeptanztest 06

Test 07	Tester:	Datum:
<b>Kriterium</b>	Responsive - Gig Ansicht	
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.7: Akzeptanztest 07

Test 08	Tester:	Datum:
<b>Kriterium</b>	Browserkompatibilität - Gig Ansicht	
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.8: Akzeptanztest 08

Test 09	Tester:	Datum:
Kriterium	Responsive - Login	
Erwartetes Ergebnis	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.	
Testergebnis		
Fehlerbeschreibung		

TABELLE H.9: Akzeptanztest 09

Test 10	Tester:	Datum:
Kriterium	Browserkompatibilität - Login	
Erwartetes Ergebnis	Funktioniert in den unterstützten Browsern.	
Testergebnis		
Fehlerbeschreibung		

TABELLE H.10: Akzeptanztest 10

<b>Test 11</b>	<b>Tester:</b>	<b>Datum:</b>
<b>Kriterium</b>	Responsive - Registrierung	
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.11: Akzeptanztest 11

<b>Test 12</b>	<b>Tester:</b>	<b>Datum:</b>
<b>Kriterium</b>	Browserkompatibilität - Registrierung	
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.12: Akzeptanztest 12

Test 13	Tester:	Datum:
<b>Kriterium</b>	Responsive - Gig erfassen	
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.13: Akzeptanztest 13

Test 14	Tester:	Datum:
<b>Kriterium</b>	Browserkompatibilität - Gig erfassen	
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.14: Akzeptanztest 14

Test 15	Tester:	Datum:
<b>Kriterium</b>	Gig erfassen	
<b>Erwartetes Ergebnis</b>	Folgende Daten können erfasst werden: <ul style="list-style-type: none"> <li>• Name</li> <li>• Bild (<i>optional</i>)</li> <li>• Location</li> <li>• Datum</li> <li>• Zeit (<i>optional</i>)</li> <li>• Künstler mit optionaler Start-Zeit</li> <li>• Beschreibung</li> <li>• Link zum Ticketvertreiber (<i>optional</i>)</li> </ul>	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.15: Akzeptanztest 15

Test 16	Tester:	Datum:
<b>Kriterium</b>	Neue Gigs tauchen in der Suche auf.	
<b>Erwartetes Ergebnis</b>	Der neu erstellte Gig taucht in der Suche auf.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.16: Akzeptanztest 16

<b>Test 17</b>	<b>Tester:</b>	<b>Datum:</b>
<b>Kriterium</b>	Responsive - Benutzerprofil	
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.17: Akzeptanztest 17

<b>Test 18</b>	<b>Tester:</b>	<b>Datum:</b>
<b>Kriterium</b>	Browserkompatibilität - Benutzerprofil	
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.18: Akzeptanztest 18

Test 19	Tester:	Datum:
<b>Kriterium</b>	Responsive - Passwort-Reset	
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.19: Akzeptanztest 19

Test 20	Tester:	Datum:
<b>Kriterium</b>	Browserkompatibilität - Passwort-Reset	
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.20: Akzeptanztest 20



<b>Test 21</b>	<b>Tester:</b>	<b>Datum:</b>
<b>Kriterium</b>	Security - Suche	
<b>Erwartetes Ergebnis</b>	Das Suchfeld ist resistent gegen XSS und SQL-Injection	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.21: Akzeptanztest 21

<b>Test 22</b>	<b>Tester:</b>	<b>Datum:</b>
<b>Kriterium</b>	Security - Login	
<b>Erwartetes Ergebnis</b>	Das Login ist resistent gegen XSS und SQL-Injection	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.22: Akzeptanztest 22

Test 23	Tester:	Datum:
<b>Kriterium</b>	Security - Benutzerprofil	
<b>Erwartetes Ergebnis</b>	Das Benutzerprofil ist resistent gegen XSS und SQL-Injection	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.23: Akzeptanztest 23

Test 24	Tester:	Datum:
<b>Kriterium</b>	Security - Gig erfassen	
<b>Erwartetes Ergebnis</b>	Das Gig erfassen Formular ist resistent gegen XSS und SQL-Injection	
<b>Testergebnis</b>		
<b>Fehlerbeschreibung</b>		

TABELLE H.24: Akzeptanztest 24

## **H.6 Fazit**

### **H.6.1 Abweichungen**

Das erstellte Datenbankschema weicht vom Anforderungskatalog leicht ab, so wird vorgesehen, dass für die Zuweisung von Musik-Genres nicht der Künstler sondern das Konzert/der Gig verwendet wird. Dies soll das Erfassen eines Gigs vereinfachen, da so nicht der Prozess des Erfassens unterbrochen werden muss, nur um einem Künstler das entsprechende Genre hinzuzufügen.

Ausserdem wurde Entschieden, dass für Künstler und Locations keine eigenen Verwaltungen implementiert werden. Die Daten werden von externen Quellen wie z.B. der Google Places API bezogen.

### **H.6.2 Probleme**

Für die Location Autocomplete Funktion, ist es nicht klar, ob die Google Places API die nötigen Daten zur Verfügung stellt. So kann es gut sein, dass Locations nicht gefunden werden oder zuviele nicht relevante Resultate zurück gibt.

### **H.6.3 Machbarkeit**

Für die Machbarkeit während dem im Konzept ein Test mit der Google Places API gemacht. Die Google Places API liefert alle nötigen Daten, um die gewünschten Features umzusetzen.

### **H.6.4 Wirtschaftlichkeit**

In der Konzeptphase hat es keine Anzeichen auf Veränderungen, die die Wirtschaftlichkeit beeinflussen sollte.

### **H.6.5 Erweiterbarkeit**

Es bestehen einige Ideen, wie man die Applikation erweitern könnte. Ein Feature wäre zum Beispiel, dass beim Erfassen eines Gigs sogleich ein Facebook-Event angelegt würde. Weiter denkbar wäre eine Filter-Funktion, die die Gigs basierend auf einem Radius in Kilometer um eine Stadt einschränken würde.

### **H.6.6 Projektplan**

Trotz einer Verschiebung im Projektplans während der Initialisierung, wird nach der Konzeptphase der Projektplan soweit eingehalten werden können. Da die Screen-designs sowie das Testkonzept weniger Zeit beansprucht haben, bin ich nach dem Zwischenmeeting wieder im geplanten Bereich.



## Anhang I

# Sitzungsprotokoll — Zwischenmeeting

Teilnehmer:

- Sandro Bertolino
- Severin Rätz
- Damian Senn

Folgende Themen/Aktionen haben wir am Zwischenmeeting besprochen:

- Es wurden keine Reservern geplant. In Zukunft muss dies Berücksichtigt werden!
- Das Diagramm betreffend der Wirtschaftlichkeit mit dem Ansatz von Werbung ist unklar und wäre besser mit einem Linien-Diagramm gelöst worden.
- Bei den Projektkosten wurde ein Copy&Paste Fehler gemacht.
- Punkt E5 im Anhang verweist auf sich selbst.
- Neben dem Abkürzungsverzeichnis soll ein Glossar geführt werden, welcher Begriffe ausführlicher Beschreibt.
- Der Abgabetermin für den Diplombericht wurde auf den 17.05.2019 datiert.
- Das Abschlussmeeting wird am 21.05.2019 um 09:00 in Bern an der Belpstrasse 37 gehalten.



## Anhang J

# Realisierung

### J.1 HTML-Prototyp

Der HTML-Prototyp ist im Verzeichnis «html-prototype» zu finden, dieser wurde mit **Ember.js** und SASS umgesetzt. Um den Prototypen zu starten muss **Node.js** und **Yarn** installiert sein.

Den Prototypen kann man mit dem folgenden Kommando starten:

```
1 $ yarn && yarn start
```

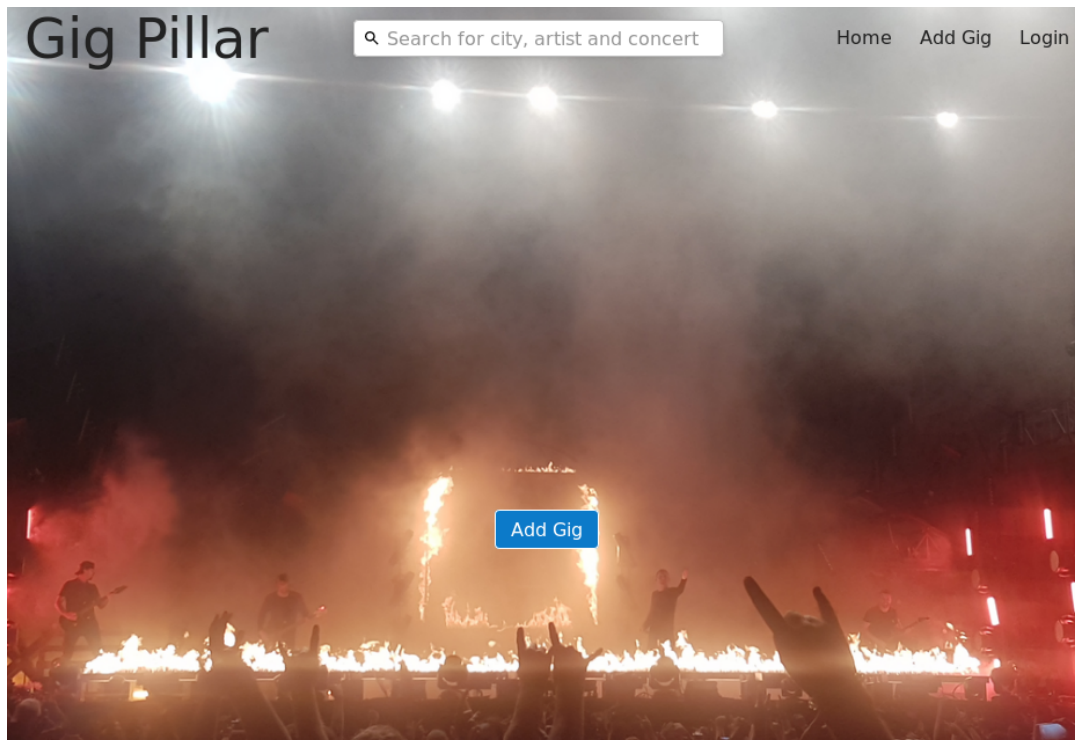
Danach kann der Prototyp über die acrshorturl <http://localhost:4200/> geöffnet werden.

