Konzertkalender

Damian Senn

TSBE

DIPLOMARBEIT

Konzertkalender

Author:
Damian Senn

Experten:
Sandro Bertolino
Severin Räz

Eine Diplomarbeit für den Abschluss Dipl. Techniker Informatik

15. Mai 2019

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: _____

TSBE

Management Summary

Dipl. Techniker Informatik

Konzertkalender

von Damian Senn

Inhalt für Management Summary folgt hier…

Danksagung

TODO

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

TSBE	**T**elematik**s**chule **Be**rn
HTML	**H**yper**t**ext **M**arkup **L**anguage
CSS	**C**ascading **S**yle **S**heets
SASS	**S**yntactically **A**wesome **S**tyle**s**heets
HTTP	**H**yper**t**ext **T**ransfer **P**rotocol
SEO	**S**earch **E**ngine **O**ptimization
OWASP	**O**pen **W**eb **A**pplication **S**ecurity **P**roject
XSS	**C**r**oss**-**s**ite **s**cripting
SSR	**S**erver **S**ide **R**endered
CPC	**C**ost **P**er **C**lick
CPM	**C**ost **P**er **M**ile
SMTP	**S**imple **M**ail **T**ransfer **P**rotocol
ERD	**E**ntity **R**elationship **D**iagram
API	**A**pplication **P**rogramming **I**nterface
JSON	**J**ava**S**cript **O**bject **N**otation
URL	**U**nified **R**esource **L**ocator
AWS	**A**mazon **W**eb **S**ervices
SQL	**S**tructured **Q**uery **L**anguage

For/Dedicated to/To my…

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 Verfügungs 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
	Produktziele	
1.1	Besucher können im Produkt nach Konzerten suchen	**Muss**
1.2	Suchresultate können nach Musik-Genre und Ort gefiltert werden	**Muss**
1.3	Besucher können Details zu einem Konzert ansehen	**Muss**
1.4	Das Produkt soll ein modernes responsives Design vorweisen	**Muss**
1.5	Konzerte sollen von Suchmaschinen indexiert werden können	**Muss**
1.6	Benutzer können isch im Produkt registrieren	**Muss**
1.7	Benutzer können ihr Passwort nach Verlust neu setzen	**Muss**
1.8	Inhalte des Portals sind durch die Benutzer erfassbar und bearbeitbar	**Muss**
1.9	Kompatibilität mit aktuellem Google Chrome und Mozilla Firefox Browser	**Muss**
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 1 halten	**Muss**
	Abwicklungsziele	
2.1	Das Projekt soll nach HERMES 5 unter Berücksichtigung der Richtlinien von der TSBE dokumentiert werden	**Muss**
2.2	Das Produkt muss bis Projektende fertiggestellt, getestet und bereit für die Einführung sein	**Muss**
2.3	Die Technische-Umsetzung wird durch Damian Senn erstellt	**Muss**
2.4	Die Kommunikation zwischen Experten und Diplomanden erfolgt wie im Projektauftrag E.7.2 beschrieben.	**Muss**
2.5	Das Projekt muss bis Ende Mai 2019 abgeschlossen sein	**Muss**

TABELLE 1.1: Ziele

1.3 Projektorganisation

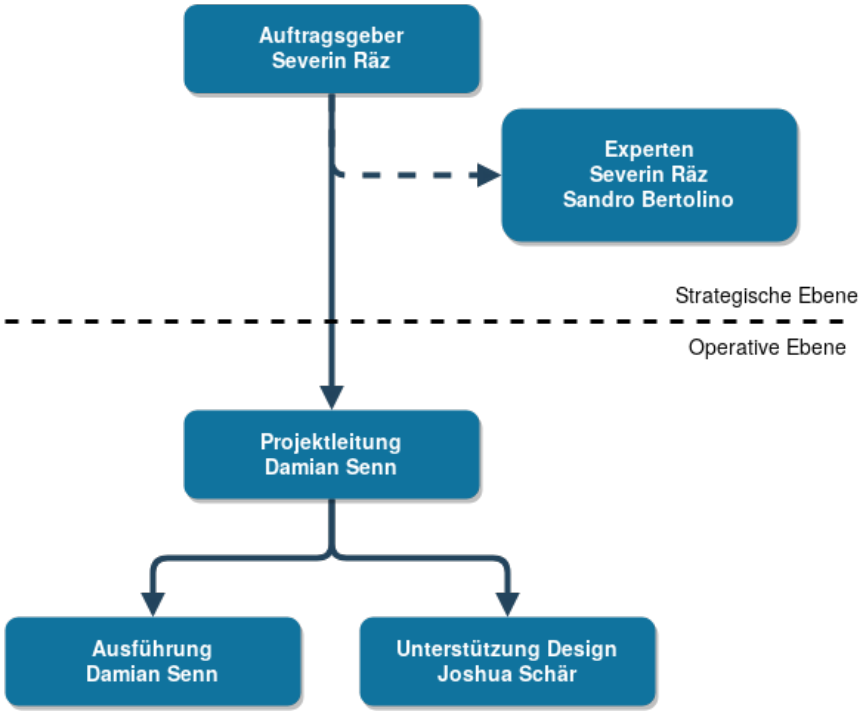
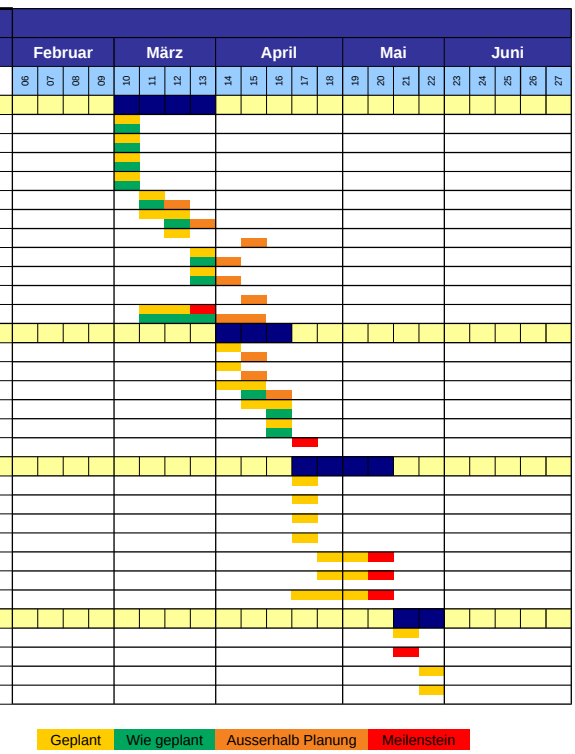


ABBILDUNG 1.1: Organigram

1.4 Ausgefüllter Projektplan

Projektplan: Konzertkalender

Aktivität		Dauer [h]			Status	Wer
		Soll	Ist	Abw.		
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	36	-30		
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	in arbeit	DS,JS
3.4	Software Architektur	12	6	-6	in arbeit	DS
3.5	Test Konzept	12	8	-4	in arbeit	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:		**286**	**89**	**-197**		



286	89	197
Total Soll:	Bereits benötigt:	Restliche Stunden:

Legende

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

planung.ods

1.5 Lieferergebnisse

- Studie (Anhang D)
- Projektauftrag (Anhang E)
- Konzept (Anhang H)

1.6 Ressourcenplan

1.7 Risiken

1.8 Abgrenzungen

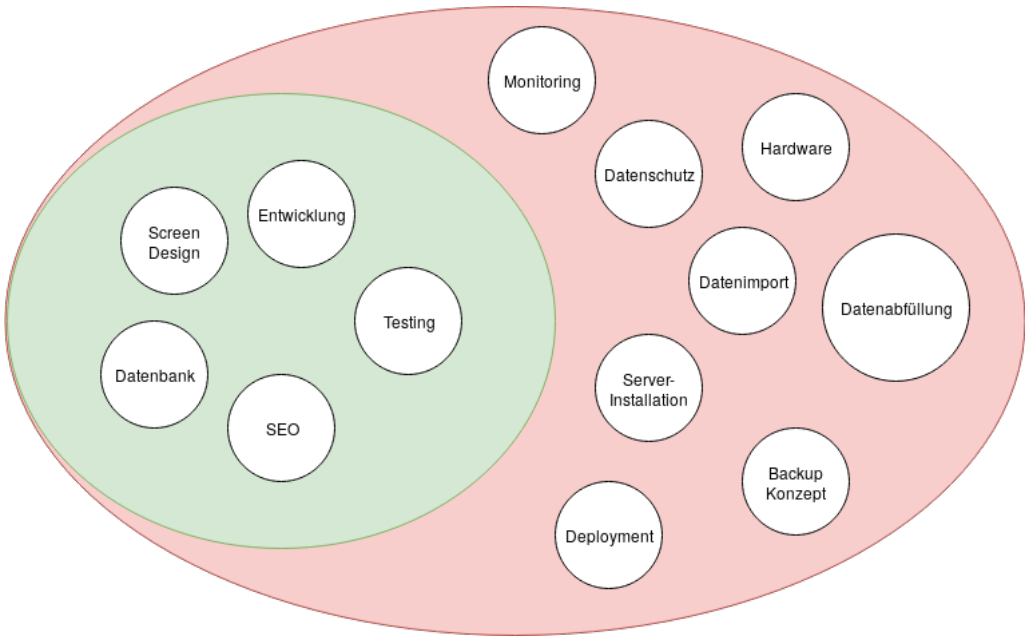


ABBILDUNG 1.2: Abgrenzungen

Die detaillierten Erklärung zu den Abgrenzungen sind im Projektauftrag E.8 zu finden.

1.9 Studie

1.9.1 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 1.2: Informationsbeschaffung

1.9.2 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	**Muss**
	Suche nach Konzertlocation	1.2	Schränkt die Such-Resultate nach gegebener Konzertlocation ein	1.2	**Muss**
	Suche nach Ort	1.2	Schränkt die Such-Resultate nach gegebenem Ort ein	1.2	**Muss**
	Suche nach Genre	1.2	Schränkt die Such-Resultate nach gegebenem Musik-Genre ein	1.2	**Muss**
Design	Desktop	2.1	Alle Ansichten haben eine Desktop-Optimierte Variante	1.4	**Muss**
	Tablet	2.2	Alle Ansichten haben eine Tablet-Optimierte Variante	1.4	**Muss**
	Mobile	2.3	Alle Ansichten haben eine Mobile-Optimierte Variante	1.4	**Muss**
	Browser Kompatibilität	2.4	Alle Ansichten müssen in aktuellem Google Chrome und Mozilla Firefox dem Grundlayout folgen	1.9	**Muss**
SEO	Indexierbarkeit	3.1	Das Produkt ist von Suchmaschinen indexierbar	1.5	**Muss**
	Linked Data	3.2	Konzert Detailseiten sind mit dem Event-Schema[1] ausgestattet	1.5	**Muss**
Benutzer	Registrierung	4.1	Besucher können sich einen Benutzer registrieren, Benutzernamen und E-Mail Adressen müssen einzigartig sein	1.6	**Muss**
	Passwort-Vergessen	4.2	Benutzer können sich einen Passwort-Reset Link anfordern	1.7	**Muss**
	Social	4.3	Benutzer können auf Konzerten vermerken ob sie Teilnehmen oder nicht	1.11	Kann

[1] https://schema.org/MusicEvent

Feature	Titel	Nr.	Kriterium	Ziel	Muss
Erfassung	Artist	5.1	Benutzer können Artisten mit einem Genre erfassen	1.8	**Muss**
	Location	5.2	Benutzer können eine Konzertlocation mit Ort/Strasse erfassen	1.8	**Muss**
	Konzert	5.3	Benutzer können ein Konzert mit Konzertlocation und Artisten erfassen	1.8	**Muss**
	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	**Muss**
	HTML-Injection	6.2	Das Produkt soll resistent gegen HTML-Injection / XSS sein	1.12	**Muss**
	Passwort encryption	6.3	Passwörter von Benutzer müssen mit einem sicheren Verfahren gespeichert werden	1.12	**Muss**
	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		**Muss**
Sonstiges	User Tracking	8.1	Benutzerverhalten soll analysiert und nachvollziehbar sein.		Kann

TABELLE 1.3: Anforderungskatalog

1.9.3 Mögliche Varianten

1.9.4 Evaluation Varianten

1.9.5 Entscheid Varianten

1.10 Wirtschaftlichkeit

1.10.1 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	64	5'400.- CHF
Total:	286	42'900.- CHF

TABELLE 1.4: Projektkosten

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

Kostenstelle	**Jährliche Kosten**
Software	Keine
.com Domain	20.- CHF
Hosting	1'800.- CHF
Total:	1'820.- CHF

TABELLE 1.5: 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.

1.10.2 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».

