Projektdokumentation Twitter Analyse

Joris Baiutti
Patrick Wyss

Version 1.0, 19.01.2018

Berner Fachhochschule
Departement für Technik und Informatik

Inhaltsverzeichnis

1 Zweck des Dokuments

Das Dokument soll zusätzlich zur technischen Dokumentation, dem Javadoc und der Anforderungsspezifikation über unser Vorgehen, unsere Zusammenarbeit, Erkenntnisse und Herausforderungen sowie über unser Fazit dieser Arbeit informieren.

2 Vorgehen

2.1 Sammeln von Anforderungen

Um die Anforderungen zu definieren haben wir zuerst ein Interview mit Annett Laube geführt. Durch dieses Gespräch könnten wir einen guten Überblick der Anforderungen seitens BFH Digital Society erhalten. Zusätzlich hat uns Frau Laube an Anne-Careen Stoltze-Siebmann für weitere Ideen weitergeleitet.
Hier einige Beispiele dieser Inputs:

- Graph von Personen welche den Kanal mit Informationen füttern Likes, Retweets
- Welche Themen werden wie oft bedient
- Follower Untersuchen Themenbereiche, Retweets
- Fake News erkennen
- Follower Zuwachs beobachten
- Welche Hashtags werden oft benutzt

Anschliessend haben wir folgende funktionalen Anforderungen definiert:
(auch im Dokument Anforderungsspezifikation Twitteranalyse V1.0 zu finden)

ID	Status	Prio	Beschreibung
F1.1	Entwurf	P2	Sentiment Analyse von Twitter Information mit einer bestehenden Java Library (Opennlp). Trainingsdaten werden aus dem Internet verwendet.
F1.2	Entwurf	P1	Likes, Tweets, Retweets zählen und Graphisch darstellen. Es werden nicht einfach nur die Likes eines Accounts gezählt, sondern Likes aus einer Sammlung von Tweets welche durch ein Stichwort gefiltert wurde, z.B. alle Likes zu "Big Data"
F1.3	Entwurf	P3	Graph von Personen welche die Tweets retweeten.
F1.4	Entwurf	P1	Anzahl der Tweets, über einen gewissen Zeitraum, gruppiert nach den einzelnen Themen der Digital Society, vergleichen und grafisch darstellen
F1.5	Entwurf	P1	Welche Hashtags werden in einer Sammlung von Tweets am meisten gebraucht. Die Sammlung der Tweets wird nach einem Stichwort gefiltert
F2.1	Entwurf	P1	Eine zentrale Datenbank von Twitterdaten, für das Entwickeln von Analysen. Über Schnittstellen bekommt man von Twitter nur beschränkt vergangene Tweets. Daher wird eine zentrale Datenbank mit schon vorhandenen Twitterdaten zu gewissen Themen zur Verfügung gestellt.
F2.2	Entwurf	P2	Zentrale Datenbank für Training Models für Analysen mit machine learning. Damit mit dem System gleich getestet und ausprobiert werden kann, werden Test und Trainigsdaten Zentral hinterlegt.
F3.1	Entwurf	P3	Englische und deutsche Tweets analysieren
F3.2	Entwurf	P1	Analysieren von Informationen in der Vergangenheit
F3.3	Entwurf	W	Fake News erkennen und filtern.
F4.1	Entwurf	P2	Die Grafischen Analysen werden auf Webseiten der BFH eingebunden
F4.2	Entwurf	P3	Das Frontend respektive die Graphischen Analysen lassen sich auf einem Desktop, sowie auf einem mobilen Gerät betrachten
F4.3	Entwurf	W	Charts mit einem Generator erstellen lassen (Widget)

Tabelle 1: Funktionale Anforderungen

2.2 Ideen Sammeln

Da unser Projekt vor allem den Backend Teil priorisiert hat und das Frontend eher als Beispiel dient haben wir und für das Design nur rudimentäre Überlegungen gemacht. Diese ersten Zeichnungen konnten wir mit charts.js abdecken. Weitere graphische Darstellungen wie GoogleMaps sind innerhalb der Sprints entstanden.