Folgende URLs sind verfügbar:

- <http://localhost:4200/>
- <http://localhost:4200/search>
- <http://localhost:4200/gig-detail>
- <http://localhost:4200/add-gig>
- <http://localhost:4200/login>
- <http://localhost:4200/register>

### J.1.1 Screenshots

#### Homepage



#### Gigs in Bern [change location](#)



24.12.2019  
[Anti-Flag](#)



24.12.2019  
[Darkside](#)



24.12.2019  
[Liquid Session](#)

ABBILDUNG J.1: HTML Prototyp: Homepage



## Suche

# Gig Pillar

[Home](#) [Add Gig](#) [Login](#)

---

**Bern** [change location](#)


From

05 / 16 / 2019


To

mm / dd / yyyy


EDM




24.12.2019  
Darkside - Gridlok  
Dachstock - Bern, Switzerland




24.12.2019  
Darkside - Gridlok  
Dachstock - Bern, Switzerland




24.12.2019  
Darkside  
Dachstock - Bern, Switzerland



24.12.2019  
Liquid Session  
Dachstock - Bern, Switzerland



24.12.2019  
Darkside - Gridlok  
Dachstock - Bern, Switzerland



24.12.2019  
Anti-Flag  
Dachstock - Bern, Switzerland

ABBILDUNG J.2: HTML Prototyp: Suche

## Gig erfassen

# Gig Pillar

[Home](#) [Add Gig](#) [Login](#)

---

Name

Where

When

Doors

Who

Describe Gig

Tickets

ABBILDUNG J.3: HTML Prototyp: Gig erfassen

**Login**

---

**Gig Pillar**

[Home](#) [Add Gig](#) [Login](#)

---

**Login**

Email

Password

[Not registered yet?](#)

ABBILDUNG J.4: HTML Prototyp: Login

**Registrierung**

---

**Gig Pillar**

Search for city, artist and concert

Home Add Gig Login

---

**Username**

**Email**

**Password**

**Repeat password**

Create your Account

ABBILDUNG J.5: HTML Prototyp: Registrierung

## J.2 Projekt Setup

Die Initialisierung des Projekts wurde mit der Phoenix Framework «Umbrella» Struktur erstellt.

```
1 $ mix phx.new gigpillar —umbrella
```

Die Umbrella Struktur trennt die Applikation in kleinere Teilapplikationen, dies ermöglicht eine klarere Trennung zwischen der Webapplikation und der Businesslogik.

Projektstruktur:

- **/apps/**  
Die Phoenix Teilapplikationen
- **/apps/gigpillar/**  
Die Gigpillar Grundapplikation
- **/apps/gigpillar\_web/**  
Die Gigpillar Webapplikation
- **/apps/gigpillar\_web/assets/**  
Der CSS und JavaScript Code für die Webapplikation
- **/html-prototype/**  
Der HTML Prototyp
- **/doc/**  
Die Projektdokumentation
- **/doc/main.pdf**  
Das PDF der Projektdokumentation

### J.3 Dependency Management

Für das Dependency Management, wurde ein Bot eingerichtet, der Benachrichtigungen, bzw. Pull-Requests, bei Updates zustellt.

Für das Projekt wurde der Bot «Dependabot»<sup>1</sup> ausgewählt, da dieser Elixir sowie JavaScript Abhängigkeiten unterstützt.

Durch die Benützung des Dependabot, können während der Entwicklung des Projektes die Software Abhängigkeiten jederzeit auf dem neusten Stand gehalten werden.

Installation von Dependabot:

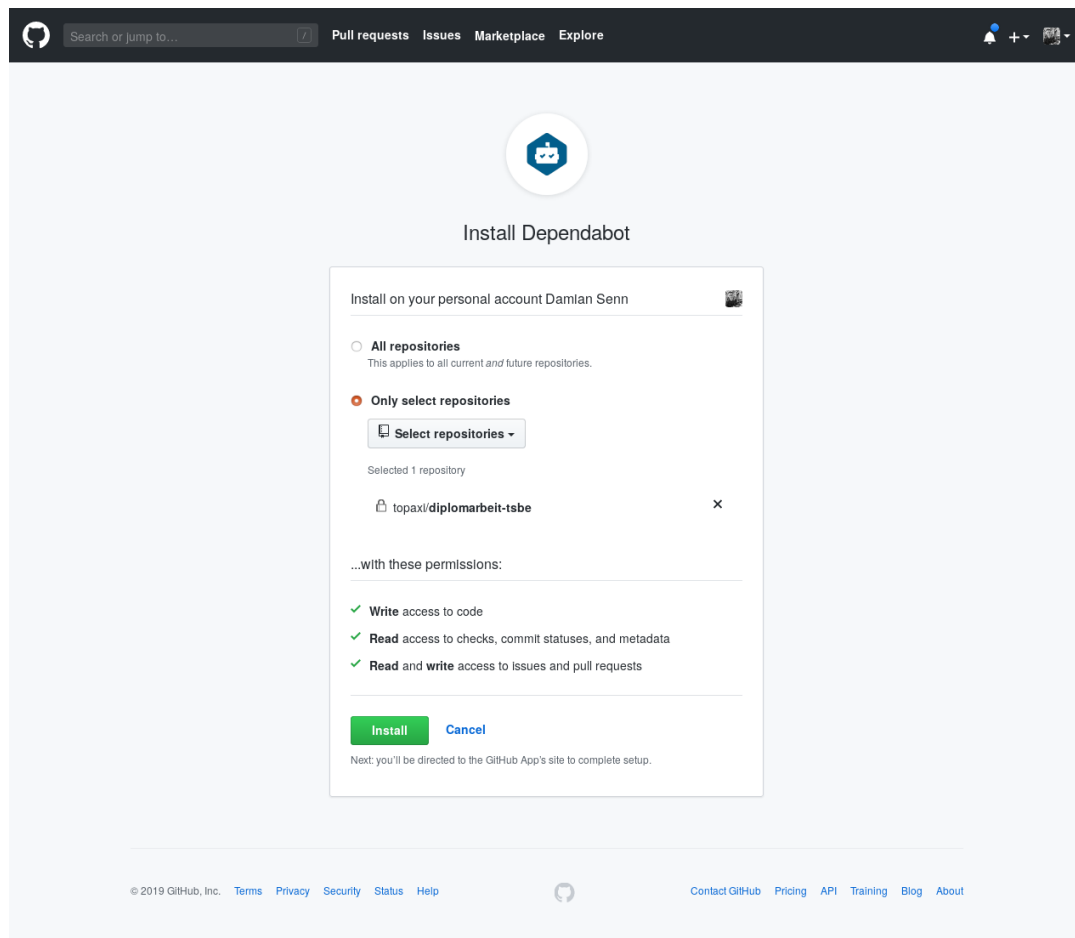


ABBILDUNG J.6: Dependabot: Installation

<sup>1</sup><https://github.com/marketplace/dependabot>

diplomarbeit-tsbe

private

>

/mix.exs

...

last checked 8 hours ago

Bump now

### Settings

#### Update schedule

Daily updates

Dependabot will batch pull requests daily

[Change run day and time for this account](#)

#### Directory (optional)

/

Relative to repository's root

#### Target branch (optional)

Branch to create pull requests against. If blank Dependabot will use your repo's default branch (master).

#### Filters

☐ Only security updates

☐ Only lockfile updates (ignore updates that require Mixfile changes)

☒ Only top-level dependencies (and security patches for subdependencies)

Update settings

### GitHub PR Defaults

#### Reviewers

None yet

+ Add a reviewer

#### Assignees

None yet

+ Add an assignee

#### Labels

None yet

+ Add a label

Defaults set on new PRs for this repo and language

### Delete language

When you delete this language Dependabot won't close any of the open pull requests it has created against this repo ([view pull requests](#))

Delete

ABBILDUNG J.7: Dependabot: Konfiguration für Elixir

diplomarbeit-tsbeprivate>...igpillar\_web/assets/package.json...last checked 8 hours agoBump now

### Settings

**Update schedule**

Daily updates

Dependabot will batch pull requests daily  
[Change run day and time for this account](#)

**Directory (optional)**

/apps/gigpillar\_web/assets

Relative to repository's root

**Target branch (optional)**

Branch to create pull requests against. If blank Dependabot will use your repo's default branch (master).

**Filters**

☐ Only security updates

☐ Only lockfile updates (ignore updates that require package.json changes)

**Update strategy for package.json**

How should Dependabot update your package.json (as opposed to your lockfile)?

Auto (bump versions if an app, widen ranges if a library)

Update settings

### GitHub PR Defaults

**Reviewers**

None yet

+ Add a reviewer

**Assignees**

None yet

+ Add an assignee

**Labels**

None yet

+ Add a label

Defaults set on new PRs for this repo and language

### Delete language

When you delete this language Dependabot won't close any of the open pull requests it has created against this repo ([view pull requests](#))

ABBILDUNG J.8: Dependabot: Konfiguration für JavaScript



Diverse Updates konnten via Pull-Requests von Dependabot während der Entwicklung vorgenommen werden:

The screenshot shows the GitHub interface for the repository `topaxi/diplomarbeit-tsbe`. The 'Pull requests' tab is selected, displaying a list of 10 closed pull requests generated by Dependabot. Each entry includes a title, a list of labels (e.g., `dependencies`, `elixir`, `javascript`), and a comment indicating when the pull request was merged.

Open	Closed	Author	Labels	Projects	Milestones	Reviews	Assignee	Sort
<input type="checkbox"/>	<input checked="" type="checkbox"/>	dependabot[bot]	dependencies, elixir					
Bump phoenix from 1.4.5 to 1.4.6 #10 by dependabot[bot] was merged 3 hours ago								
<input type="checkbox"/>	<input checked="" type="checkbox"/>	dependabot[bot]	dependencies, javascript					
Bump babel-loader from 8.0.5 to 8.0.6 in /apps/gigpillar_web/assets #9 by dependabot[bot] was merged a day ago								
<input type="checkbox"/>	<input checked="" type="checkbox"/>	dependabot[bot]	dependencies, elixir					
Bump ecto_sql from 3.1.1 to 3.1.2 #8 by dependabot[bot] was merged a day ago								
<input type="checkbox"/>	<input checked="" type="checkbox"/>	dependabot[bot]	dependencies, javascript					
Bump rxjs from 6.5.1 to 6.5.2 in /apps/gigpillar_web/assets #7 by dependabot[bot] was merged 4 days ago								
<input type="checkbox"/>	<input checked="" type="checkbox"/>	dependabot[bot]	dependencies, javascript					
Bump webpack from 4.30.0 to 4.31.0 in /apps/gigpillar_web/assets #6 by dependabot[bot] was merged 4 days ago								
<input type="checkbox"/>	<input checked="" type="checkbox"/>	dependabot[bot]	dependencies, elixir					
Bump postgres from 0.14.2 to 0.14.3 #5 by dependabot[bot] was merged 5 days ago								
<input type="checkbox"/>	<input checked="" type="checkbox"/>	dependabot[bot]	dependencies, elixir					
Bump phoenix from 1.4.4 to 1.4.5 #4 by dependabot[bot] was merged 5 days ago								
<input type="checkbox"/>	<input checked="" type="checkbox"/>	dependabot[bot]	dependencies, elixir					
Bump coverex from 1.4.15 to 1.5.0 #3 by dependabot[bot] was merged 8 days ago								
<input type="checkbox"/>	<input checked="" type="checkbox"/>	dependabot[bot]	dependencies, javascript					
Bump webpack-cli from 3.3.1 to 3.3.2 in /apps/gigpillar_web/assets #2 by dependabot[bot] was merged 9 days ago								
<input type="checkbox"/>	<input checked="" type="checkbox"/>	dependabot[bot]	dependencies, elixir					
Bump phoenix from 1.4.3 to 1.4.4 #1 by dependabot[bot] was merged 9 days ago								

ABBILDUNG J.9: Dependabot: Pull-Requests für Updates

## J.4 Datenbankschema

Die folgenden Änderungen wurden am von der Konzeptphase vorgegebenen Schema vorgenommen.