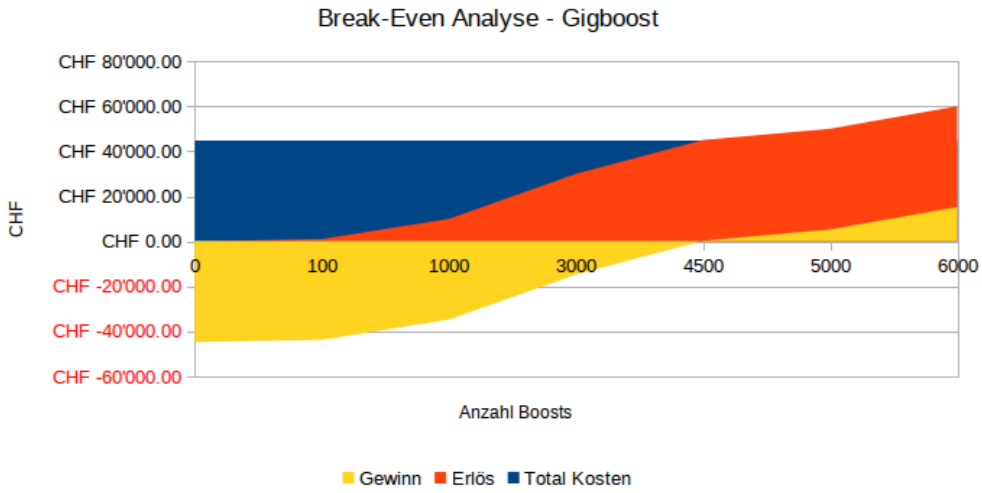


ABBILDUNG 1.3: Break-Even Analyse - Gigboost

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 **40'0000 Besucher**[2] erhalten wir folgendes Bild:

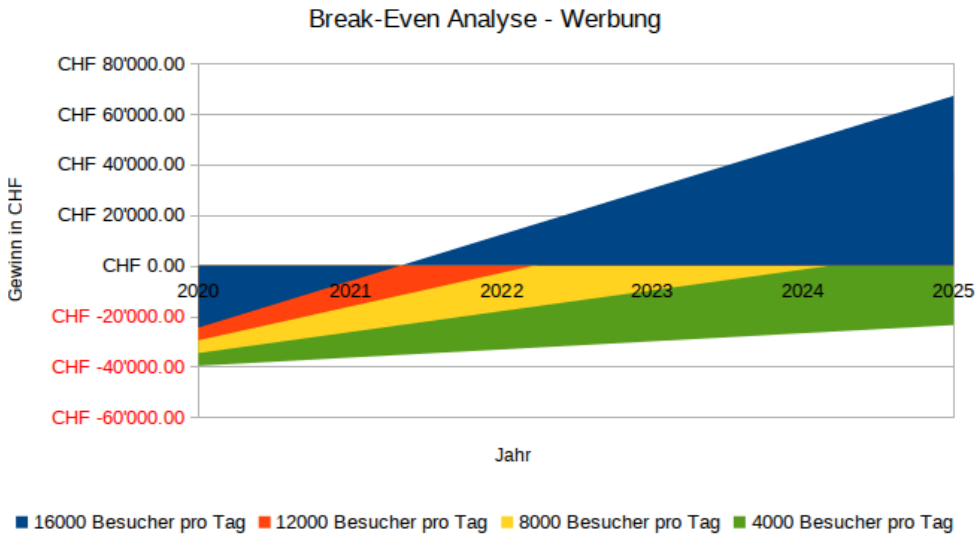


ABBILDUNG 1.4: Break-Even Analyse - Werbung

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 1.6: 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 Beucher pro Tag schon im dritten Jahr.

[2] 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 Erkentnissen, 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 Design

2.2 Software

2.3 Testing

2.4 Fazit

2.4.1 Probleme

2.4.2 Machbarkeit

2.4.3 Wirtschaftlichkeit

2.4.4 Erweiterbarkeit

2.4.5 Projektplan

Kapitel 3

Realisierung

3.1 Umsetzung

3.2 Tests

3.3 Auswertung

Kapitel 4

Einführung

4.1 Projektcontrolling

4.2 Wirtschaftlichkeit

Kapitel 5

Schlussbetrachtung

PROJEKTINITIALISIERUNGSAUFTRAG

WEBBASIERTER KONZERTKALENDER

Auftraggeber: Damian Senn

Projektleiter: Damian Senn

Autor: Damian Senn

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 Verfügungs 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.

Damian Senn | Hirslandersteig 3 | 8032 Zürich damian.senn@gmail.com | Mobile 079 672 02 97

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.

Damian Senn | Hirslandersteig 3 | 8032 Zürich damian.senn@gmail.com | Mobile 079 672 02 97

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.

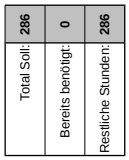
Gruss
Damian

Anhang C

Terminplan

Projektplan: Konzertkalender

Aktivität		Dauer [h]			Status	Wer
		Soll	Ist	Abw.		
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:	**286**	**0**	**-286**		



Legende

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

planung-empty.ods

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	**Muss**
	Suche nach Konzertlocation	1.2	Schränkt die Such-Resultate nach gegebener Konzertlocation ein	1.2	**Muss**
	Suche nach Ort	1.2	Schränkt die Such-Resultate nach gegebenem Ort ein	1.2	**Muss**
	Suche nach Genre	1.2	Schränkt die Such-Resultate nach gegebenem Musik-Genre ein	1.2	**Muss**
Design	Desktop	2.1	Alle Ansichten haben eine Desktop-Optimierte Variante	1.4	**Muss**
	Tablet	2.2	Alle Ansichten haben eine Tablet-Optimierte Variante	1.4	**Muss**
	Mobile	2.3	Alle Ansichten haben eine Mobile-Optimierte Variante	1.4	**Muss**
	Browser Kompatibilität	2.4	Alle Ansichten müssen in aktuellem Google Chrome und Mozilla Firefox dem Grundlayout folgen	1.9	**Muss**
SEO	Indexierbarkeit	3.1	Das Produkt ist von Suchmaschinen indexierbar	1.5	**Muss**
	Linked Data	3.2	Konzert Detailseiten sind mit dem Event-Schema[1] ausgestattet	1.5	**Muss**
Benutzer	Registrierung	4.1	Besucher können sich einen Benutzer registrieren, Benutzernamen und E-Mail Adressen müssen einzigartig sein	1.6	**Muss**
	Passwort-Vergessen	4.2	Benutzer können sich einen Passwort-Reset Link anfordern	1.7	**Muss**
	Social	4.3	Benutzer können auf Konzerten vermerken ob sie Teilnehmen oder nicht	1.11	Kann

[1] https://schema.org/MusicEvent

Feature	Titel	Nr.	Kriterium	Ziel	Muss
Erfassung	Artist	5.1	Benutzer können Artisten mit einem Genre erfassen	1.8	**Muss**
	Location	5.2	Benutzer können eine Konzertlocation mit Ort/Strasse erfassen	1.8	**Muss**
	Konzert	5.3	Benutzer können ein Konzert mit Konzertlocation und Artisten erfassen	1.8	**Muss**
	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	**Muss**
	HTML-Injection	6.2	Das Produkt soll resistent gegen HTML-Injection / XSS sein	1.12	**Muss**
	Passwort encryption	6.3	Passwörter von Benutzer müssen mit einem sicheren Verfahren gespeichert werden	1.12	**Muss**
	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		**Muss**
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, das 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 beinträchtigt 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 **S**erver**s**ide **R**endering 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

Bewertung x Gewichtung = Punktzahl

Kriterium	**Gewichtung**	**Variante: React**		**Variante: Next.js**		**Variante: SSR**	
		Bewertung	Punkte	Bewertung	Punkte	Bewertung	Punkte
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
Total:		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 Server-Side Rendered Webseite favorisierend.

Es ist durchaus vorstellbar, dass in einem zweiten Schritt, nach diesem Projekt, die Server-Side Rendered 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 beinträchtigt werden. Vorallem auf dem Server ist wichtig, dass die Businesslogik gut abdeckend gestestet 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

Bewertung x Gewichtung = Punktzahl

Kriterium	**Gewichtung**	**Variante: koa.js**		**Variante: Phoenix**		**Variante: Next.js**	
		Bewertung	Punkte	Bewertung	Punkte	Bewertung	Punkte
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
Total:		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 Kombinierung 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 Kanditat 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

Bewertung x Gewichtung = 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
Total:		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	64	5'400.- CHF
Total:	286	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
Total:	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».

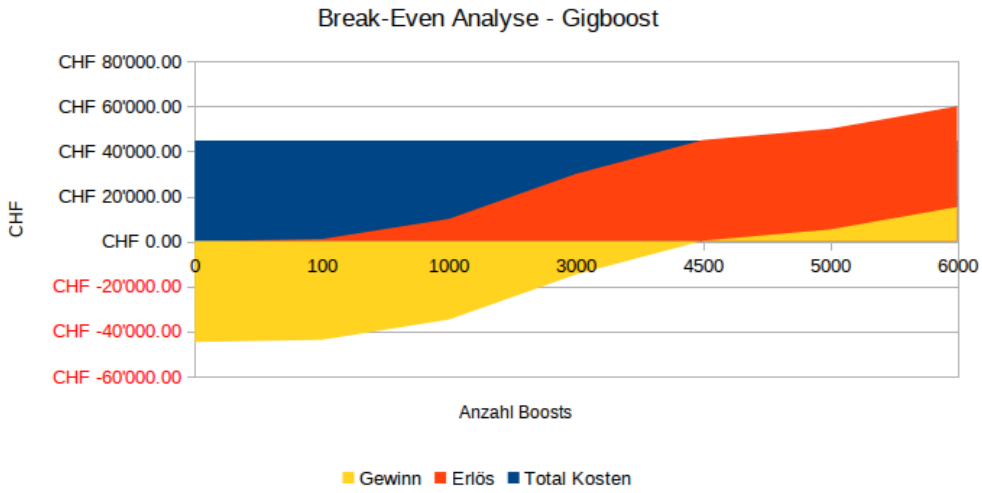


ABBILDUNG D.1: Break-Even Analyse - Gigboost

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 **40'0000 Besucher**[2] erhalten wir folgendes Bild:

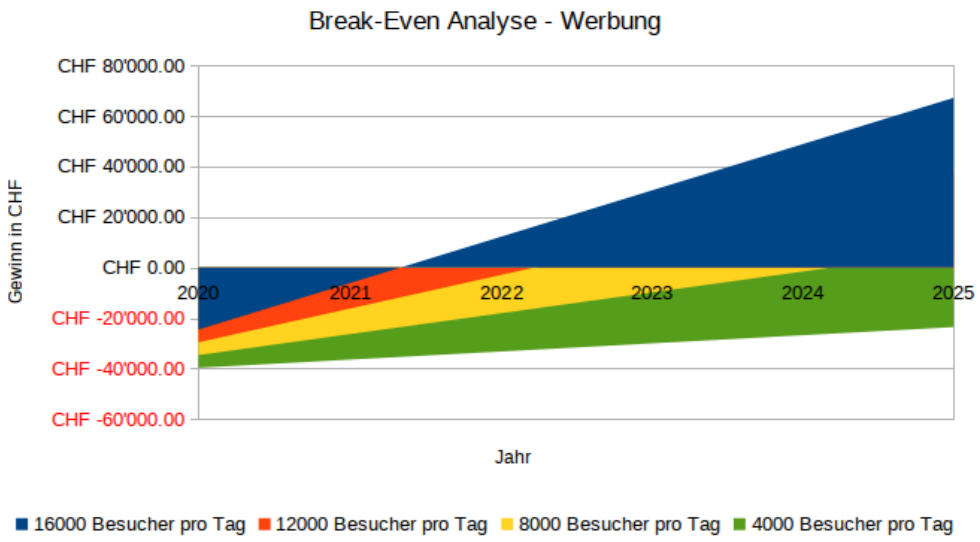


ABBILDUNG D.2: Break-Even Analyse - Werbung

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 Beucher pro Tag schon im dritten Jahr.

[2] 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 Verfügungs 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
	Produktziele	
1.1	Besucher können im Produkt nach Konzerten suchen	**Muss**
1.2	Suchresultate können nach Musik-Genre und Ort gefiltert werden	**Muss**
1.3	Besucher können Details zu einem Konzert ansehen	**Muss**
1.4	Das Produkt soll ein modernes responsives Design vorweisen	**Muss**
1.5	Konzerte sollen von Suchmaschinen indexiert werden können	**Muss**
1.6	Benutzer können isch im Produkt registrieren	**Muss**
1.7	Benutzer können ihr Passwort nach Verlust neu setzen	**Muss**
1.8	Inhalte des Portals sind durch die Benutzer erfassbar und bearbeitbar	**Muss**
1.9	Kompatibilität mit aktuellem Google Chrome und Mozilla Firefox Browser	**Muss**
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 1 halten	**Muss**
	Abwicklungsziele	
2.1	Das Projekt soll nach HERMES 5 unter Berücksichtigung der Richtlinien von der TSBE dokumentiert werden	**Muss**
2.2	Das Produkt muss bis Projektende fertiggestellt, getestet und bereit für die Einführung sein	**Muss**
2.3	Die Technische-Umsetzung wird durch Damian Senn erstellt	**Muss**
2.4	Die Kommunikation zwischen Experten und Diplomanden erfolgt wie im Projektauftrag E.7.2 beschrieben.	**Muss**
2.5	Das Projekt muss bis Ende Mai 2019 abgeschlossen sein	**Muss**

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 verwendet, angepasst auf das Projekt.
- Sämtliche Projekt-Dokumente sowie Programmcode wird regelmässig ins private Github Repository[1] geladen.

[1] https://github.com/topaxi/diplomarbeit-tsbe

E.5 Terminplan

Nachfolgend ist der grobe Terminplan für die geplanten Phasen. Im Anhang E.5 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:	286

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 Räz, festgelegt. Der Abschluss der Phase Abschluss ist Abhängig vom Abschluss-Meeting und wird mindestens eine Woche vor dem Meeting stattfinden.