### Alle Entitäten

Alle «created\_at» Felder wurden nach «inserted\_at» umbenannt, da dies die Standardbenennung des Phoenix Frameworks ist.

### User

Der Benutzer Entität wurde das Feld «password» nach «password\_hash» umbenannt, damit klar ist, dass nicht ein Passwort sondern nur ein Hash abgespeichert wird. Das Passwort wird mit Argon2 verschlüsselt und ist somit eine sichere Verschlüsselung.

### Genre

Der Genre Entität sind im Konzept die Datumsfelder «update\_at» und «inserted\_at» vergessen gegangen und wurden in der Realisierung nachgeführt.

### Gig

In der Gig Entität wurden drei weitere Felder hinzugefügt. Die Felder «uuid» und «picture» dienen dazu, die beim Erfassen sowie Bearbeiten eines Gigs hochgeladenen Bilder zu identifizieren. Das zusätzliche Feld «tickets» ermöglicht es, beim Erfassen eines Gigs einen Link zum Ticketvorverkauf zu hinterlegen.

### Location

Die Location Entität erhielt bei der Realisierung zwei neue Felder, «address» für die Adresse der Location und «google\_place\_id» um die Referenz der Google API zu erhalten.

## J.4.1 Finales Schema

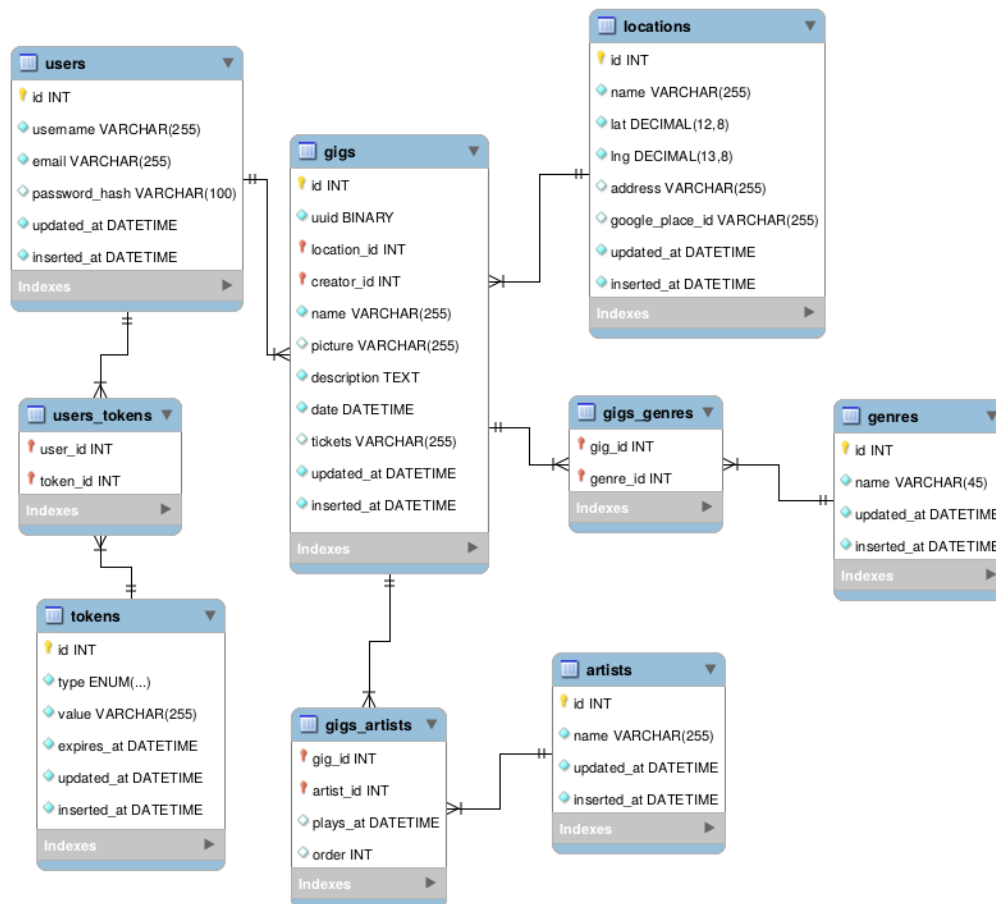


ABBILDUNG J.10: Realisierung: Entity Relationship Diagram

## J.5 Berechtigungssystem

Das Bearbeiten von Gigs wurde so eingeschränkt, dass nur angemeldete Benutzer Gigs erstellen dürfen und Benutzer nur eigene Gigs bearbeiten können.

Dazu wurde die Library «Canary»<sup>2</sup> verwendet. In der Datei «apps/gigpillar/-lib/abilities.ex» wurden die Berechtigungen wie gefolgt umgesetzt:

```
1 alias Gigpillar.Accounts.User
2 alias Gigpillar.Gigs.Gig
3
4 defimpl Canada.Can, for: User do
5   # User can list their gigs and create new gigs
6   def can?(%User{}, action, Gig)
7     when action in [:index, :new, :create],
8       do: true
9
10  # User can edit and delete their own gigs
11  def can?(%User{id: uid}, action, %Gig{creator_id: uid})
12    when action in [:edit, :update, :delete],
13      do: true
14
15  # User can show any gig
16  def can?(%User{}, :show, %Gig{}), do: true
17
18  # User is not allowed to do anything else
19  def can?(%User{}, _, _), do: false
20 end
21
22 defimpl Canada.Can, for: Atom do
23   # Anyone can show a specific gig
24   def can?(nil, :show, %Gig{}), do: true
25
26   # By default, nothing is permitted
27   def can?(nil, _, _), do: false
28 end
```

---

<sup>2</sup><https://github.com/cpjk/canary>

## J.6 Location Autocomplete

Das Location Autocomplete Feld wurde mit der «Google Place API»<sup>3</sup> umgesetzt.

Die verwendete Library «google-api-elixir-client» implementiert die Autocomplete API, jedoch fehlte noch die entsprechende Zusatzfunktion um weitere Details, wie die geographischen Koordinaten, Adresse, etc., zu den gefundenen Locations abzufragen.

Die API um Details abzufragen wurden im Rahmen von diesem Projekt umgesetzt und zurück an das originale Projekt beigesteuert.

Ausserdem musste eine Abhängigkeit auf den aktuellsten Stand gebracht werden.

Die beiden Beiträge für die Library sind auf Github zu finden:

- <https://github.com/seanabrahams/google-api-elixir-client/pull/10/files>
- <https://github.com/seanabrahams/google-api-elixir-client/pull/11/files>

---

<sup>3</sup><https://developers.google.com/places/web-service/autocomplete>

## J.7 HTML Erweiterungen

Während der Entwicklung des Projektes, wurden einige spezielle HTML-Elemente definiert. Diese Elemente wurden mit **LitElement**<sup>4</sup> umgesetzt, und bieten gegenüber den Standard HTML-Elementen eigens definierte Verhaltensweisen.

### <search-box>

Das **<search-box>** Element wird für die Suche sowie diverse Autocomplete-Elemente verwendet. Stylistisch sieht die Suchbox aus wie ein normales Texteingabefeld, mit einer kleiner Lupe als Symbol.

Beispiel:

```
1 <search-box
2   inputId="suche"
3   src="/api/autocomplete"
4   name="suche"
5   placeholder="Suche ..."
6   value="Suchbegriff"
7   debounce-time="300"></search-box>
```

Attribute:

- **inputId**: Das **id** Attribut für das Input Element innerhalb der Suchbox, z.B. im Zusammenhang mit einem **<label>** Element.
- **src**: acrshorturl für die Datenabfrage, bei Eingaben in das Textfeld wird jeweils eine **HTTP** Abfrage ausgelöst und ein **search-result** Ereignis ausgelöst.
- **name**: Der Name des Form-Elements.
- **placeholder**: Platzhaltertext welcher dargestellt wird wenn das Textfeld leer ist.
- **value**: Der Wert, welcher mit dem HTML-Formular mitgeschickt wird.
- **debounce-time**: Zeit in Millisekunden die mindestens vergehen muss, bevor eine neue Abfrage an den Server geschickt wird.

### <with-dropdown>

Mit dem **<with-dropdown>** Element können Dropdowns realisiert werden. Elemente mit dem Attribut **slot="dropdown"** werden initial nicht dargestellt. Wird ein Element innerhalb des **<with-dropdown>** Element fokussiert, so werden die mit **slot="dropdown"** gekennzeichneten Elemente innerhalb eines Dropdown-Elements dargestellt.

Beispiel:

```
1 <with-dropdown>
2   <input type="text">
3   <ul slot="dropdown">
4     <li>Lorem</li>
5     <li>Ipsum</li>
6   </ul>
7 </with-dropdown>
```

<sup>4</sup><https://lit-element.polymer-project.org/>

### <location-input>

Das **<location-input>** Element wird verwendet um für einen Gig eine Location auszuwählen. Es repräsentiert ein Suchfeld, das eine Abfrage über die Google Place API macht. Wird eine Location ausgewählt, wird das Suchfeld durch einen Text sowie Button ersetzt. Der Text beinhaltet den Namen der Location und der Button ermöglicht es, die ausgewählte Location mit einer Neuen zu ersetzen.

Beispiel:

```
1 <location-input
2   inputId="gig_location"
3   name="gig[location]"
4   location="{
5     "name": "Dachstock",
6     "google_place_id": "ChIJ-SskKr45jkcRPqmGB-ZGsRE"
7   }"></location-input>
```

Attribute:

- **inputId**: Das **id** Attribut für das Input Element innerhalb des Location-Input, z.B. im Zusammenhang mit einem **<label>** Element.
- **name**: Der Name des Form-Elements.
- **location**: Ein **acrshortjson**-Objekt einer Location, bestehend aus **name** und einer **id** oder **google\_place\_id**.

### <picture-input>

Als verbessertes File-Input, wurde ein Ersatz für ein **<input type=file>** Element geschrieben. Das **<picture-input>** Element funktioniert als kompatibler Ersatz und bietet die selbe Funktionalität an. Der Hauptunterschied liegt in der Darstellung, so wird beim **<picture-input>** jederzeit eine Vorschau für das ausgewählte Bild dargestellt.

Beispiel:

```
1 <picture-input
2   inputId="gig-picture"
3   name="gig[picture]"
4   value="http://example.com/my-picture.png"></picture-input>
```

Attribute:

- **inputId**: Das **id** Attribut für das Input Element innerhalb des Picture-Input, z.B. im Zusammenhang mit einem **<label>** Element.
- **name**: Der Name des Form-Elements.
- **value**: **acrshorturl** oder Datei des ausgewählten Bildes.

**<datetime-input>**

Das **<datetime-input>** Element kombiniert ein **<input type=date>** mit einem **<input type=time>** zu einem Feld zusammen.

Beispiel:

```

1 <datetime-input
2   inputId="datetime"
3   dateLabel="Datum"
4   timeLabel="Uhrzeit"
5   name="datetime"
6   value="2019-05-14T17:50:14.608Z"></datetime-input>

```

Attribute:

- **inputId**: Das **id** Attribut für das Input Element innerhalb des Datetime-Input, z.B. im Zusammenhang mit einem **<label>** Element.
- **dateLabel**: Beschriftung für das Datumsfeld.
- **timeLabel**: Beschriftung für das Zeitfeld.
- **name**: Der Name des Form-Elements.
- **value**: Datum mit Zeit im **ISO 8601**<sup>5</sup> Format.

**<artists-input>**

Das **<artists-input>** ist ein spezifisches Feld für das Erfassen von Künstlern für ein Konzert. Es Sucht über den Server bereits existierende Künstler, und bietet diese zur Auswahl an. Zusätzlich kann für jeden ausgewählten Künstler eine Zeit angegeben werden, an welcher deren Auftritt stattfindet.

Beispiel:

```

1 <artists-input
2   inputId="artists"
3   name="gig[artists]"
4   placeholder="K&uuml;nstler_suchen ..."
5   value="[
6     { "id":42, "name": "Parkway Drive", "plays_at": "23:00" },
7     { "id":23, "name": "The Ghost Inside", "plays_at": "21:00" }
8   ]"></artists-input>

```

Attribute:

- **inputId**: Das **id** Attribut für das Input Element innerhalb des Artists-Input, z.B. im Zusammenhang mit einem **<label>** Element.
- **name**: Der Name des Form-Elements.
- **placeholder**: Platzhalter für das Suchfeld.
- **value**: Eine Liste von **acrshortjson**-Objekten der einem Gig assoziierten Künstler.

<sup>5</sup>[https://de.wikipedia.org/wiki/ISO\\_8601](https://de.wikipedia.org/wiki/ISO_8601)



## J.8 Asset Optimierungen

Für die Produktionsumgebung, wurden diverse Optimierungen vorgenommen:

- Terser Plugin um JavaScript Code zu optimieren.
- Plugin um Funktionen zu deduplizieren.
- HTML Minifier Plugin um HTML innerhalb von JavaScript zu optimieren.
- Imagemin Plugin um Bilder auf Dateigrösse zu optimieren.
- CSS Optimierungsplugin
- Zopfli Kompression
- Brotli Kompression

Diese Plugins reduzieren die Dateigrößen von JavaScript, CSS sowie Bilddateien. Die reduzierte Grösse kommt Benutzern mit schlechteren Internetverbindungen, wie z.B. im mobilen Netz, entgegen.

Optimierungen	Datei	Grösse
Keine	app.js	1000 KiB
Produktionsmodus	app.js	397 KiB
" und Terser Plugin	app.js	125 KiB
" und Funktionsdeduplizierung	app.js	123 KiB
" und HTML Minifier	app.js	122 KiB
" mit Zopfli	app.js.gz	25.2 KiB
" mit Brotli	app.js.br	23.0 KiB

TABELLE J.1: JavaScript Optimierungen

Optimierungen	Datei	Grösse
Keine	app.css	10.5 KiB
Produktionsmodus	app.css	10.5 KiB
" und CSS Optimierungsplugin	app.css	8.58 KiB

TABELLE J.2: CSS Optimierungen

Optimierungen	Datei	Grösse
Keine	background-1.jpg	1.79 MiB
Produktionsmodus	background-1.jpg	1.79 MiB
" und Imagemin (Qualität 75)	background-1.jpg	278 KiB

TABELLE J.3: Bilder Optimierungen

## J.9 File Upload

Für den Upload der Bilder von Gigs, wird die Software «**minio**»<sup>6</sup> eingesetzt. Die Software ist mit der **Amazon Simple Storage Service** (kurz **Amazon S3**) kompatibel, und bietet sich daher stark an da es bereits viele Libraries gibt, die den Upload erleichtern.

Für die Anbindung an **minio** werden die Libraries **ArcEcto**, **Arc** und **ExAws** eingesetzt.

Da die Bilder während dem Erstellen eines Gigs bereits hochgeladen werden, existiert für die Gigs noch kein **id** Feld, dass von der Datenbank automatisch generiert wird. Um die Bilder dennoch referenzieren zu können, wurde im Datenbankschema ein Feld «**uuid**» eingefügt (siehe J.4). Das **uuid** Feld wird mit einem möglichst einzigartigen Wert abgefüllt, der Wert entspricht dem **Universally Unique Identifier (UUID) version 4**<sup>7</sup> Standard.

Die **Arc** Library bietet eine einfache API an, um die hochgeladenen Bilder auf die von der Webapplikation verwendeten Grössen zuzuschneiden. Dazu wird auf dem Server die Software «**ImageMagick**»<sup>8</sup> benötigt.

Es werden für jedes hochgeladene Bild zwei Grössen generiert, **288×224** Pixel für in Auflistungen und **160×56** Pixel für Suchresultate sowie im Bearbeitungsformular.

So werden bei einem Upload folgende Bilder abgelegt:

- gigs/{uuid}/original.ext
- gigs/{uuid}/list.ext
- gigs/{uuid}/thumbnail.ext

## J.10 OpenStreetMap

Auf der Gig Detailansicht, wird eine OpenStreetMap eingebunden. Damit die OpenStreetMap Karte korrekt dargestellt wird, mussten die Geokoordinaten auf einzelne sogenannte «Tiles» berechnet werden. Ein Tile ist ein einzelnes **256×256** Pixel Teilbild der Weltkarte, die Tiles unterscheiden sich je nach Zoomlevel und muss entsprechend ausgerechnet werden.

Die genauere Beschreibung der Funktionsweise ist auf dem **OpenStreetMap Wiki** Dokumentiert<sup>9</sup>.

Für die Programmiersprache Elixir ist auf dem Wiki kein Beispiel vorhanden und musste entsprechend adaptiert werden.

---

<sup>6</sup><https://min.io/>

<sup>7</sup>[https://de.wikipedia.org/wiki/Universally\\_Unique\\_Identifier](https://de.wikipedia.org/wiki/Universally_Unique_Identifier)

<sup>8</sup><https://imagemagick.org/>

<sup>9</sup>[https://wiki.openstreetmap.org/wiki/Slippy\\_map\\_tilenames](https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames)

## J.11 SEO - Microdata

Auf der Gig Detailansicht wird für Suchmaschinen ein zusätzliches Element mit **acrshortjsonld** ausgegeben. Dieses Element ist für die normalen Besucher nicht sichtbar.

Beispiel:

```
1 <script type="application/ld+json">
2 {
3   "@context": "http://schema.org",
4   "@type": "MusicEvent",
5   "name": "Darkside",
6   "startDate": "2019-04-20T22:00:00Z",
7   "image": "http://localhost:4040/uploads/gigs/051cd-...",
8   "description": "Die_DARKSIDE_beherbergt_diesen...",
9   "performer": [
10    { "name": "Dom_&_Roland", "@type": "MusicGroup" },
11    { "name": "FD", "@type": "MusicGroup" },
12    { "name": "Ryck", "@type": "MusicGroup" }
13  ],
14  "location": {
15    "@type": "MusicVenue",
16    "name": "Dachstock",
17    "address": "Neubrueckstrasse_8,_Bern,_Switzerland"
18  },
19  "offers": {
20    "@type": "Offer",
21    "url": "https://www.petzitickets.ch/"
22  }
23 }
24 </script>
```

Während der Umsetzung ist durch das Validierungs-Tool<sup>10</sup> von Google aufgefallen, dass diverse empfohlene Felder nicht vorhanden sind. Diese Felder sind optional und sind vom umgesetzten Datenmodell noch nicht abgedeckt.

MusicEvent		All (1) ▾
<div> <div>MusicEvent</div> <div>PREVIEW</div> <div>0 ERRORS 5 WARNINGS ^</div> </div>		
@type	MusicEvent	
name	Darkside	
startDate	2019-04-20T22:00:00+00:00	
image	https://www.dachstock.ch/wp-content/uploads/2019/02/dom_roland.jpg	
description	Die DARKSIDE beherbergt diesen...	
performer		
@type	MusicGroup	
name	Dom & Roland	
performer		
@type	MusicGroup	
name	FD	
performer		
@type	MusicGroup	
name	Ryck	
location		
@type	MusicVenue	
name	Dachstock	
address		
@type	PostalAddress	
name	Neubrueckstrasse 8, Bern, Switzerland	
offers		
@type	Offer	
url	https://www.petzitickets.ch/	
▲ availability	The <i>availability</i> field is recommended. Please provide a value if available.	
▲ price	The <i>price</i> field is recommended. Please provide a value if available.	
▲ priceCurrency	The <i>priceCurrency</i> field is recommended. Please provide a value if available.	
▲ validFrom	The <i>validFrom</i> field is recommended. Please provide a value if available.	
▲ endDate	The <i>endDate</i> field is recommended. Please provide a value if available.	

ABBILDUNG J.11: acrshortjsonld: Validierung

<sup>10</sup><https://search.google.com/structured-data/testing-tool>

## J.12 Suche

Die Suche wurde mit einem «einfachen» SQL umgesetzt, der eingegebene Suchtext wird nach Wörter aufgeteilt und in den entsprechenden Datenbankfeldern gesucht. Das Suchresultat kann ausserdem mit den Parametern **from**, **to** und **genre** weiter eingeschränkt werden.

Nachfolgend, die implementierte SQL Abfrage in Elixir:

```
1  def search_gigs(query, %{from: from} = options) do
2    search_gigs(query, Map.delete(options, :from))
3    |> where([g], g.date >= ^from)
4  end
5
6  def search_gigs(query, %{to: to} = options) do
7    search_gigs(query, Map.delete(options, :to))
8    |> where([g], g.date >= ^to)
9  end
10
11 def search_gigs(query, %{genre: genre} = options) do
12   search_gigs(query, Map.delete(options, :genre))
13   |> where([g, l, a, gg], gg.genre_id == ^genre)
14 end
15
16 def search_gigs(query, _options \\ []) do
17   query
18   |> String.split(" ")
19   |> Enum.reduce(
20     from(g in Gig,
21       distinct: [asc: g.date, desc: g.id],
22       left_join: l in assoc(g, :location),
23       left_join: a in assoc(g, :artists),
24       left_join: gg in assoc(g, :gig_genres),
25       preload: [:location, :artists]
26     ),
27     fn term, query ->
28       query
29       |> where(
30         [g, l, a],
31         ilike(g.name, ^"%#{term}%") or
32         ilike(l.name, ^"%#{term}%") or
33         ilike(l.address, ^"%#{term}%") or
34         ilike(a.name, ^"%#{term}%")
35       )
36     end
37   )
38 end
```

## J.13 Probleme

### J.13.1 Dependency Konflikt

#### ExAws

Die Library **ExAws** hat nach der Installation folgenden Fehler ausgelöst:

```
1 Failed to use "poison" (version 4.0.1) because
2   arc (version 0.11.0) requires ~> 2.2 or ~> 3.1
3   coverex (version 1.5.0)
4     requires ~> 3.0 or ~> 3.1 or ~> 4.0
5   deps/google_api_client/mix.exs
6     requires ~> 1.5 or ~> 2.0 or ~> 3.0 or ~> 4.0
7   wallaby (version 0.22.0) requires >= 1.4.0
8   mix.lock specifies 4.0.1
9
10  ** (Mix) Hex dependency resolution failed, relax the version
11     requirements of your dependencies or unlock them (by using
12     mix deps.update or mix deps.unlock). If you are unable to
13     resolve the conflicts you can try overriding with
14     {:dependency, "~> 1.0", override: true}
```

Erste Versuche, die Abhängigkeit in der Gigpillar oder GigpillarWeb Applikation zu überschreiben ist gescheitert. Dieses Problem konnte behoben werden, in dem im **mix.exs** der Umbrella Applikation die **poison** Abhängigkeit mit der **override** Option eingefügt wurde.

```
1 defmodule Gigpillar.Umbrella.MixProject do
2   defp deps do
3     [
4       {:poison, "~> 4.0", runtime: false, override: true}
5     ]
6   end
7 end
```

### J.13.2 Formularbehandlung in Phoenix

Da ich das Phoenix Framework bisher nur für **REST** APIs verwendet habe, hatte ich einige Probleme mit den vom Framework zur Verfügung gestellten Formularfunktionen. Dies hat mir viel mehr Zeit beansprucht als initial angenommen.