E.7 Organigramm

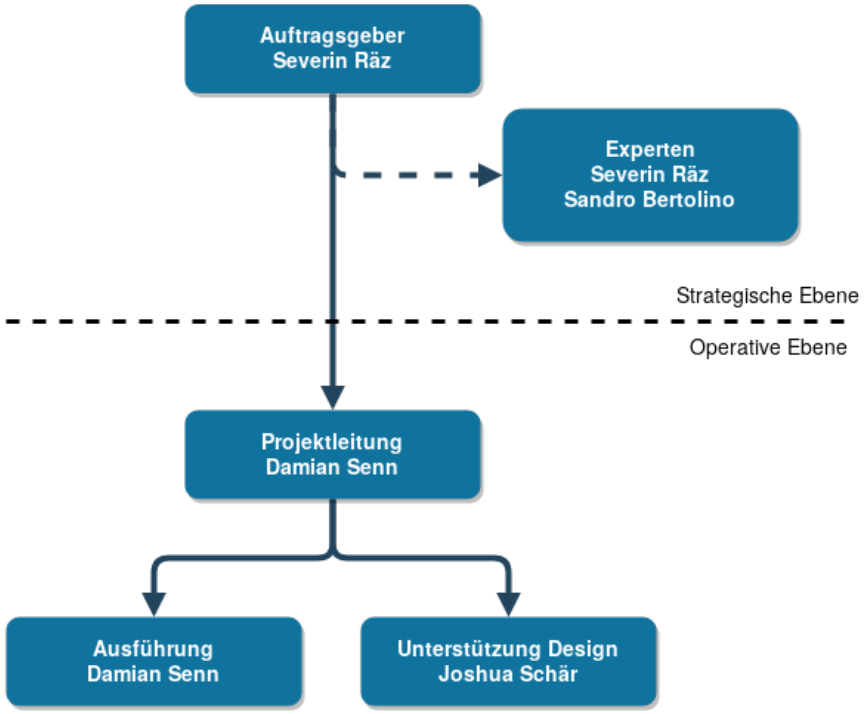


ABBILDUNG E.1: Organigram

E.7.1 Tätigkeiten im Projekt

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

Name	Funktions- und Tätigkeitsbereich
Severin Räz	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 Räz 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

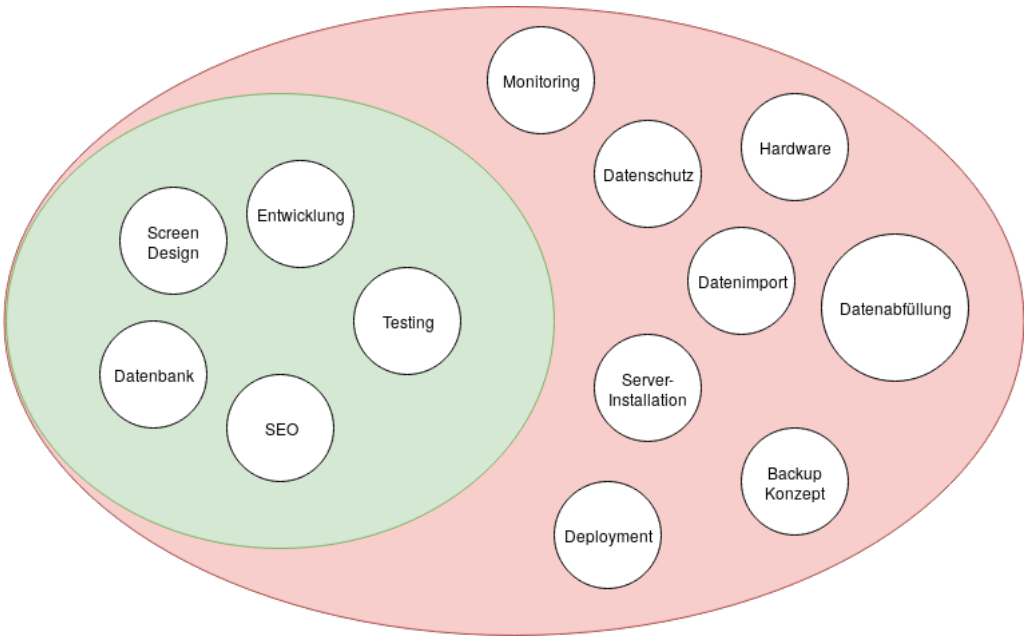


ABBILDUNG E.2: Abgrenzungen

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	**Muss**
	Suche nach Konzertlocation	1.2	Schränkt die Such-Resultate nach gegebener Konzertlocation ein	1.2	**Muss**
	Suche nach Ort	1.2	Schränkt die Such-Resultate nach gegebenem Ort ein	1.2	**Muss**
	Suche nach Genre	1.2	Schränkt die Such-Resultate nach gegebenem Musik-Genre ein	1.2	**Muss**
Design	Desktop	2.1	Alle Ansichten haben eine Desktop-Optimierte Variante	1.4	**Muss**
	Tablet	2.2	Alle Ansichten haben eine Tablet-Optimierte Variante	1.4	**Muss**
	Mobile	2.3	Alle Ansichten haben eine Mobile-Optimierte Variante	1.4	**Muss**
	Browser Kompatibilität	2.4	Alle Ansichten müssen in aktuellem Google Chrome und Mozilla Firefox dem Grundlayout folgen	1.9	**Muss**
SEO	Indexierbarkeit	3.1	Das Produkt ist von Suchmaschinen indexierbar	1.5	**Muss**
	Linked Data	3.2	Konzert Detailseiten sind mit dem Event-Schema[2] ausgestattet	1.5	**Muss**
Benutzer	Registrierung	4.1	Besucher können sich einen Benutzer registrieren, Benutzernamen und E-Mail Adressen müssen einzigartig sein	1.6	**Muss**
	Passwort-Vergessen	4.2	Benutzer können sich einen Passwort-Reset Link anfordern	1.7	**Muss**
	Social	4.3	Benutzer können auf Konzerten vermerken ob sie Teilnehmen oder nicht	1.11	Kann

[2] https://schema.org/MusicEvent

Feature	**Titel**	**Nr.**	**Kriterium**	**Ziel**	**Muss**
Erfassung	Artist	5.1	Benutzer können Artisten mit einem Genre erfassen	1.8	**Muss**
	Location	5.2	Benutzer können eine Konzertlocation mit Ort/Strasse erfassen	1.8	**Muss**
	Konzert	5.3	Benutzer können ein Konzert mit Konzertlocation und Artisten erfassen	1.8	**Muss**
	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	**Muss**
	HTML-Injection	6.2	Das Produkt soll resistent gegen HTML-Injection / XSS sein	1.12	**Muss**
	Passwort encryption	6.3	Passwörter von Benutzer müssen mit einem sicheren Verfahren gespeichert werden	1.12	**Muss**
	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		**Muss**
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 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	64	5'400.- CHF
Total:	286	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
Total:	1'820.- CHF

TABELLE E.7: Betriebskosten

E.12 Risiken

Die Risikobewertung erfolgt mit folgender Formel:

Bewertung = Schaden x 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

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

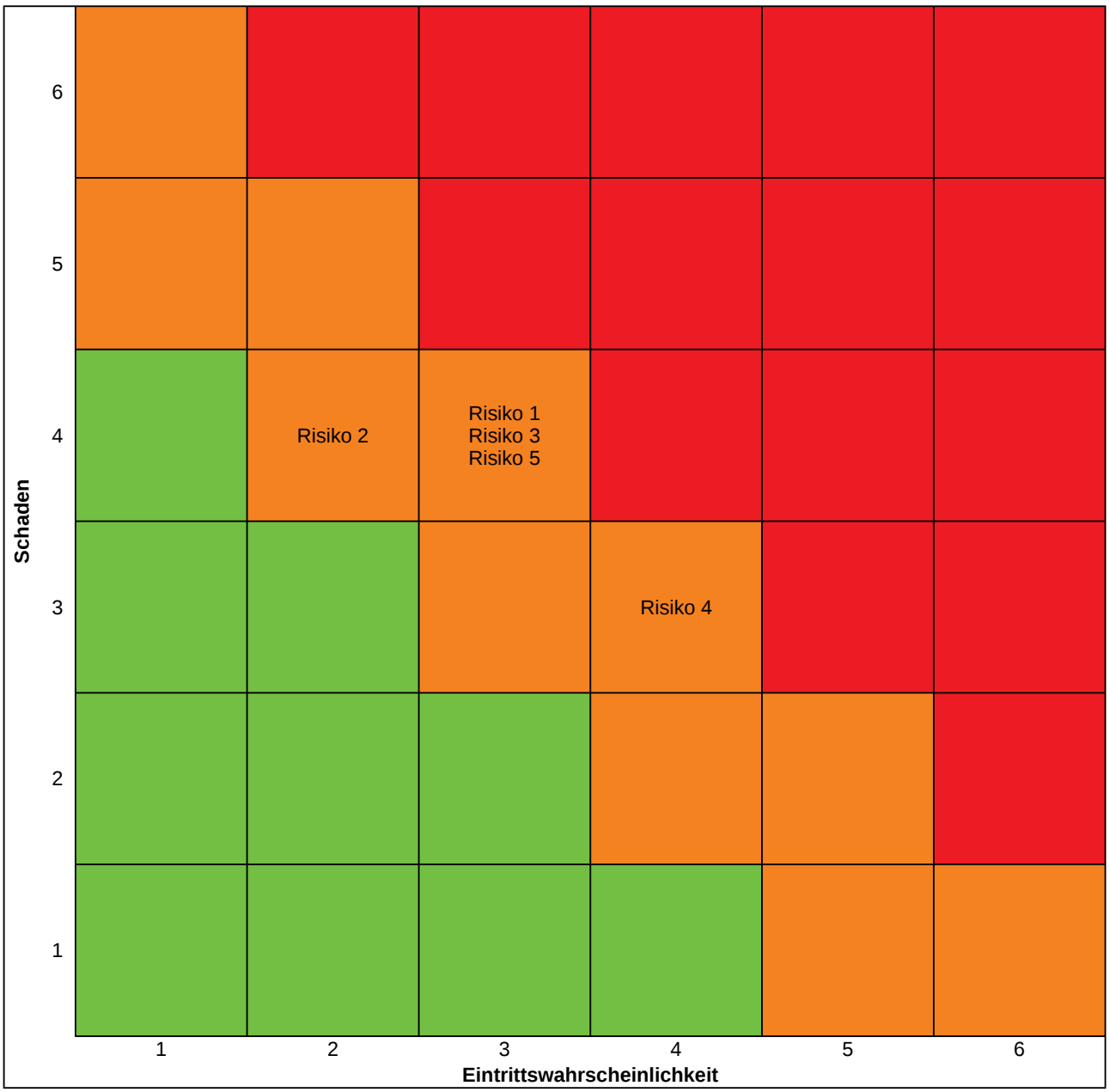
TABELLE E.11: Projektrisiken

E.12.2 Massnahmen

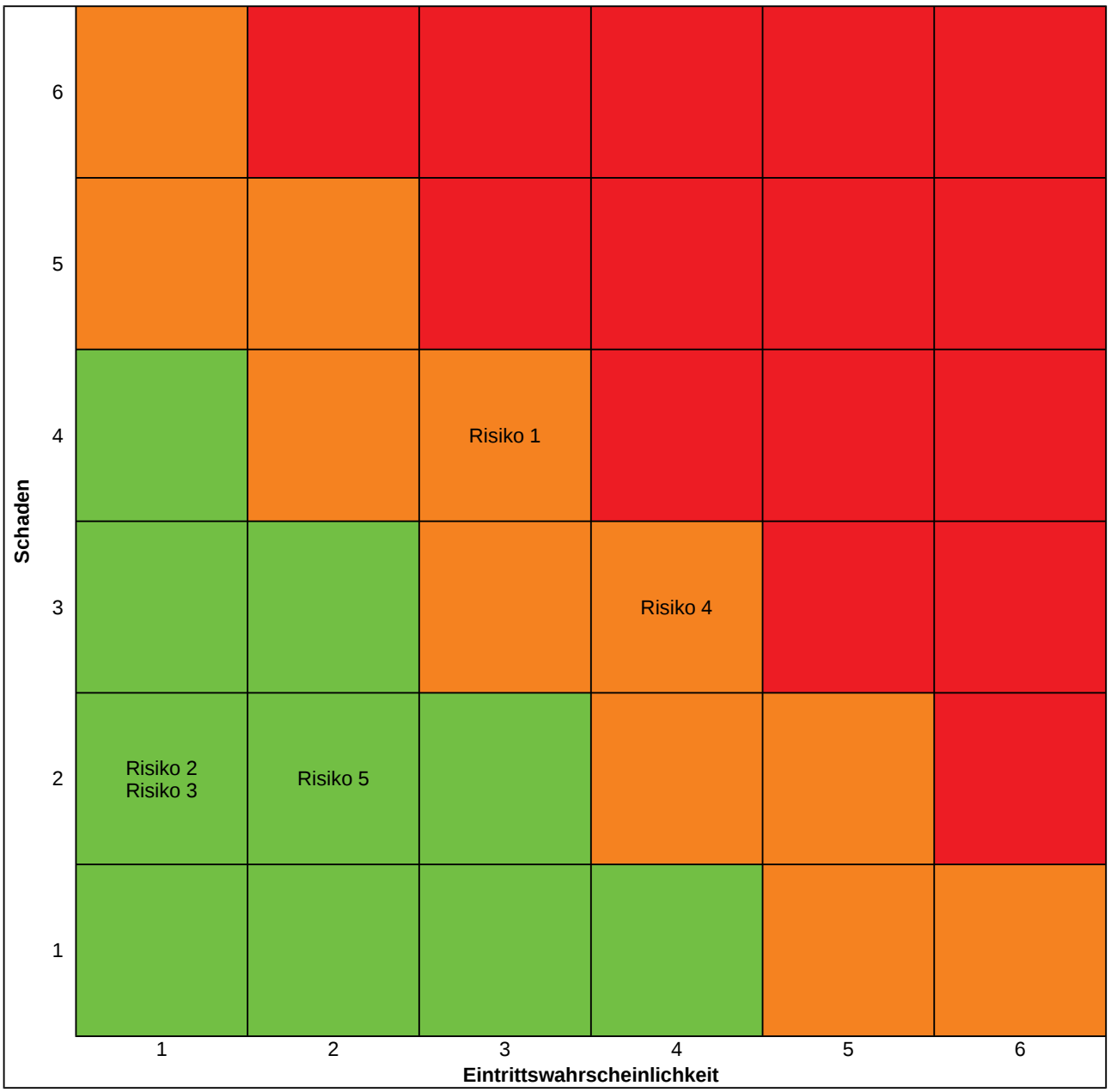
			Bewertung nach Massnahme		
Nr.	**Massnahme**	**Handlung**	**Schaden**	**Wahrsch.**	**Bewertung**
1	Arzt aufsuchen, ggf. Projekt-Pause oder Abbruch	Akzeptanz	4	3	Mittel
2	Statusbericht alle zwei Wochen, bei Fragen sofort Hilfe suchen	Verminderung	2	1	Gering
3	Genügend Buffer-Zeit einplanen, ggf. Ferientage 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