In den meisten Tutorials, wurde die API mit einer **form\_for** Funktion vorgestellt, diese funktioniert jeweils nur im Zusammenhang mit einem Ecto-Changeset<sup>11</sup>. Jedoch wird beim Login sowie der Suche kein solches Changeset verwendet.

Nach ausgiebigem durchlesen der offiziellen Dokumentation vom Phoenix Framework, fand ich schlussendlich eine **form\_tag**<sup>12</sup> Funktion die auch ohne Changeset funktioniert.

Einige Zeit später fand ich jedoch im Phoenix Source Code<sup>13</sup> heraus, dass für das von Phoenix bereitgestellte Verbindungsobjekt **Plug.Conn** das **Phoenix.HTML.FormData** Protokoll implementiert wird.

Das Verwenden des **Plug.Conn** Objektes erleichtert die Benützung der Formular Funktionen um einiges, da entsprechende Form Aktionen nicht mehr explizit definiert werden müssen. Zudem wird das Zuordnen der Daten ins HTML automatisch vom Framework übernommen, was diverse Fehlerquellen eliminiert.

### J.13.3 Datumsbehandlung

Die Datum und Zeitanzeigen in der Applikation wurden mit der Zeit immer wie komplizierter. Insbesondere wenn Zeitzonen involviert sind, war es kaum mehr übersichtlich wann welche Daten vorhanden sind. Um nicht mehr zwischen den verschiedenen Datenstrukturen von Elixir zu unterscheiden müssen, wurde die Library «**Timex**»<sup>14</sup> installiert.

Timex bietet diverse Funktionen um jegliche Datum/Zeit bezogenen Datenstrukturen zu manipulieren und formatieren.

### J.13.4 Behandlung von Datenrelationen

Die Behandlung von Relationen im Datenmodell war für mich in dieser Form in Phoenix noch unbekannt. Es wurde einige Stunden darin investiert, die Daten des Gig erstellen Formulars korrekt abzuspeichern. Die in der Dokumentation behandelten Funktionen **put\_assoc** und **cast\_assoc** sind nicht eindeutig und klar beschrieben. Derzeit ist die Implementation mit **cast\_assoc** umgesetzt, soweit das aufgebaute Verständnis ist, sollte dies allerdings mit **put\_assoc** umgesetzt werden.

---

<sup>11</sup><https://hexdocs.pm/ecto/Ecto.Changeset.html>

<sup>12</sup>[https://hexdocs.pm/phoenix\\_html/Phoenix.HTML.Tag.html#form\\_tag/2](https://hexdocs.pm/phoenix_html/Phoenix.HTML.Tag.html#form_tag/2)

<sup>13</sup>[https://github.com/phoenixframework/phoenix\\_html/blob/f9a4f38/lib/phoenix\\_html/form\\_data.ex#L51](https://github.com/phoenixframework/phoenix_html/blob/f9a4f38/lib/phoenix_html/form_data.ex#L51)

<sup>14</sup><https://github.com/bitwalker/timex>

## J.14 Tests

Die Tests können über das Kommando **mix test** ausgeführt werden.

Beispiel:

```
topaxi@home:~/diplomarbeit-tsbe$ mix test
==> gigpillar
Excluding tags: [:skip]

.....

Finished in 3.1 seconds
72 tests, 0 failures, 25 excluded

Randomized with seed 245431
==> gigpillar_web
Excluding tags: [:skip]

.....

Finished in 7.7 seconds
18 tests, 0 failures, 4 excluded

Randomized with seed 245431
topaxi@home:~/diplomarbeit-tsbe$
```

ABBILDUNG J.12: Tests: Ausführung der automatisierten Tests

Einige Tests werden derzeit noch übersprungen, da nicht genug Zeit vorhanden war.

Damit die Browsertests funktionieren, muss auf dem Zielsystem der **Chromedriver** installiert werden. Chromedriver kann entweder von der offiziellen Webseite<sup>15</sup> heruntergeladen oder über den Node Package Manager (NPM) installiert werden.

Installation mit NPM:

```
1 $ npm install -g chromedriver
```

<sup>15</sup><http://chromedriver.chromium.org/>



## J.15 Testprotokoll

<b>Test 01</b>	<b>Tester:</b> Damian Senn <b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Es ist möglich nach Konzerten in einem bestimmten Ort zu suchen.
<b>Erwartetes Ergebnis</b>	Nach auswählen von «Berlin» in der Suche, werden nur noch Konzerte in Berlin aufgelistet.
<b>Testergebnis</b>	OK
<b>Fehlerbeschreibung</b>	

TABELLE J.4: Akzeptanztest 01

<b>Test 02</b>	<b>Tester:</b> Damian Senn <b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Es ist möglich eine Suche weiter nach Genre einzuschränken.
<b>Erwartetes Ergebnis</b>	Nach auswählen von «Rock» in einem Suchresultat, werden nur noch Rock-Konzerte aufgelistet.
<b>Testergebnis</b>	OK
<b>Fehlerbeschreibung</b>	

TABELLE J.5: Akzeptanztest 02

<b>Test 03</b>	<b>Tester:</b> Damian Senn	<b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Responsive - Homepage	
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.	
<b>Testergebnis</b>	OK	
<b>Fehlerbeschreibung</b>		

TABELLE J.6: Akzeptanztest 03

<b>Test 04</b>	<b>Tester:</b> Damian Senn	<b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Browserkompatibilität - Homepage	
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.	
<b>Testergebnis</b>	OK	
<b>Fehlerbeschreibung</b>		

TABELLE J.7: Akzeptanztest 04

<b>Test 05</b>	<b>Tester:</b> Damian Senn	<b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Responsive - Suche	
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.	
<b>Testergebnis</b>	OK	
<b>Fehlerbeschreibung</b>		

TABELLE J.8: Akzeptanztest 05

<b>Test 06</b>	<b>Tester:</b> Damian Senn	<b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Browserkompatibilität - Suche	
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.	
<b>Testergebnis</b>	OK	
<b>Fehlerbeschreibung</b>		

TABELLE J.9: Akzeptanztest 06

<b>Test 07</b>	<b>Tester:</b> Damian Senn <b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Responsive - Gig Ansicht
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.
<b>Testergebnis</b>	OK
<b>Fehlerbeschreibung</b>	

TABELLE J.10: Akzeptanztest 07

<b>Test 08</b>	<b>Tester:</b> Damian Senn <b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Browserkompatibilität - Gig Ansicht
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.
<b>Testergebnis</b>	OK
<b>Fehlerbeschreibung</b>	

TABELLE J.11: Akzeptanztest 08

<b>Test 09</b>	<b>Tester:</b> Damian Senn	<b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Responsive - Login	
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.	
<b>Testergebnis</b>	OK	
<b>Fehlerbeschreibung</b>		

TABELLE J.12: Akzeptanztest 09

<b>Test 10</b>	<b>Tester:</b> Damian Senn	<b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Browserkompatibilität - Login	
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.	
<b>Testergebnis</b>	OK	
<b>Fehlerbeschreibung</b>		

TABELLE J.13: Akzeptanztest 10

<b>Test 11</b>	<b>Tester:</b> Damian Senn <b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Responsive - Registrierung
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.
<b>Testergebnis</b>	OK
<b>Fehlerbeschreibung</b>	

TABELLE J.14: Akzeptanztest 11

<b>Test 12</b>	<b>Tester:</b> Damian Senn <b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Browserkompatibilität - Registrierung
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.
<b>Testergebnis</b>	OK
<b>Fehlerbeschreibung</b>	

TABELLE J.15: Akzeptanztest 12

<b>Test 13</b>	<b>Tester:</b> Damian Senn <b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Responsive - Gig erfassen
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.
<b>Testergebnis</b>	OK
<b>Fehlerbeschreibung</b>	

TABELLE J.16: Akzeptanztest 13

<b>Test 14</b>	<b>Tester:</b> Damian Senn <b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Browserkompatibilität - Gig erfassen
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.
<b>Testergebnis</b>	OK
<b>Fehlerbeschreibung</b>	

TABELLE J.17: Akzeptanztest 14

<b>Test 15</b>	<b>Tester:</b> Damian Senn <b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Gig erfassen
<b>Erwartetes Ergebnis</b>	Folgende Daten können erfasst werden: <ul style="list-style-type: none"> <li>• Name</li> <li>• Bild (<i>optional</i>)</li> <li>• Location</li> <li>• Datum</li> <li>• Zeit (<i>optional</i>)</li> <li>• Künstler mit optionaler Start-Zeit</li> <li>• Beschreibung</li> <li>• Link zum Ticketvertreiber (<i>optional</i>)</li> </ul>
<b>Testergebnis</b>	OK
<b>Fehlerbeschreibung</b>	

TABELLE J.18: Akzeptanztest 15

<b>Test 16</b>	<b>Tester:</b> Damian Senn <b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Neue Gigs tauchen in der Suche auf.
<b>Erwartetes Ergebnis</b>	Der neu erstellte Gig taucht in der Suche auf.
<b>Testergebnis</b>	OK
<b>Fehlerbeschreibung</b>	

TABELLE J.19: Akzeptanztest 16



<b>Test 17</b>	<b>Tester:</b> Damian Senn <b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Responsive - Benutzerprofil
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.
<b>Testergebnis</b>	Fehlerhaft
<b>Fehlerbeschreibung</b>	Die Benutzerprofil Funktionalität fehlt komplett.

TABELLE J.20: Akzeptanztest 17

<b>Test 18</b>	<b>Tester:</b> Damian Senn <b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Browserkompatibilität - Benutzerprofil
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.
<b>Testergebnis</b>	Fehlerhaft
<b>Fehlerbeschreibung</b>	Die Benutzerprofil Funktionalität fehlt komplett.

TABELLE J.21: Akzeptanztest 18

<b>Test 19</b>	<b>Tester:</b> Damian Senn <b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Responsive - Passwort-Reset
<b>Erwartetes Ergebnis</b>	Sieht auf Desktop, Tablet und Mobile gut aus und stellt jeweils alle relevanten Daten dar.
<b>Testergebnis</b>	Fehlerhaft
<b>Fehlerbeschreibung</b>	Es gibt keine Möglichkeit ein vergessenes Passwort zurückzusetzen.

TABELLE J.22: Akzeptanztest 19

<b>Test 20</b>	<b>Tester:</b> Damian Senn <b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Browserkompatibilität - Passwort-Reset
<b>Erwartetes Ergebnis</b>	Funktioniert in den unterstützten Browsern.
<b>Testergebnis</b>	Fehlerhaft
<b>Fehlerbeschreibung</b>	Es gibt keine Möglichkeit ein vergessenes Passwort zurückzusetzen.

TABELLE J.23: Akzeptanztest 20

<b>Test 21</b>	<b>Tester:</b> Damian Senn	<b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Security - Suche	
<b>Erwartetes Ergebnis</b>	Das Suchfeld ist resistent gegen XSS und SQL-Injection	
<b>Testergebnis</b>	OK	
<b>Fehlerbeschreibung</b>		

TABELLE J.24: Akzeptanztest 21

<b>Test 22</b>	<b>Tester:</b> Damian Senn	<b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Security - Login	
<b>Erwartetes Ergebnis</b>	Das Login ist resistent gegen XSS und SQL-Injection	
<b>Testergebnis</b>	OK	
<b>Fehlerbeschreibung</b>		

TABELLE J.25: Akzeptanztest 22

<b>Test 23</b>	<b>Tester:</b> Damian Senn	<b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Security - Benutzerprofil	
<b>Erwartetes Ergebnis</b>	Das Benutzerprofil ist resistent gegen XSS und SQL-Injection	
<b>Testergebnis</b>	Fehlerhaft	
<b>Fehlerbeschreibung</b>	Die Benutzerprofil Funktionalität fehlt komplett.	

TABELLE J.26: Akzeptanztest 23

<b>Test 24</b>	<b>Tester:</b> Damian Senn	<b>Datum:</b> 16.05.2019
<b>Kriterium</b>	Security - Gig erfassen	
<b>Erwartetes Ergebnis</b>	Das Gig erfassen Formular ist resistent gegen XSS und SQL-Injection	
<b>Testergebnis</b>	OK	
<b>Fehlerbeschreibung</b>		

TABELLE J.27: Akzeptanztest 24

## J.16 Offene Punkte

Einige Kriterien konnten bisher noch nicht umgesetzt werden, dies ist vor allem auf die bereits erläuterten Probleme (J.13) zurückzuführen.

### J.16.1 Genre Zuordnung für Gigs

Während die Filterung nach Genre bereits funktioniert, ist es noch nicht möglich, einem Gig ein oder mehrere Genres zuzuweisen.

Der Aufwand dies nachzuliefern wird auf circa 8 bis 10 Stunden geschätzt.

### J.16.2 Benutzer Profil

Das Benutzerprofil, in welchem angemeldete Besucher ihre Daten wie E-Mail, Benutzername, Passwort, etc. bearbeiten können, wurde noch nicht umgesetzt. Da das Benutzerprofil nicht ein zentraler Bestandteil der Applikation ist, habe ich mich dazu entschieden, die Umsetzung zu verschieben.

### J.16.3 Passwort-Reset Funktionalität

Benutzer die ihr Passwort vergessen haben, haben derzeit keine Möglichkeit dieses zurückzusetzen. Da die Applikation ohne diese Funktion ihre Grundfunktionalität erfüllen kann, wurde dieses Feature noch nicht umgesetzt.

### J.16.4 Screenshot Tests

Wie bereits in der Studie (D.12) identifiziert, ist das Wallaby Test-Framework leider nicht mit [percy.io](https://percy.io) kompatibel, es ist denkbar eine Integration selber umzusetzen. Da die Umsetzung der Screenshot Tests keine direkter Nutzen für die Applikation bietet, wurde dies noch nicht umgesetzt.

### J.16.5 Expliziter Ort Kontext

Implizit ist in den Konzept Mockups zu erkennen, dass auf der Startseite sowie in der Suche automatisch nach der Stadt, in welcher sich der Besucher befindet, gefiltert wird. Dies wurde im Software-Konzept weiter ausgearbeitet und es wurde definiert, dass die Stadt des Besuchers über eine GeoIP Datenbank zugeordnet wird. Diese Zuordnung von Besucher IP nach Stadt wurde umgesetzt, jedoch werden die Inhalte der Webseite noch nicht implizit danach gefiltert.

Das Kriterium, dass die Suche nach Ort einschränken soll, ist jedoch durch den Suchfilter abgedeckt.

### J.16.6 Mögliche Erweiterungen

Durch die geographischen Koordinaten die wir durch die Google Places API erhalten, ist es denkbar, einen weiteren Suchfilter zu implementieren, der über einen Radius arbeitet. Als Beispiel könnte so nach Konzerten in der Nähe von Bern mit einem Radius von 30 Kilometern gesucht werden.

Dazu habe ich bereits eine Lösung im Internet<sup>16</sup> gefunden, die relativ einfach zu implementieren wäre.

## J.17 Installationsanleitung

### J.17.1 Benötigte Software

#### Server

Auf dem Server wo die Applikation betrieben werden soll müssen die folgenden Software Pakete installiert werden:

- Erlang OTP Version 22.0
- rebar Version 3.10.0
- Elixir Version 1.8.0
- Imagemagick 6.9

Alternativ kann zu einem offiziellen Docker Image<sup>17</sup> gegriffen werden, dieses stellt bereits alle nötigen Software Pakete zur Verfügung.

Bei der Installation des Servers oder Images, muss ausserdem die GeoIP Datenbank installiert werden, diese kann unter der folgenden acrshorturl heruntergeladen werden:

<https://geolite.maxmind.com/download/geoip/database/GeoLite2-City.tar.gz>

Damit die Server Applikation die Datenbank findet, muss die System Environment Variable **GEOIP\_DATABASE\_FILE** gesetzt werden.

#### Testumgebung

Die Testumgebung hat die selben Anforderungen wie der Server.

Damit die Browsertests funktionieren, muss auf dem Zielsystem der **Chromedriver** installiert werden. Chromedriver kann entweder von der offiziellen Webseite<sup>18</sup> heruntergeladen oder über den NPM installiert werden.

Installation mit NPM:

```
1 $ npm install -g chromedriver
```

<sup>16</sup><https://www.movable-type.co.uk/scripts/latlong-db.html>

<sup>17</sup>[https://hub.docker.com/\\_/elixir/](https://hub.docker.com/_/elixir/)

<sup>18</sup><http://chromedriver.chromium.org/>

### J.17.2 Benötigte Dienste

Für den Betrieb der Server Applikation sowie der Testumgebung werden zusätzliche Dienste wie die Datenbank oder Dateiserver benötigt.

Für die Umsetzung wurde die von Phoenix als Standard definierte Datenbanksoftware **PostgreSQL**<sup>19</sup> verwendet.

Für das Sessionmanagement von Besuchern wird der Key-Value Store **redis** verwendet. Es ist denkbar, das Redis in Zukunft auch für Caching von Ressourcen verwendet werden kann.

File-Uploads, vor allem für Bilder, wird mit der **minio** Software verwaltet.

Folgende Dienste sind nötig für den Betrieb der Applikation:

- PostgreSQL Version 11
- Redis 5
- minio (RELEASE.2019-05-02T19-07-09Z)

---

<sup>19</sup><https://www.postgresql.org/>

### J.17.3 Umgebungsvariablen

Name	Beschreibung
AWS_S3_BUCKET	Amazon S3 oder minio Bucket Name
AWS_ACCESS_KEY_ID	Amazon oder minio Access Key
AWS_SECRET_ACCESS_KEY	Amazon oder minio Secret Acces Key
AWS_REGION	Amazon Region («local» für minio)
AWS_S3_SCHEME	Amazon acrshorturl Schema («https» oder «http»)
AWS_S3_HOST	Amazon S3 oder minio Hostname
AWS_S3_PORT	Amazon S3 oder minio Port
GIGPILLAR_WEB_HOST	Hostname der Applikation, z.B. «gigpillar.com»
GIGPILLAR_WEB_SECRET_KEY_BASE	Ein zufällig generierter Wert, wird für die Verschlüsselung der Sessiondaten benötigt.
GIGPILLAR_WEB_DEFAULT_CITY	Stadt die angezeigt werden soll, wenn für den Besucher keine zugeordnet werden kann.
GIGPILLAR_ASSET_HOST	Basis acrshorturl für den File-Upload Server, z.B. «http://localhost:4040»
POSTGRES_USER	Der Datenbank Benutzername
POSTGRES_PASSWORD	Das Datenbank Benutzer Passwort
POSTGRES_DB	Der Datenbank Name
POSTGRES_HOSTNAME	Der Hostname des Datenbank Servers
POSTGRES_POOL_SIZE	Die Anzahl Verbindungen die zur Datenbank gemacht werden sollen.
GEOIP_DATABASE_FILE	Pfad zum GeoIP Datenbank Archiv.
GOOGLE_API_KEY	Der Google API Schlüssel. <sup>20</sup>
REDIS_HOST	Der redis Hostname
REDIS_PORT	Der redis Port

TABELLE J.28: Betrieb: Umgebungsvariablen

<sup>20</sup><https://developers.google.com/places/web-service/get-api-key>



### J.17.4 Initialen Benutzer erstellen

Es wurde ein Skript erstellt um den ersten, initialen Benutzer zu erstellen. Dies sollte vor allem für Testzwecke verwendet werden und nicht auf der produktiven Umgebung.

```
1 $ mix gigpillar.create_user
```

Beispiel einer erfolgreichen Durchführung des Skripts:



```
topaxi@home:~/diplomarbeit-tsbe$ mix gigpillar.create_user
Username: damian
Email: damian.senn@topaxi.codes
Password:
Password (confirm):

14:50:41.487 [debug] QUERY OK db=3.8ms decode=0.9ms queue=0.5ms
INSERT INTO "users" ("email","password_hash","username","inserted_at","u
t") VALUES ($1,$2,$3,$4,$5) RETURNING "id" ["damian.senn@topaxi.codes",
id$v=19$m=131072,t=8,p=4$YdWmJo4F/JA531ZCvJxLGg$Wlc/yTK8IG4N2yiFk/WZxFSf
PcNWB8SSV+Y", "damian", ~N[2019-05-15 12:50:41], ~N[2019-05-15 12:50:41]
topaxi@home:~/diplomarbeit-tsbe$
```

ABBILDUNG J.13: Initialen Benutzer erstellen

### J.17.5 Grundsetup und Start

Zuerst muss ein Secret erstellt werden, dieses wird für die Verschlüsselung der Besucher Session Daten verwendet, dies muss generiert und sicher aufbewahrt werden. Wird das Secret ausgetauscht, werden alle Benutzer die eingeloggt sind automatisch ausgeloggt. Das Secret darf daher **nur** ausgetauscht werden, falls dieses aus Versehen in die Öffentlichkeit gelangt!

```
1 $ mix phx.gen.secret
2 wcR6neHuJjwErwFABZPLUPGj5yo3ciue4dLRNz6EwQIGYhL0T5PexUfyx/zC
```

Unter **keinen** Umständen soll das obige Secret verwendet werden! **Es muss ein neues Secret für den Betrieb generiert werden!**

```
1 $ export GIGPILLAR_SECRET_KEY_BASE=wcR6neHuWFAbZLUPGj5yiue4d
```

Nach dem Erstellen des Secrets müssen die Abhängigkeiten installiert und kompiliert werden.