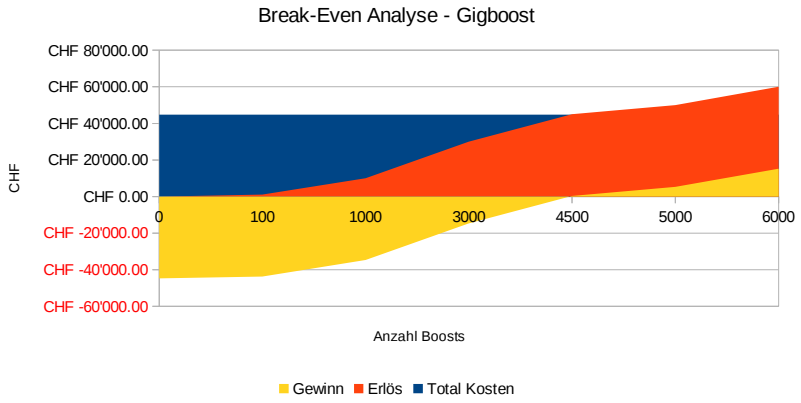


Anhang F

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



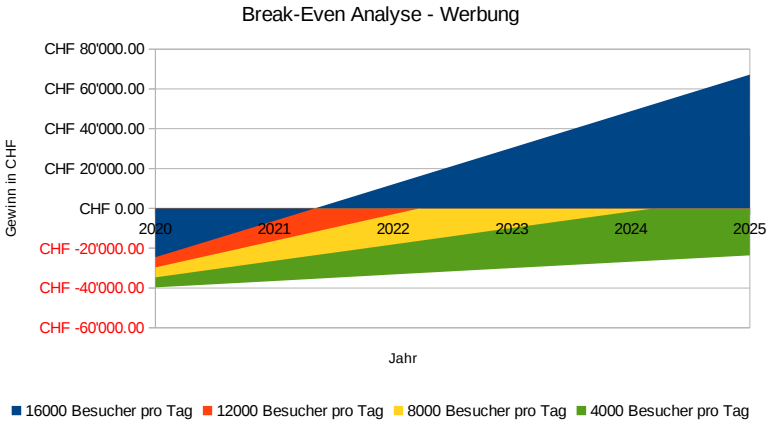
Anhang G

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 un umgesetzt werden.

H.1.1 Teilkonzepte

Durch die in der Studie gewonnenen Erkentnissen, 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
- gigsin.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 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.

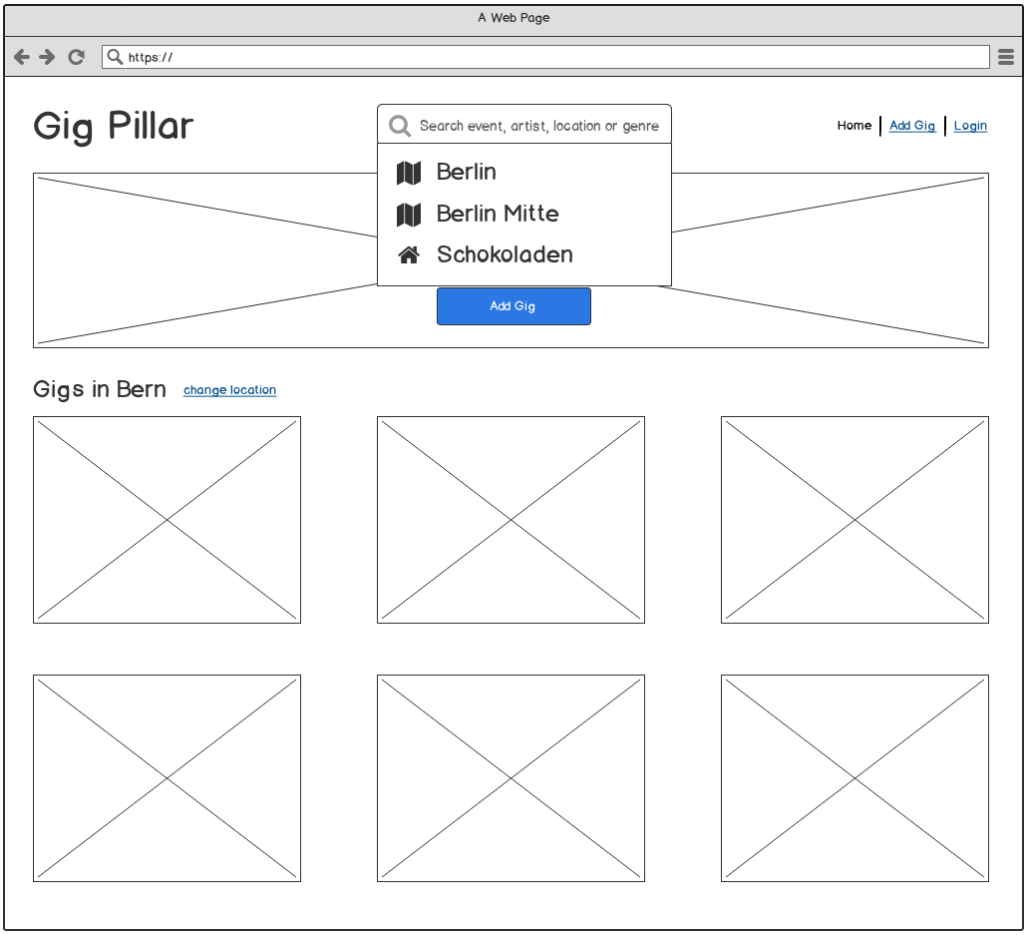


ABBILDUNG H.1: Mockup: Homepage

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.

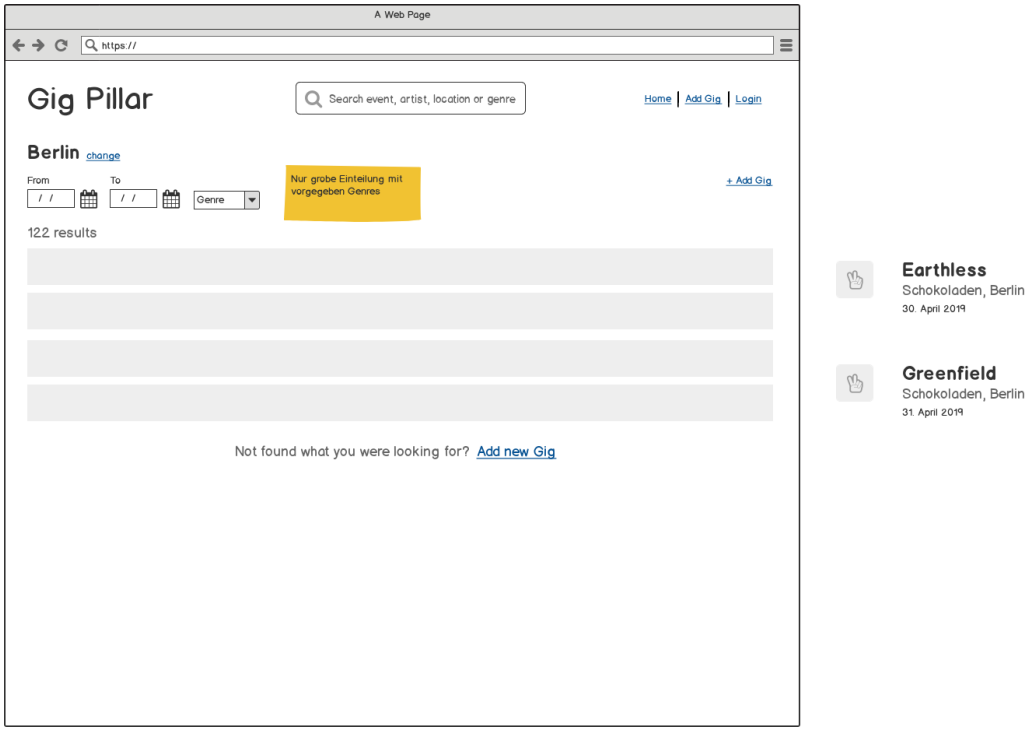


ABBILDUNG H.2: Mockup: Suchresultate

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.