Herunterladen und kompilieren der Elixir Abhängigkeiten:

```
1 $ mix deps.get --only prod
2 $ MIX_ENV=prod mix compile
```

Herunterladen und kompilieren der JavaScript Abhängigkeiten:

```
1 $ cd apps/gigpillar_web/assets
2 $ yarn --frozen-lockfile
3 $ npx webpack --mode production
4 $ cd -
5 $ mix phx.digest
```

Die Applikation ist nun bereit für den Start im Produktionsmodus, dazu muss nun nur noch das **MIX\_ENV** sowie der **Port** definiert werden:

```
1 $ MIX_ENV=prod PORT=4004 mix phx.server
```

## Anhang K

# Einführung

### K.1 Projektcontrolling

#### K.1.1 Erfüllung der Ziele

Mit der folgenden Tabelle wird aufgezeigt, welche Projektziele erfüllt, und welche Projektziele **nicht** erfüllt wurden. Projektziele die nicht erfüllt wurden, müssen genauer erläutert werden wieso diese nicht erfüllt wurden.

Nr.	Zielbeschreibung	Erfüllt
<b>Produktziele</b>		
1.1	Besucher können im Produkt nach Konzerten suchen	Ja
1.2	Suchresultate können nach Musik-Genre und Ort gefiltert werden	Ja
1.3	Besucher können Details zu einem Konzert ansehen	Ja
1.4	Das Produkt soll ein modernes responsives Design vorweisen	Ja
1.5	Konzerte sollen von Suchmaschinen indexiert werden können	Ja
1.6	Benutzer können sich im Produkt registrieren	Ja
1.7	Benutzer können ihr Passwort nach Verlust neu setzen	NEIN
1.8	Inhalte des Portals sind durch die Benutzer erfassbar und bearbeitbar	Ja
1.9	Kompatibilität mit aktuellem Google Chrome und Mozilla Firefox Browser	Ja
1.10	Konzerte können vom Produkt nach Facebook exportiert werden	Nein
1.11	Ein angemeldeter Benutzer kann vermerken ob er einem Konzert teilnimmt	Nein
1.12	Das Produkt soll sich an die Security Best-Practices von OWASP halten	Ja
<b>Abwicklungsziele</b>		
2.1	Das Projekt soll nach HERMES 5 unter Berücksichtigung der Richtlinien von der TSBE dokumentiert werden	Ja
2.2	Das Produkt muss bis Projektende fertiggestellt, getestet und bereit für die Einführung sein	Ja
2.3	Die Technische-Umsetzung wird durch Damian Senn erstellt	Ja

Nr.	Zielbeschreibung	Erfüllt
2.4	Die Kommunikation zwischen Experten und Diplomanden erfolgt wie im Projektauftrag <b>E.7.2</b> beschrieben.	Ja
2.5	Das Projekt muss bis Ende Mai 2019 abgeschlossen sein	Ja

TABELLE K.1: Controlling: Ziele

**Nicht erfüllte Ziele**

Das Ziel 1.7 konnte nicht erfüllt werden. In Retrospektive, hätte dieses Feature kein MUSS Kriterium sein müssen. Die verfügbare Zeit der Realisierung wurde in wichtigere Kernkomponenten des Produktes investiert.

## K.1.2 Erfüllung der Anforderungen

Feature	Titel		Nr.	Kriterium	Erfüllt
Suche	Suche nach	Konzertname	1.1	Listet alle Konzerte die Wörter der Suche im Konzertnamen beinhalten	Ja
	Suche nach	Konzertlocation	1.2	Schränkt die Such-Resultate nach gegebener Konzertlocation ein	Ja
	Suche nach	Ort	1.2	Schränkt die Such-Resultate nach gegebenem Ort ein	Ja
	Suche nach	Genre	1.2	Schränkt die Such-Resultate nach gegebenem Musik-Genre ein	Ja
Design	Desktop		2.1	Alle Ansichten haben eine Desktop-Optimierte Variante	Ja
	Tablet		2.2	Alle Ansichten haben eine Tablet-Optimierte Variante	Ja
	Mobile		2.3	Alle Ansichten haben eine Mobile-Optimierte Variante	Ja
	Browser Kompatibilität		2.4	Alle Ansichten müssen in aktuellem Google Chrome und Mozilla Firefox dem Grundlayout folgen	Ja
SEO	Indexierbarkeit		3.1	Das Produkt ist von Suchmaschinen indexierbar	Ja
	Linked Data		3.2	Konzert Detailseiten sind mit dem Event-Schema <sup>1</sup> ausgestattet	Ja
Benutzer	Registrierung		4.1	Besucher können sich einen Benutzer registrieren, Benutzernamen und E-Mail Adressen müssen einzigartig sein	Ja
	Passwort-Vergessen		4.2	Benutzer können sich einen Passwort-Reset Link anfordern	NEIN
	Social		4.3	Benutzer können auf Konzerten vermerken ob sie teilnehmen oder nicht	Nein

<sup>1</sup><https://schema.org/MusicEvent>

Feature	Titel	Nr.	Kriterium	Erfüllt
Erfassung	Artist	5.1	Benutzer können Artisten mit einem Genre erfassen	NEIN
	Location	5.2	Benutzer können eine Konzertlocation mit Ort/Strasse erfassen	NEIN
	Konzert	5.3	Benutzer können ein Konzert mit Konzertlocation und Artisten erfassen	Ja
	Facebook	5.4	Benutzer können ein Konzert in ein Facebook-Event exportieren	Nein
Security	SQL-Injection	6.1	Das Produkt soll resistent gegen SQL-Injection sein	Ja
	HTML-Injection	6.2	Das Produkt soll resistent gegen HTML-Injection / XSS sein	Ja
	Passwort encryption	6.3	Passwörter von Benutzer müssen mit einem sicheren Verfahren gespeichert werden	Ja
	Session	6.4	Session-Cookies dürfen nicht durch JavaScript ausgelesen werden	Ja
Performance	Ladezeit	7.1	Die Seitenansichten dürfen nicht länger als 6 Sekunden auf einem 3G Netz laden	Ja
Sonstiges	User Tracking	8.1	Benutzerverhalten soll analysiert und nachvollziehbar sein.	Nein

TABELLE K.2: Anforderungskatalog

### Nicht erfüllte Anforderungen

Wie bereits bei den Zielen ist auch bei den Anforderungen die Passwort-Reset (4.2) Funktionalität nicht erfüllt. Zusätzlich sind die Anforderungen 5.1 und 5.2 nicht erfüllt. Die beiden Anforderungen wurden nicht umgesetzt, da in der Konzeptphase entschieden wurde, dass die Konzertlocations nur über die Google Places API ausgewählt werden können. Somit gibt es für die Anforderung 5.2 keine Umsetzung. Auch die Anforderung 5.1 wurde in der Konzeptphase angepasst, so wird nicht einem Artisten das Genre sondern einem Gig die Genres angehängt. Allerdings wurde das Feature aus Zeitgründen noch nicht umgesetzt.

## Anhang L

# Arbeitsjournal

### L.1 Sonntag 3. März

2h:

- Vorbereitung Kick-off
- Abgrenzung erweitern
- Grobe Anforderungen
- Auflistung möglicher Variantenentscheide
- TODO ergänzt

### L.2 Dienstag 5. März

2h:

- Vorbereitung Kick-off

### L.3 Mittwoch 6. März

3h:

- Vorbereitung Sitzungszimmer
- Kick-off Meeting

4h:

- An Projektauftrag arbeiten - Auftraggeber geändert nach Empfehlung von Marc Aeby

### L.4 Samstag 9. März

2h:

- An Studie/Pflichtenheft arbeiten

### L.5 Dienstag 12. März

2h:

- PDF Generierung und Ordnerstruktur angepasst

**L.6 Samstag 16. März**

3h:

- Projektplan von gantt nach ods migrieren

**L.7 Dienstag 19. März**

0.5h:

- Projektplan in Berichtanhang angehängt

**L.8 Mittwoch 27. März**

1.5h:

- Projektplan an korrekte HERMES 5 Struktur angepasst
- Titelblatt von Ilias hinzugefügt
- Termin am 12.04.2019 mit Joshua Schär für einen Screendesign-Workshop abgemacht

**L.9 Sonntag 31. März**

5h:

- Projektziele erweitert
- Anforderungskatalog erweitert

**L.10 Sonntag 31. März**

8h:

- Definitive Projektziele definiert
- Anforderungskatalog fertig gestellt
- Gesamte Berichtstruktur ausgelegt, Ziele und Abgrenzungen in Bericht hinterlegt und referenziert

**L.11 Freitag 5. April**

2h:

- An Studie weiter gearbeitet
- Variantenkriterien definiert

**L.12 Samstag 6. April**

4h:

- An Studie weiter gearbeitet
- Variantenbeschreibungen erstellt
- Variantenbewertungen



## **L.13 Mittwoch 10. April**

2h:

- Brainstorming für Portalnamen

## **L.14 Freitag 12. April**

12h:

- Screens definiert
- Mit Joshua Schär an Mockups gearbeitet

## **L.15 Samstag 13. April**

12h:

- Risiko Management
- Variantenbeschreibungen angepasst/erweitert
- Projektauftrag

## **L.16 Sonntag 14. April**

8h:

- Wirtschaftlichkeit
- Projektauftrag

## **L.17 Montag 15. April**

2h:

- Initiales Datenbankschema basierend auf Mockups

## **L.18 Mittwoch 17. April**

0.5h:

- Zwischen-Meeting in Planung in richtiger KW eingetragen.
- Biweekly Reports in Anhang eingefügt.

## **L.19 Freitag 19. April**

10h:

- Alle Mockups detailreicher beschrieben.
- Kleinere Mockup Anpassungen.
- Software Konzept mit Datenfluss-Charts beschrieben

**L.20 Sonntag 21. April**

10h:

- Mockup Beschreibungen
- Datenbankschema
- Mit Testkonzept begonnen

**L.21 Montag 22. April**

2h:

- An Testkonzept gearbeitet
- An Datenbankschema gearbeitet

**L.22 Dienstag 23. April**

6h:

- Zwischenmeeting Präsentation erstellt
- An Datenbankschema gearbeitet

**L.23 Mittwoch 24. April**

3h:

- Zwischenmeeting
- Initiales Projektsetup erstellt

**L.24 Mittwoch 1. Mai**

14h:

- Begin an HTML-Prototypen

**L.25 Sonntag 5. Mai**

11h:

- An HTML-Prototypen gearbeitet
- Initiale Login Implementation

**L.26 Montag 6. Mai**

12h:

- Script zum erstellen von Benutzern
- Registrierungsprozess implementiert
- Session Management eingerichtet
- Berechtigungssystem definiert
- Mit Erfassungsformular für Gigs begonnen

## L.27 Dienstag 7. Mai

1h:

- Unit-Tests für Gigs

## L.28 Mittwoch 8. Mai

10h:

- An Gig Erfassungsformular gearbeitet, diverse Problematiken mit Phoenix Formular sowie Datenbankbeziehungen aufgetreten.
- Google Places API angebunden für Location Autocomplete
- Generische Suchbox in HTML/JavaScript implementiert

## L.29 Donnerstag 9. Mai

11h:

- JavaScript, HTML und CSS Optimierungsplugins installiert
- Location Autocomplete fertiggestellt
- Künstlerfeld für Gigs implementiert
- Fehlermeldungen für invalide Felder eingefügt
- Bildinputfeld implementiert
- Debugging von diversen Probleme im Zusammenhang mit Datenbankrelationen für das Gig erfassen/updaten Formular

## L.30 Freitag 10. Mai

1h:

- Dropdown Element implementiert

## L.31 Samstag 11. Mai

13h:

- Dropdown Element implementiert
- Debugging von diversen Probleme im Zusammenhang mit Datenbankrelationen für das Gig erfassen/updaten Formular

## L.32 Sonntag 12. Mai

7h:

- Bildupload für Gigs
- Begin der Implementation der Suche

### **L.33 Montag 13. Mai**

10h:

- Bilder resizing für Gigs
- Besucher Location mit GeoIP auflösen
- Diverse kleinere Bugfixes
- Gig-Detail Ansicht implementiert

### **L.34 Dienstag 14. Mai**

14h:

- Browser-Tests für Gig erstellen, Registrierung, Login und Suche
- Location Autocomplete Sessiontoken hinzugefügt
- Realisierungsdokumentation

### **L.35 Mittwoch 15. Mai**

15h:

- Realisierungsdokumentation
- Bericht schreiben

## Anhang M

# Biweekly Reports

### M.1 Kalenderwoche 11-12

Hallo Sandro, hallo Severin

Kurzes Update was in meiner Diplomarbeit in den letzten zwei Wochen gelaufen ist:

- Ein Projektplan mit mehr Detail wurde erstellt, Details für die Realisierungsphase werden sich in der Konzeptphase noch konkretisieren.
- Das Organigramm habe ich angepasst, nach dem ich mit Marc Aeby die Auftragsgeber/Projektleiter Situation abgeklärt hatte. Marc meinte, dass der Auftragsgeber nicht gleichzeitig Projektleiter sein darf. Somit habe ich wie mit Severin in Person abgemacht ihn zum neuen Auftragsgeber ernannt.
- Der Projektauftrag wurde grob strukturiert und mit dem Organigramm, Abgrenzungen, Terminplan abgefüllt. Der Projektauftrag wird im Laufe der Studie weiter ergänzt.
- Ich habe den Anforderungskatalog angefangen und Muss/Kann Kriterien definiert.
- Die Nutzerwertanalyse für den Technologie-Einsatz habe ich letzte Woche angefangen.

Weiteres Vorgehen für die nächsten zwei Wochen habe ich wie folgt geplant:

- Die Nutzwertanalyse wird fertiggestellt.
- Ein Variantenentscheid wird gefällt.
- Es wird eine Risikoanalyse erstellt.
- Aus dem obigen Output wird der Projektauftrag fertig gestellt.
- Der aktuelle Stand wird in ein PDF generiert und euch zugeschickt.

Meine Dokumente sind derzeit noch in einfachen Text-Files abgelegt, ich werde euch einen aktuellen Stand als PDF spätestens beim nächsten Report mitschicken.

Bei Fragen oder Unklarheiten stehe ich euch gerne zur Verfügung.

Gruss  
Damian

## M.2 Kalenderwoche 13-14

Hallo Sandro, hallo Severin

Kurzes Update was in meiner Diplomarbeit in den letzten zwei Wochen gelaufen ist:

- Ich habe den Projektplan an die korrekte HERMES 5 Struktur angepasst.
- Mit Joshua habe ich einen Termin am 12.04. abgemacht um erste Screendesigns zu erstellen.
- Die Projektziele wurden ergänzt und detaillierter beschrieben.
- Der Anforderungskatalog wurde fertig gestellt.

Aus privaten und gesundheitlichen Gründen konnte ich die letzten zwei Wochen leider nicht viel am Projekt arbeiten. Ich bin aber zuversichtlich, dass ich durch die Schulferien und dem Zürcher-Feiertag am Montag dieses Wochenende mein Defizit wieder aufarbeiten kann. Somit sieht das weitere Vorgehen leider ähnlich wie im letzten Report aus:

- Die Nutzwertanalyse wird fertiggestellt.
- Ein Variantenentscheid wird gefällt.
- Es wird eine Risikoanalyse erstellt.
- Aus dem obigen Output wird der Projektauftrag fertig gestellt.

Anbei habe ich euch den momentanen Stand meiner Dokumentation angehängt. Die Inhalte sind hauptsächlich im Anhang zu finden, da ich den Teil im Bericht ausfülle sobald die entsprechenden Anhänge fertiggestellt sind.

Bei Fragen oder Unklarheiten stehe ich euch gerne zur Verfügung.

Gruss  
Damian

## M.3 Kalenderwoche 15-16

Hallo Sandro, hallo Severin

Kurzes Update was in meiner Diplomarbeit in den letzten zwei Wochen gelaufen ist:

- Die Nutzwertanalysen wurde fertiggestellt.
- Die Variantenentscheide wurden gefällt.
- Die Risikoanalyse wurde fertiggestellt.
- Die Wirtschaftlichkeit wurde berechnet.
- Somit ist der Projektauftrag fertiggestellt.
- Der Produktname ist «Gigpillar».
- Es wurden Mockups für einige Screens zusammen mit Joshua erstellt.
- Basierend auf den Mockups habe ich ein erstes grobes Datenbankschema erstellt.

Das weitere Vorgehen sieht folgendermassen aus:

- Die Screendesigns im Konzept detailliert beschreiben.
- Die Software-Architektur erstellen.
- Ein Test-Konzept erstellen.
- Die Präsentation für das Zwischen-Meeting vorbereiten.
- Begin der Realisierungsphase.

Anbei habe ich euch den momentanen Stand meiner Dokumentation angehängt. Ihr findet die Studie im Anhang **D** und den Projektauftrag im Anhang **E**.

Ich freue mich euch den Stand des Projekts nächste Woche am 24.04.2019 um 09:00 an der Belpstrasse 37 vorzustellen.

Sandro ich habe dir unseren Besucherparkplatz Nummer 39 von 08:45 bis 12:00 reserviert, du findest den Anfahrtsplan im Anhang.

Bei Fragen oder Unklarheiten stehe ich euch gerne zur Verfügung.

Gruss  
Damian

## M.4 Kalenderwoche 17-19

Hallo Sandro, hallo Severin

Kurzes Update was in meiner Diplomarbeit in den letzten drei Wochen gelaufen ist:

- Die Screendesigns wurden im Konzept beschrieben
- Das Software-Konzept wurde erarbeitet
- Das Test-Konzept wurde erstellt
- Die Screens wurden in HTML umgesetzt. Den Prototypen ohne Funktionalität könnt ihr euch unter <https://topaxi.ch/gigpillar/> ansehen.
- Das Datenbankschema wurde im Backend umgesetzt
- Der Login/Registrierungsprozess wurde umgesetzt
- Das Erstellen und Bearbeiten von Konzerten ist möglich

Das weitere Vorgehen sieht folgendermassen aus:

- Umsetzung der Suche
- Umsetzung des Standortwechsels auf der Startseite
- Passwort-Reset Funktionalität
- Fertigstellung des Berichts
- Drucken und Binden des Berichts
- Vorbereitung Präsentation / Abschluss-Meeting

Ich werde euch den fertigen Bericht Ende nächste Woche am 17.05.2019 zukommen lassen.

Bei Fragen oder Unklarheiten stehe ich euch gerne zur Verfügung.

Gruss  
Damian



# Abkürzungsverzeichnis

**API** Application Programming Interface. 1, 23, 26, 28, 33, 85, 103, 118, 121, 122, 126, 131, 146, 148, 154

**AWS** Amazon Web Services. 1

**CPC** Cost Per Click. 1

**CPM** Cost Per Mile. 1

**CSS** Cascading Syle Sheets. 1, 113

**ERD** Entity Relationship Diagram. 1

**HTML** Hypertext Markup Language. 1, 12, 13, 49, 51

**HTTP** Hypertext Transfer Protocol. 1, 25, 90, 122

**JSON** JavaScript Object Notation. 1

**JSON-LD** JavaScript Object Notation for Linked Data. 1

**NPM** Node Package Manager. 1, 132, 146

**OTP** Open Telecom Platform. 1, 146

**OWASP** Open Web Application Security Project. 1, 2, 29, 60, 151

**REST** Representational State Transfer. 1, 131

**SASS** Syntactically Awesome Stylesheets. 1, 107

**SEO** Search Engine Optimization. 1, viii, 10, 31, 46, 48, 50, 64, 127, 153

**SMTP** Simple Mail Transfer Protocol. 1

**SQL** Structured Query Language. 1, 129

**SSR** Server-Side Rendered. *Glossar*: Server-Side Rendered, 1, 12, 50, 66

**TSBE** Telematikschule Bern. 1, 2, 29, 60, 151

**URL** Unified Resource Locator. 1

**UUID** Universally Unique Identifier. 1, 126

**XSS** Cross-site Scripting. 1, 10, 31, 46, 64, 101, 102, 143, 144, 153



# Glossar

**Argon2** Argon2 ist ein sicherer Passwort-Hashing Algorithmus, siehe <https://de.wikipedia.org/wiki/Argon2>. 28, 118

**Library** In der Software-Entwicklung wird eine Sammlung an wiederverwendbaren Funktionen «Library» genannt. 12, 13, 14, 25, 49, 51, 54, 66, 90, 120, 121, 126, 130, 131

**Microdata** Ergänzungen im HTML um Suchmaschinen zusätzliche Informationen bereit zu stellen. viii, 127

**Responsive Design** Ein Design-Prinzip, bei dem jegliche Bildschirmgrößen beachtet werden. Das Interface passt sich dynamisch der verfügbaren Bildschirmgröße an. 29

**Server-Side Rendered** HTML wird auf dem Server vorbereitet und dem Browser zur Verfügung gestellt. Modernere Applikationen senden mehr JavaScript zum Browser und laden Daten vom Server im JSON Format. 12