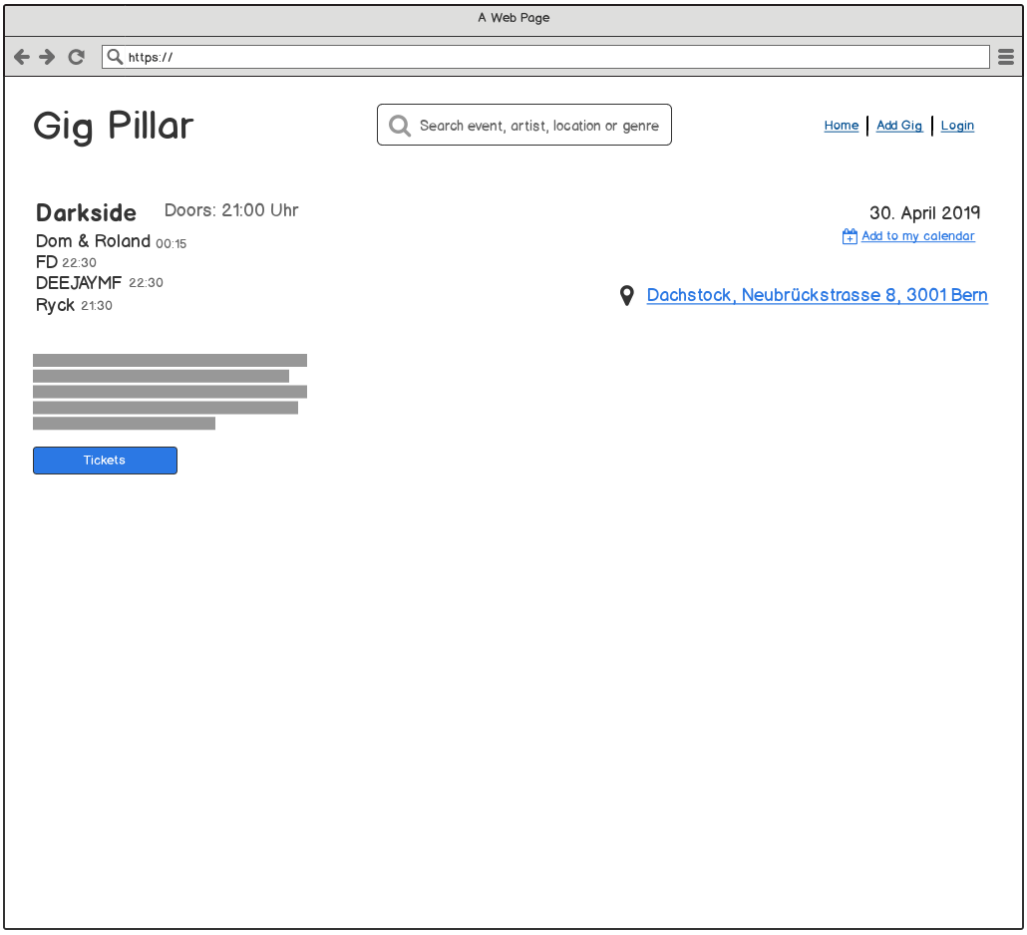


ABBILDUNG H.3: Mockup: Gig Ansicht

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

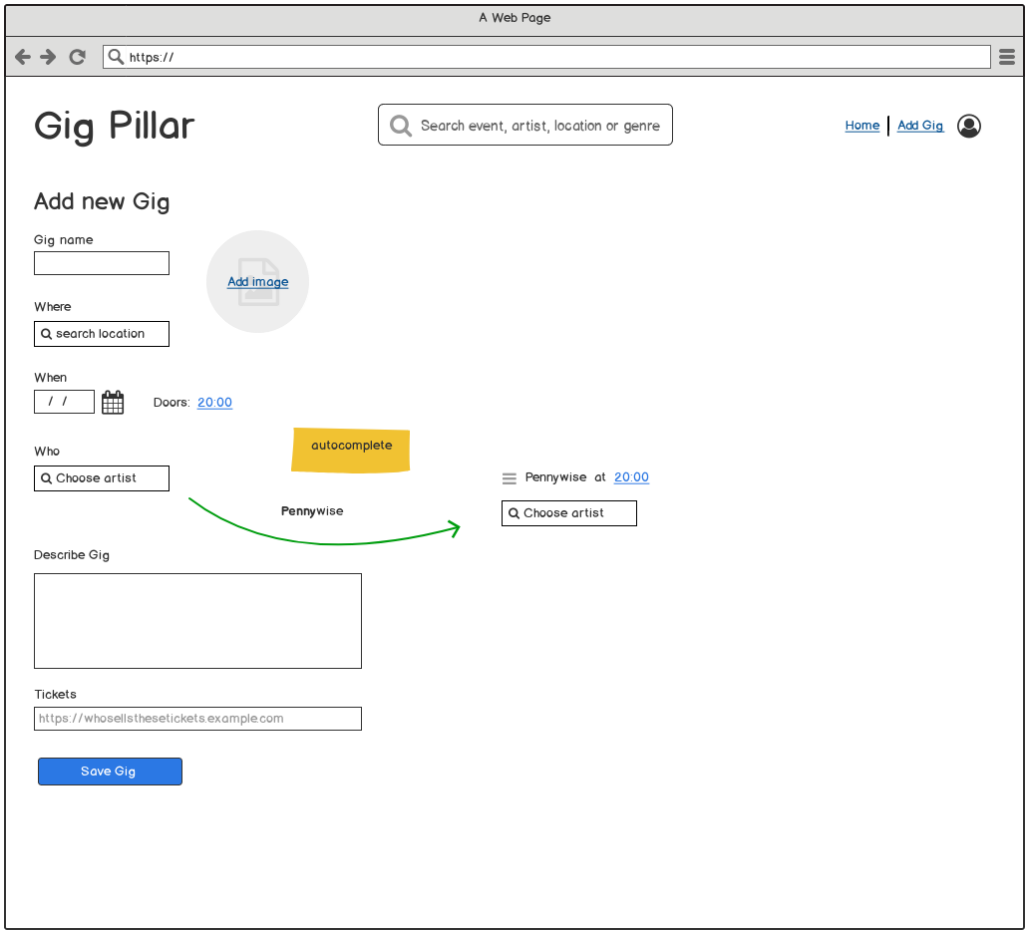


ABBILDUNG H.4: Mockup: Gig erfassen

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!)*

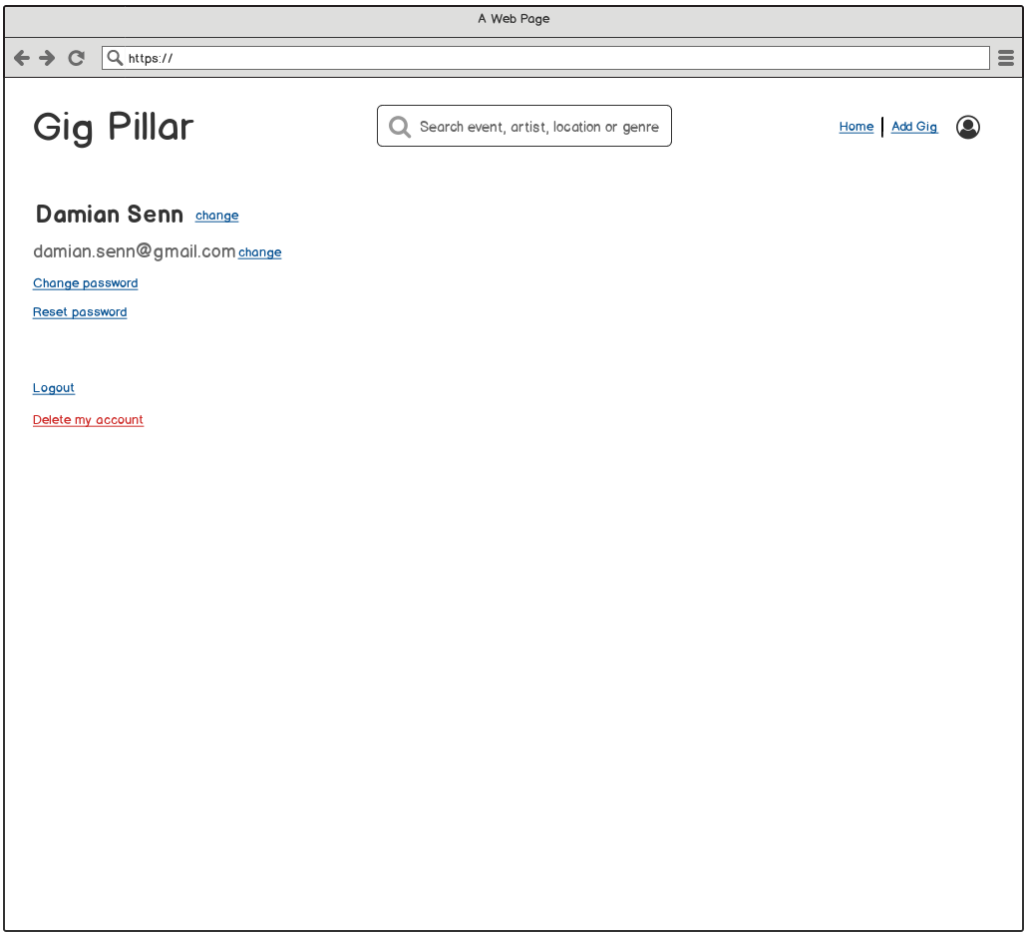


ABBILDUNG H.5: Mockup: Benutzerprofil

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.

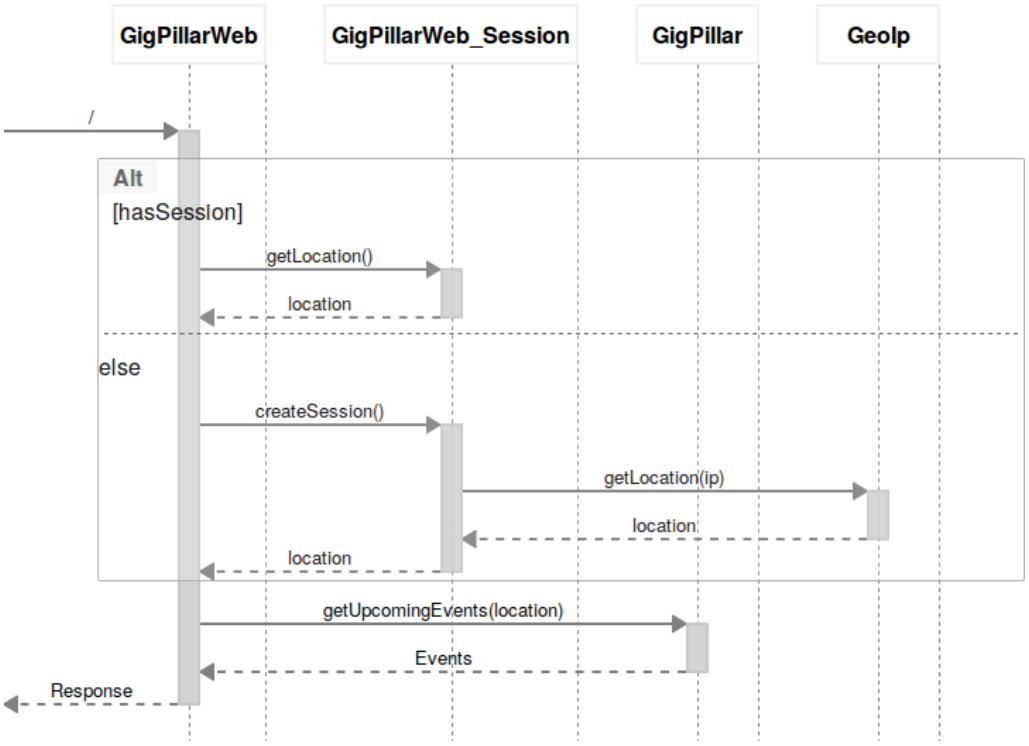


ABBILDUNG H.6: Datenfluss: Homepage

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.

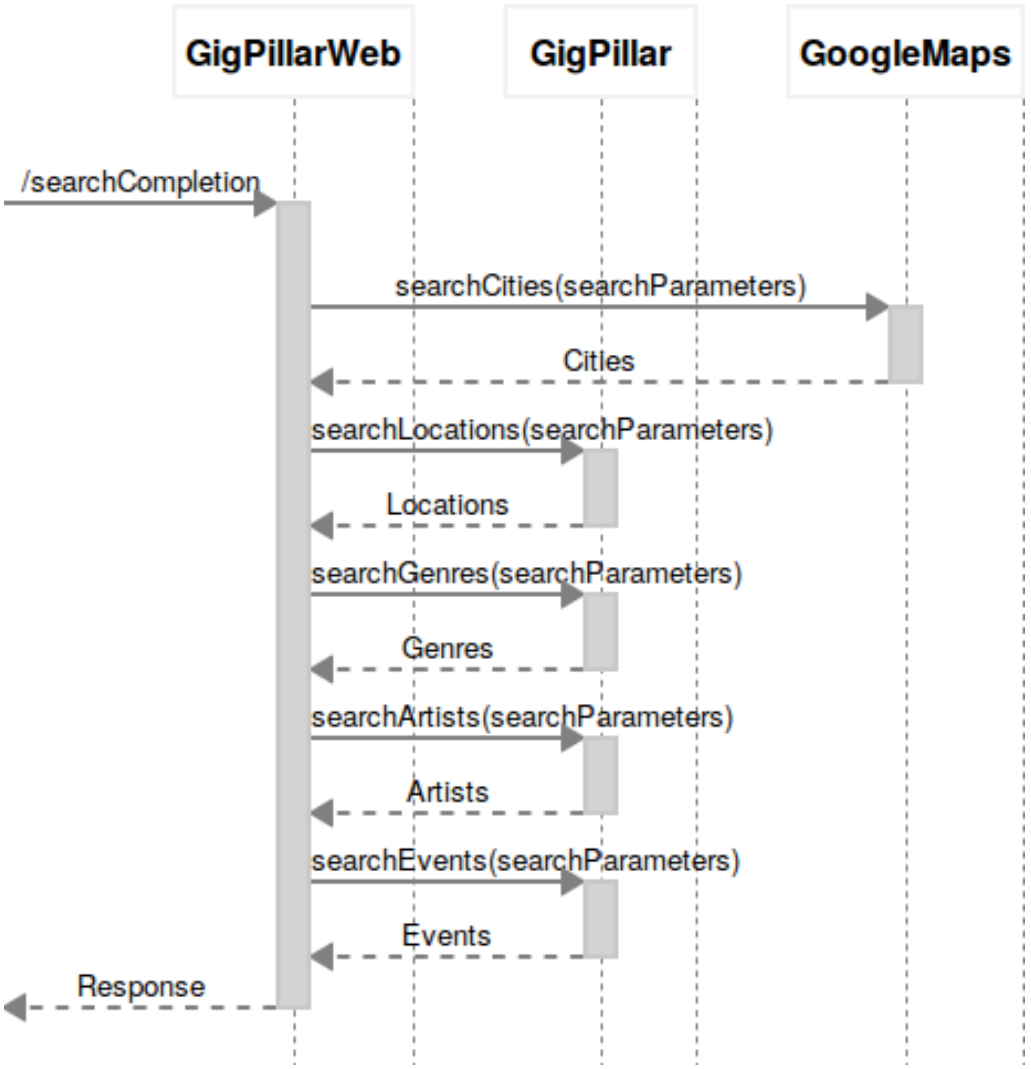


ABBILDUNG H.7: Datenfluss: Suchfeld

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.

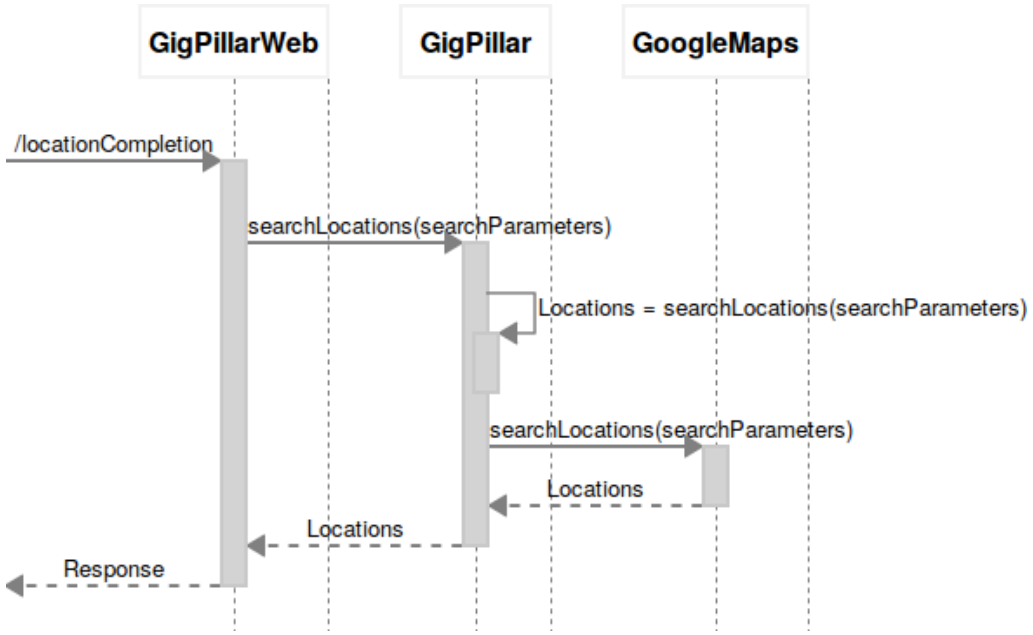


ABBILDUNG H.8: Datenfluss: Gig erstellen - Locationfeld

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.

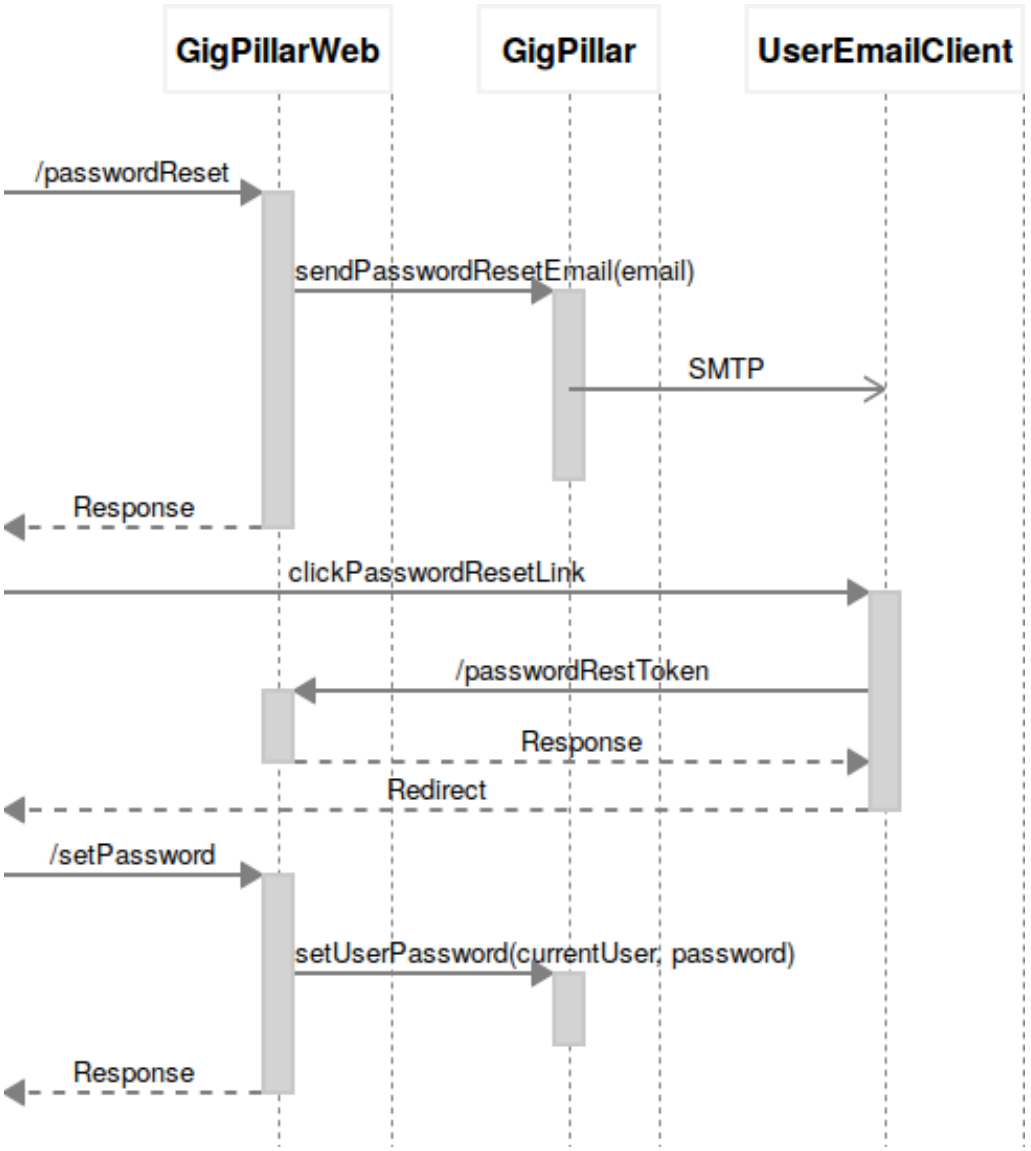


ABBILDUNG H.9: Datenfluss: Passwort-Reset

H.4.2 Datenbankstruktur

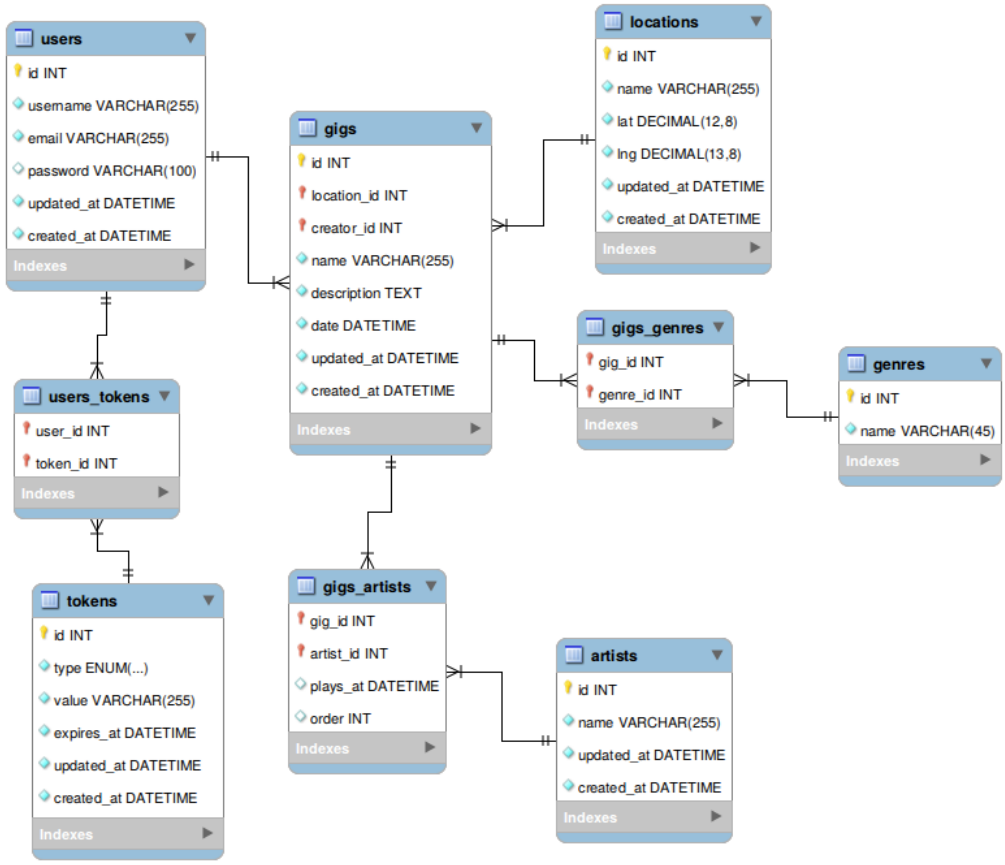


ABBILDUNG H.10: Konzept: Entity Relationship Diagram

H.5 Testkonzept

H.5.1 Unit-Tests

Der Applikationscode soll mit Unit-Tests getestet werden und mindestens eine Abdeckung von 80% des Codes erfüllen.

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

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

TABELLE H.1: Akzeptanztest 01

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

TABELLE H.2: Akzeptanztest 02

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

TABELLE H.3: Akzeptanztest 03

Test 04	**Tester:**	**Datum:**
Kriterium	Browserkompatibilität - Homepage	
Erwartetes Ergebnis	Funktioniert in den unterstützten Browsern.	
Testergebnis		
Fehlerbeschreibung		

TABELLE H.4: Akzeptanztest 04

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

TABELLE H.5: Akzeptanztest 05

Test 06	**Tester:**	**Datum:**
Kriterium	Browserkompatibilität - Suche	
Erwartetes Ergebnis	Funktioniert in den unterstützten Browsern.	
Testergebnis		
Fehlerbeschreibung		

TABELLE H.6: Akzeptanztest 06

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

TABELLE H.7: Akzeptanztest 07

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

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

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

TABELLE H.11: Akzeptanztest 11

Test 12	**Tester:**	**Datum:**
Kriterium	Browserkompatibilität - Registrierung	
Erwartetes Ergebnis	Funktioniert in den unterstützten Browsern.	
Testergebnis		
Fehlerbeschreibung		

TABELLE H.12: Akzeptanztest 12

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

TABELLE H.13: Akzeptanztest 13

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

TABELLE H.14: Akzeptanztest 14

Test 15	**Tester:** **Datum:**
Kriterium	Gig erfassen
Erwartetes Ergebnis	Folgende Daten können erfasst werden: • Name • Bild *(optional)* • Location • Datum • Zeit *(optional)* • Künstler mit optionaler Start-Zeit • Beschreibung • Link zum Ticketvertreiber *(optional)*
Testergebnis	
Fehlerbeschreibung	

TABELLE H.15: Akzeptanztest 15

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

TABELLE H.16: Akzeptanztest 16

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

TABELLE H.17: Akzeptanztest 17

Test 18	**Tester:**	**Datum:**
Kriterium	Browserkompatibilität - Benutzerprofil	
Erwartetes Ergebnis	Funktioniert in den unterstützten Browsern.	
Testergebnis		
Fehlerbeschreibung		

TABELLE H.18: Akzeptanztest 18

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

TABELLE H.19: Akzeptanztest 19

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

TABELLE H.20: Akzeptanztest 20

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

TABELLE H.21: Akzeptanztest 21

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

TABELLE H.22: Akzeptanztest 22

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

TABELLE H.23: Akzeptanztest 23

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

TABELLE H.24: Akzeptanztest 24

H.6 Fazit

H.6.1 Probleme

H.6.2 Machbarkeit

H.6.3 Wirtschaftlichkeit

H.6.4 Erweiterbarkeit

H.6.5 Projektplan

Anhang I

Realisierung

I.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 URL `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`

I.2 Projekt Setup

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

```
$ 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.
Erläuterung der Projekt/Dateistruktur:

```
/apps/gigpillar/
```

Die Gigpillar Grundapplikation

```
# Die Gigpillar Webapplikation
/apps/gigpillar_web/
# Applikations{\"u}bergreifende Konfigurationsfiles
/config/
# Die Projektdokumentation
/doc/
# Der HTML-Prototyp
/html-prototype/
# Die Dockerkonfiguration f{\"u}r die Entwicklungsumgebung
/docker-compose.yml
# Die applikationsubergreiffenden Abhangigkeiten
/mix.exs
```

I.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»[1] ausgewählt, da diser 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:

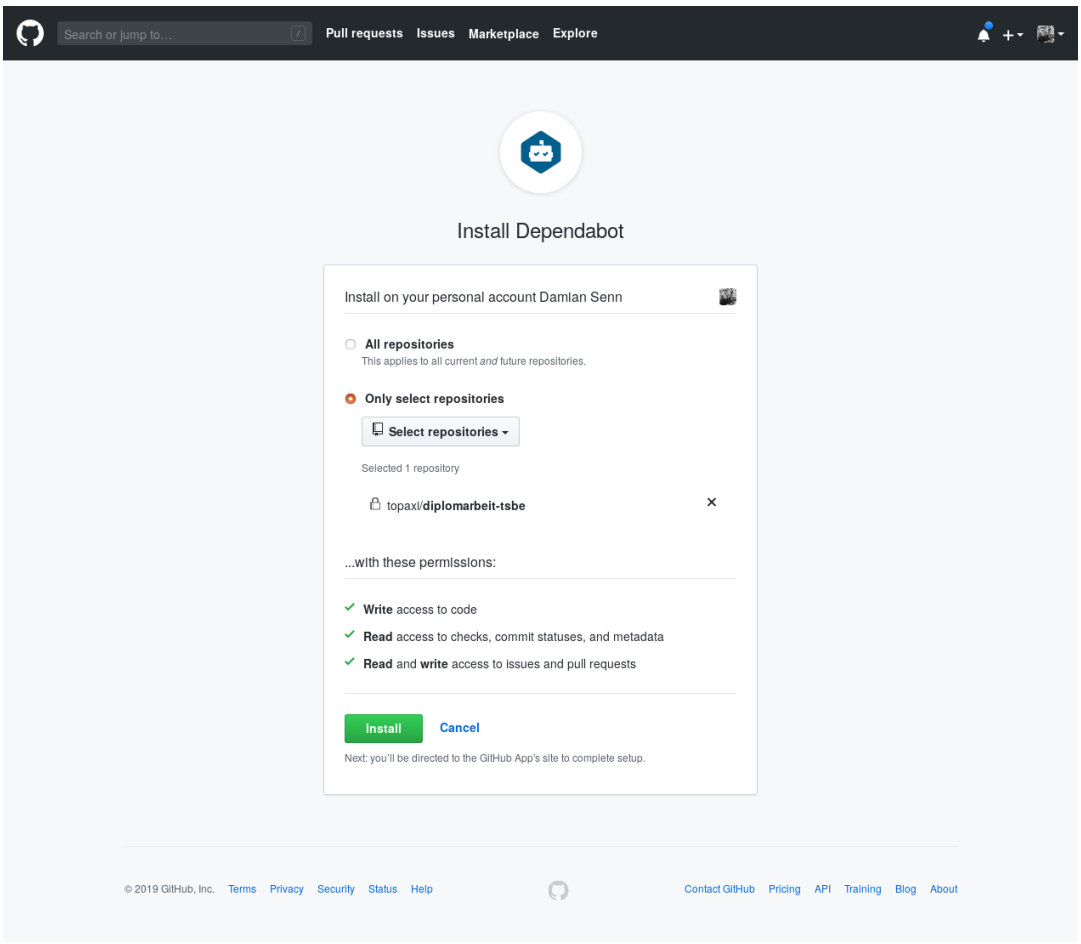


ABBILDUNG I.1: Dependabot: Installation

[1] https://github.com/marketplace/dependabot

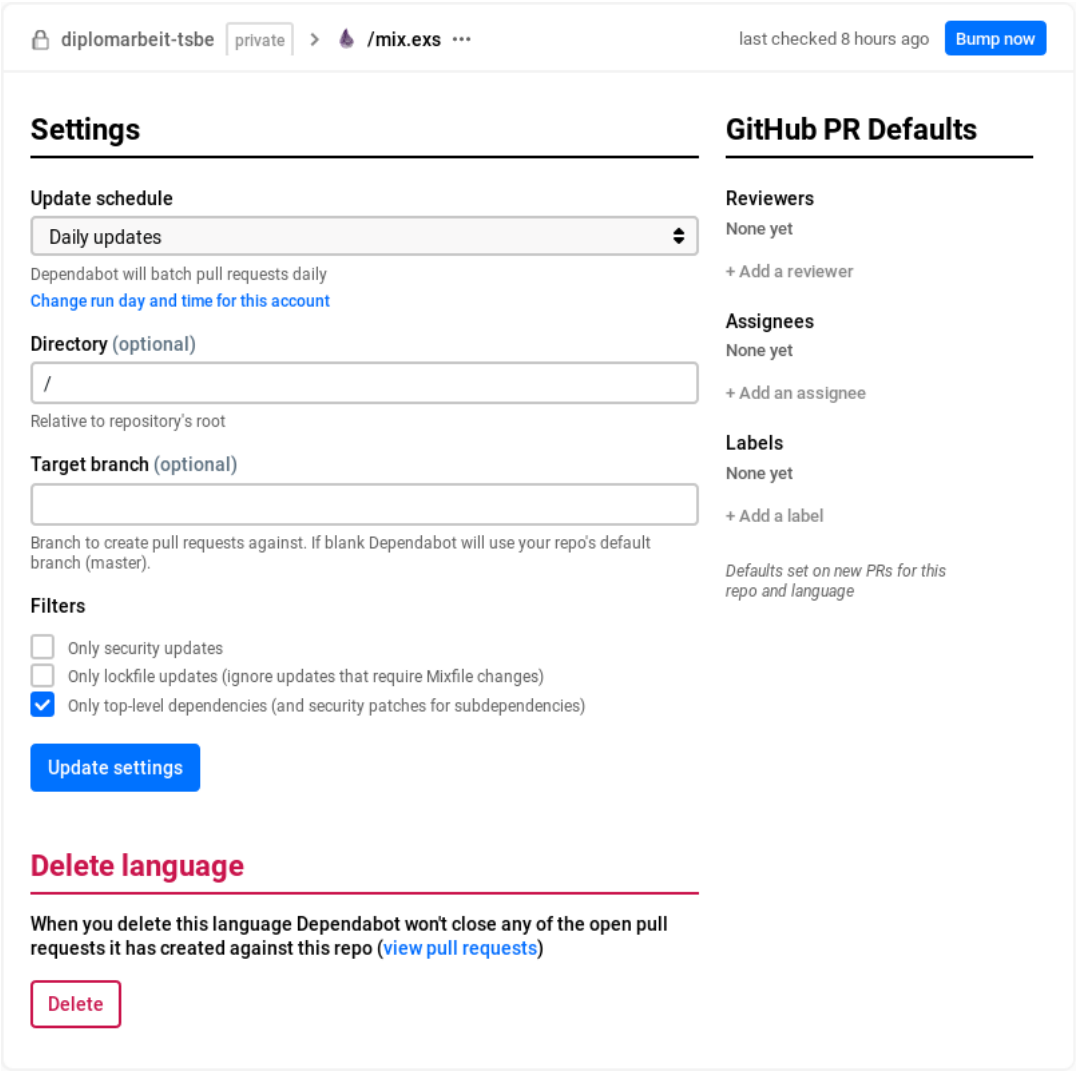


ABBILDUNG I.2: Dependabot: Konfiguration für Elixir

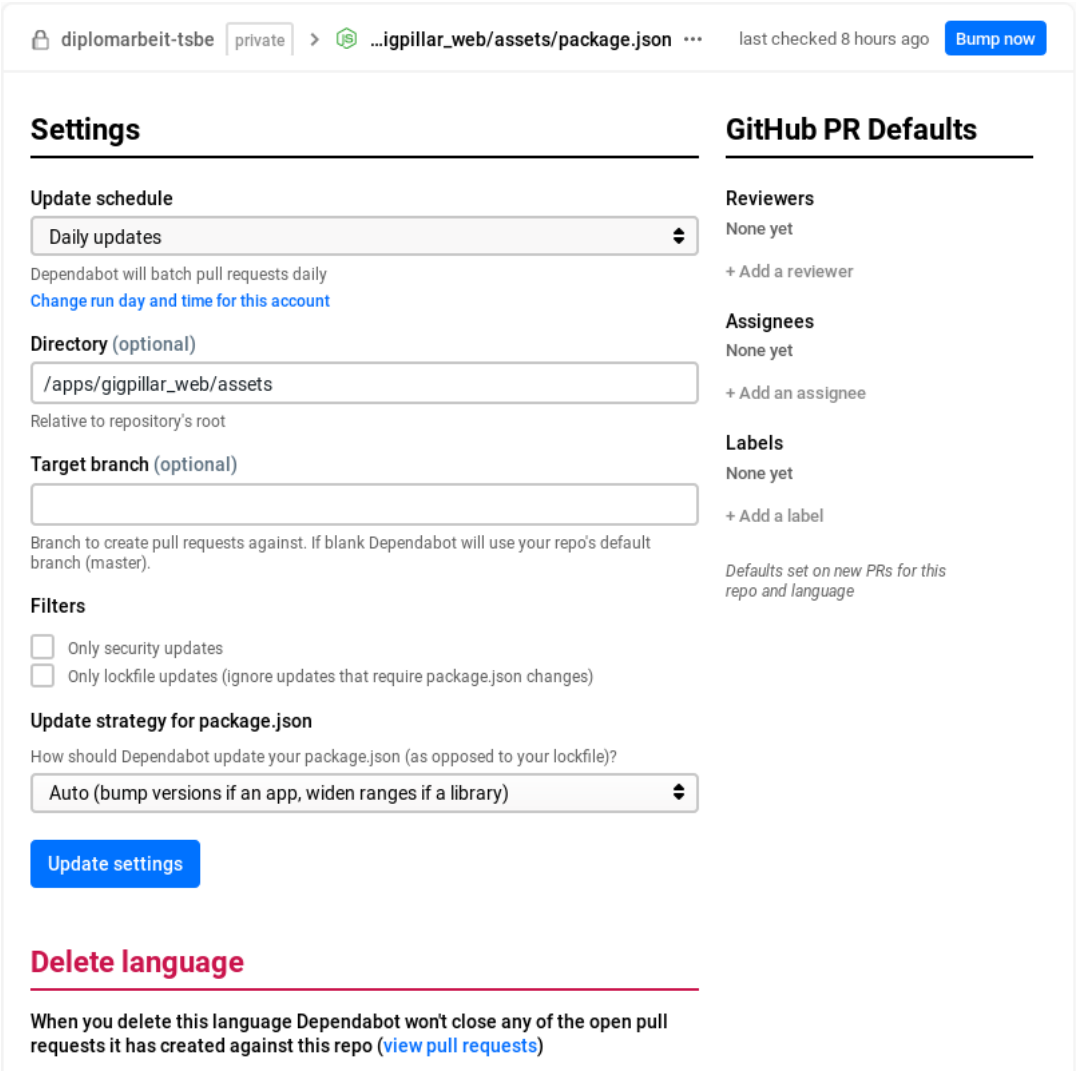


ABBILDUNG I.3: Dependabot: Konfiguration für JavaScript

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

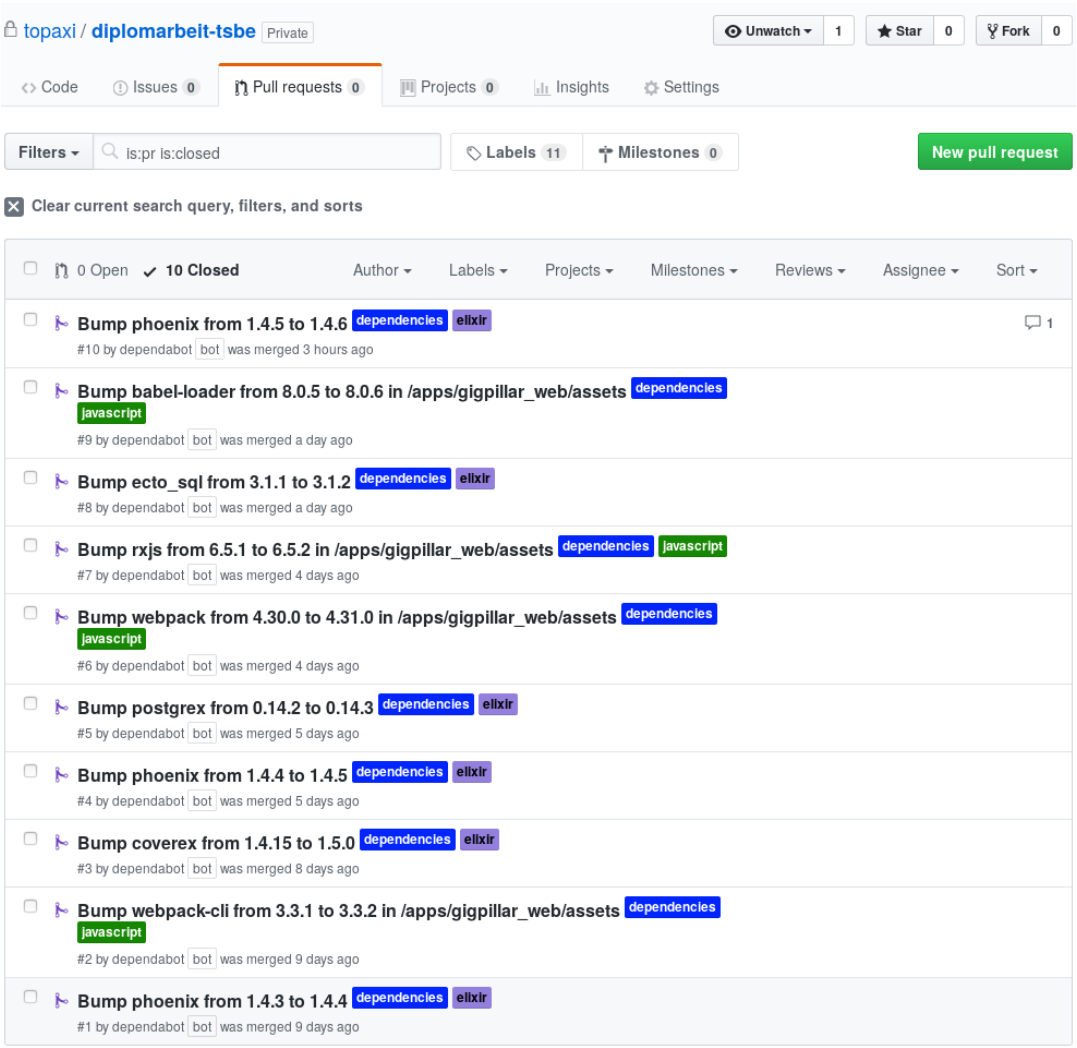


ABBILDUNG I.4: Dependabot: Pull-Requests für Updates

I.4 Datenbankschema

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.

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.

I.4.1 Finales Schema

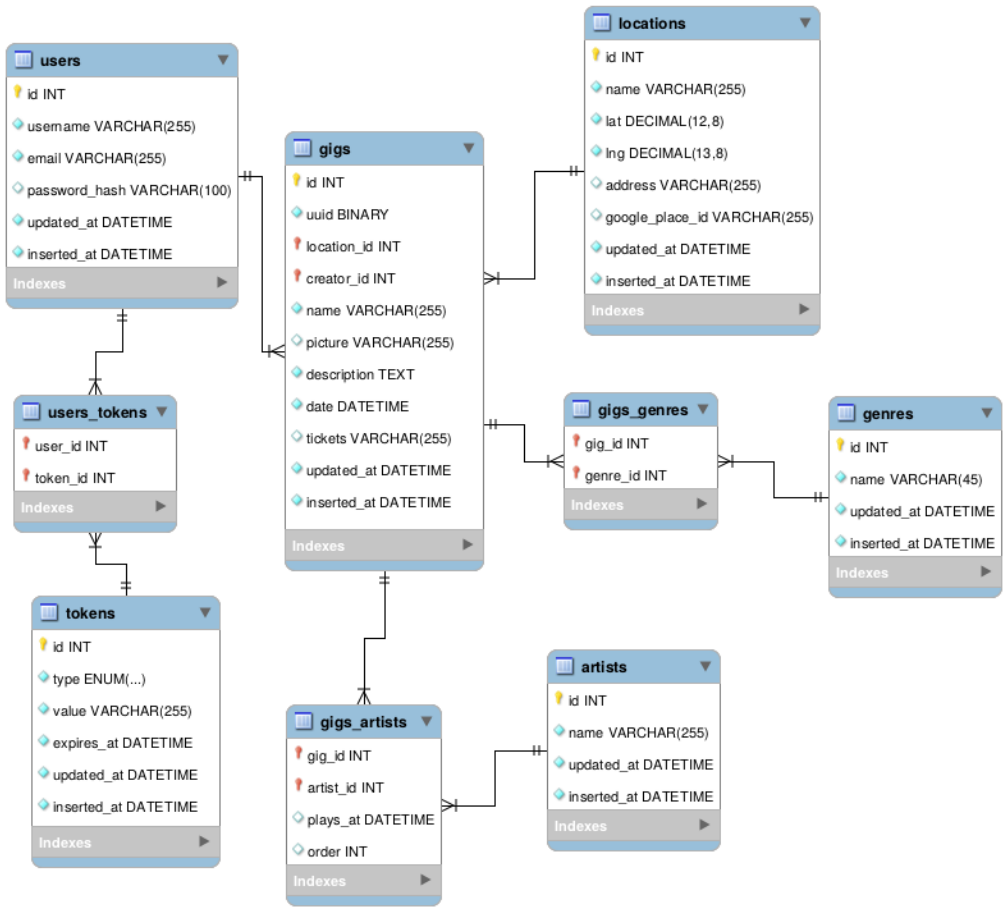


ABBILDUNG I.5: Realisierung: Entity Relationship Diagram

I.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»[2] 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
```

[2] https://github.com/cpjk/canary

I.6 Location Autocomplete

Das Location Autocomplete Feld wurde mit der «Google Place API»[3] 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

[3] https://developers.google.com/places/web-service/autocomplete

I.7 HTML Erweiterungen

Während der Entwicklung des Projektes, wurden einige spezielle HTML-Elemente definiert. Diese Elemente wurden mit **LitElement**[4] 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:

```
<search-box
  inputId="suche"
  src="/api/autocomplete"
  name="suche"
  placeholder="Suche..."
  value="Suchbegriff"
  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:** URL 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:

```
<with-dropdown>
  <input type="text">
  <ul slot="dropdown">
    <li>Lorem</li>
    <li>Ipsum</li>
  </ul>
</with-dropdown>
```

[4] 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:

```
<location-input
  inputId="gig_location"
  name="gig[location]"
  location="{
    "name": "Dachstock",
    "google_place_id": "ChIJ-SskKr45jkcRPqmGB-ZGsRE"
  }"></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 **JSON**-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:

```
<picture-input
  inputId="gig-picture"
  name="gig[picture]"
  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:** URL 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:

```
<datetime-input
  inputId="datetime"
  dateLabel="Datum"
  timeLabel="Uhrzeit"
  name="datetime"
  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**[5] 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:

```
<artists-input
  inputId="artists"
  name="gig[artists]"
  placeholder="K&uuml;nstler suchen..."
  value="[
    {"id":42,"name":"Parkway Drive","plays_at":"23:00"},
    {"id":23,"name":"The Ghost Inside","plays_at":"21:00"}
  ]"></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 **JSON**-Objekten der einem Gig assozierten Künstler.

[5] https://de.wikipedia.org/wiki/ISO_8601

I.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 I.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 I.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 I.3: Bilder Optimierungen

I.9 File Upload

Für den Upload der Bilder von Gigs, wird die Software **«minio»**[6] 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 I.4). Das **uuid** Feld wird mit einem möglichst einzigartigen Wert abgefüllt, der Wert entspricht dem **UUID version 4**[7] 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»**[8] 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

I.10 OpenStreetMap

Auf der Gig Detailansicht, wird eine OpenStreetMap eingebunden. Damit die OpenStreetMap Karte korrekt dargestellt wird, mussten die Geokoordinaten auf einzelne sogennante «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[9].

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

[6] https://min.io/
[7] https://de.wikipedia.org/wiki/Universally_Unique_Identifier
[8] https://imagemagick.org/
[9] https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames

I.11 SEO - Microdata

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

Beispiel:

```
<script type="application/ld+json">
{
  "@context": "http://schema.org",
  "@type": "MusicEvent",
  "name": "Darkside",
  "startDate": "2019-04-20T22:00:00Z",
  "image": "http://localhost:4040/uploads/gigs/051cd-...",
  "description": "Die DARKSIDE beherbergt diesen ...",
  "performer": [
    { "name": "Dom & Roland", "@type": "MusicGroup" },
    { "name": "FD", "@type": "MusicGroup" },
    { "name": "Ryck", "@type": "MusicGroup" }
  ],
  "location": {
    "@type": "MusicVenue",
    "name": "Dachstock",
    "address": "Neubrueckstrasse 8, Bern, Switzerland"
  },
  "offers": {
    "@type": "Offer",
    "url": "https://www.petzitickets.ch/"
  }
}
</script>
```

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

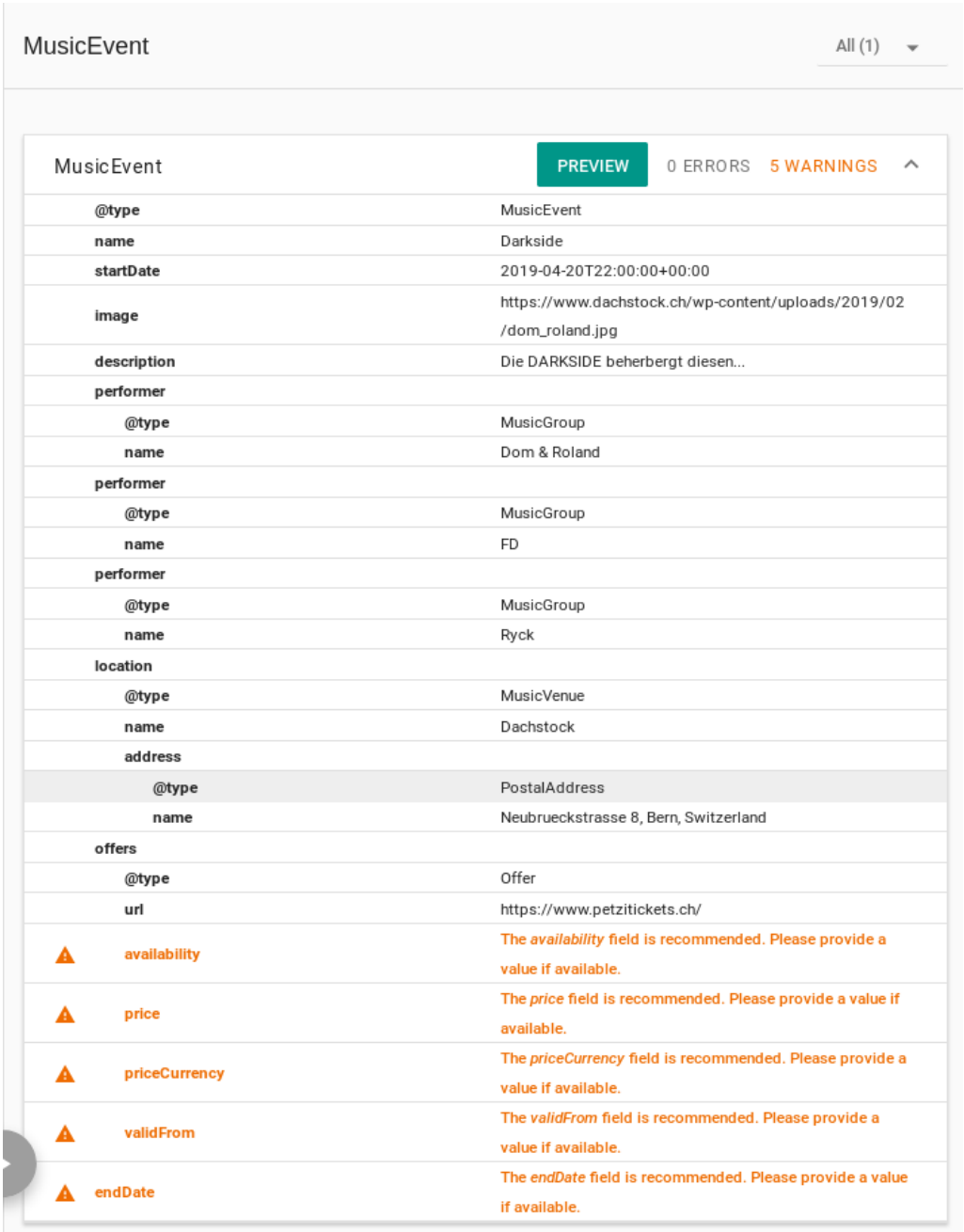


ABBILDUNG I.6: JSON-LD: Validierung

[10] https://search.google.com/structured-data/testing-tool

I.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
```

I.13 Probleme

I.13.1 Dependency Konflikt

ExAws

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

```
Failed to use "poison" (version 4.0.1) because
  arc (version 0.11.0) requires ~> 2.2 or ~> 3.1
  coverex (version 1.5.0)
    requires ~> 3.0 or ~> 3.1 or ~> 4.0
  deps/google_api_client/mix.exs
    requires ~> 1.5 or ~> 2.0 or ~> 3.0 or ~> 4.0
  wallaby (version 0.22.0) requires >= 1.4.0
  mix.lock specifies 4.0.1

** (Mix) Hex dependency resolution failed, relax the version
  requirements of your dependencies or unlock them (by using
  mix deps.update or mix deps.unlock). If you are unable to
  resolve the conflicts you can try overriding with
  {: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.

```
defmodule Gigpillar.Umbrella.MixProject do
  defp deps do
    [
      {:poison, "~> 4.0", runtime: false, override: true}
    ]
  end
end
```

I.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 gestelten 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[11]. 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**[12] Funktion die auch ohne Changeset funktioniert.

Einige Zeit später fand ich jedoch im Phoenix Source Code[13] heraus, dass für das von Phoenix bereitgestellte Verbindungsobjekt **Plug.Conn** das **Phoenix.HTML.FormData** Protokol 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.

I.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**»[14] installiert.

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

[11] https://hexdocs.pm/ecto/Ecto.Changeset.html

[12] https://hexdocs.pm/phoenix_html/Phoenix.HTML.Tag.html#form_tag/2

[13] https://github.com/phoenixframework/phoenix_html/blob/f9a4f38/lib/phoenix_html/form_data.ex#L51

[14] https://github.com/bitwalker/timex

I.14 Tests

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

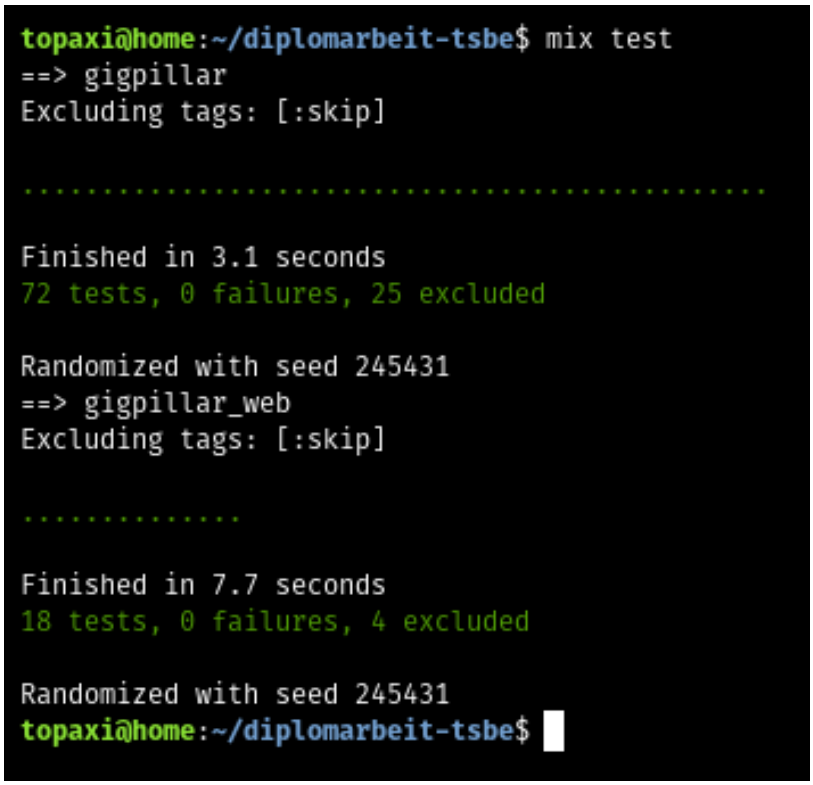


ABBILDUNG I.7: Tests: Ausführung der automatisierten Tests

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[15] heruntergeladen oder über den Node Package Manager (NPM) installiert werden.

Installation mit NPM:

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

[15] http://chromedriver.chromium.org/

I.15 Offene Punkte

I.15.1 Styling Suche

I.15.2 Genre Zuordnung für Gigs

I.15.3 Benutzer Profil

I.15.4 Passwort-Reset Funktionalität

I.15.5 Email-Bestätigung

I.15.6 Screenshot Tests

I.15.7 Expliziter Ort Kontext

Implizit in Konzept Mockups, Kriterium jedoch abgedeckt durch Suchfilter

I.15.8 Mögliche Erweiterungen

- ïn deiner nähe"

I.16 Auswertung

Anhang J

Einführung

J.1 Installationsanleitung

J.1.1 Benötigte Software

Server

- geoip db - imagemagick

Testumgebung

- webdriver (npm, für tests)

J.1.2 Umgebungsvariablen

Name	Beschreibung
AWS_S3_BUCKET	
AWS_ACCESS_KEY_ID	
AWS_SECRET_ACCESS_KEY	
AWS_REGION	
AWS_S3_SCHEME	
AWS_S3_HOST	
AWS_S3_PORT	
GIGPILLAR_ASSET_HOST	
POSTGRES_USER	
POSTGRES_PASSWORD	
POSTGRES_DB	
POSTGRES_HOSTNAME	
POSTGRES_POOL_SIZE	
GEOIP_DATABASE_FILE	

J.1.3 Initialen Benutzer erstellen

Es wurde ein Skript erstellt um den ersten, initialen Benutzer zu erstellen. Dies sollte vorallem 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.1: Initialen Benutzer erstellen

Anhang K

Arbeitsjournal

K.1 Sonntag 3. März

2h:

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

K.2 Dienstag 5. März

2h:

- Vorbereitung Kick-off

K.3 Mittwoch 6. März

3h:

- Vorbereitung Sitzungszimmer
- Kick-off Meeting

4h:

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

K.4 Samstag 9. März

2h:

- An Studie/Pflichtenheft arbeiten

K.5 Dienstag 12. März

2h:

- PDF Generierung und Ordnerstruktur angepasst

K.6 Samstag 16. März

3h:

- Projektplan von gantt nach ods migrieren

K.7 Dienstag 19. März

0.5h:

- Projektplan in Berichtanhang angehängt

K.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

K.9 Sonntag 31. März

5h:

- Projektziele erweitert
- Anforderungskatalog erweitert

K.10 Sonntag 31. März

8h:

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

K.11 Freitag 5. April

2h:

- An Studie weiter gearbeitet
- Variantenkriterien definiert

K.12 Samstag 6. April

4h:

- An Studie weiter gearbeitet
- Variantenbeschreibungen erstellt
- Variantenbewertungen

K.13 Mittwoch 10. April

2h:

- Brainstorming für Portalnamen

K.14 Freitag 12. April

12h:

- Screens definiert
- Mit Joshua Schär an Mockups gearbeitet

K.15 Samstag 13. April

12h:

- Risiko Management
- Variantenbeschreibungen angepasst/erweitert
- Projektauftrag

K.16 Sonntag 14. April

8h:

- Wirtschaftlichkeit
- Projektauftrag

K.17 Montag 15. April

2h:

- Initiales Datenbankschema basierend auf Mockups

K.18 Mittwoch 17. April

0.5h:

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

K.19 Freitag 19. April

10h:

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

K.20 Sonntag 21. April

10h:

- Mockup Beschreibungen
- Datenbankschema
- Mit Testkonzept begonnen

Anhang L

Biweekly Reports

L.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 Organigram, 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

L.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

L.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