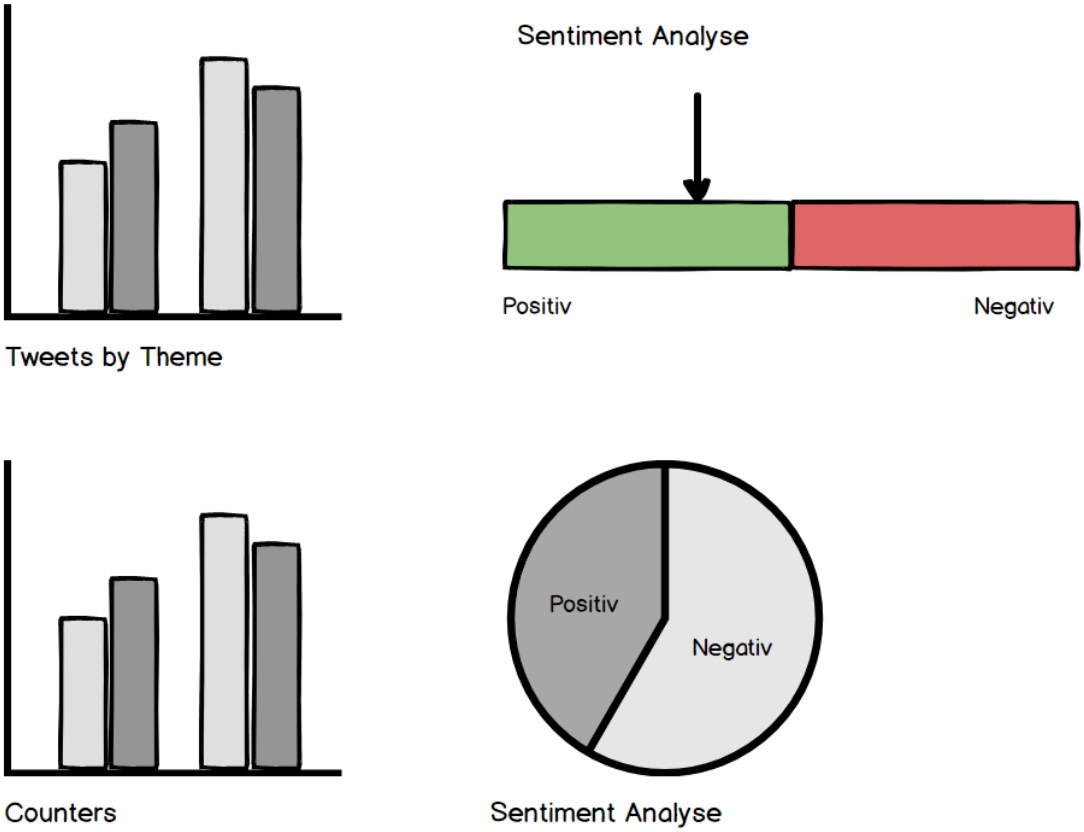

Abbildung 1: Designbeispiele Frontend

2.3 Technologien

Für das Projekt Twitter Analyse wurden zwei Technologien vorgegeben, welche wir für die Umsetzung brauchen dürfen:

- Javascript
- Java

Für uns war von Anfang an klar, dass wir das Frontend und Backend komplett trennen und nur über Schnittstellen kommunizieren wollen. Dies hat zum Vorteil, dass an beiden Seiten unabhängig weiterentwickelt werden kann. Auch die Technologie könnte ausgetauscht werden, solange die Schnittstelle gleichbleibt.

Für Java und JavaScript gibt es unendlich viele Frameworks und Libraries und es gibt immer wieder neue Entwicklungen oder neue Versionen. Da wir Beruflich beide nicht in der Entwicklung tätig sind, haben wir uns für Frameworks entschieden mit welchen wir schon in Kontakt kamen.

Im Backend haben wir uns für das Spring Framework entschieden, da wir es schon im Software Engineering gebraucht haben und somit nicht von ganz vorne beginnen mussten. Zudem erfüllte es alle unsere Anforderungen. Das Framework ermöglichte uns eine Schnittstelle zu bauen welche

beliebig erweitert werden kann. Zudem unterstützt es den Programmierer beim Verwalten der Objekte mit Annotations.

Frontend Frameworks kommen und gehen, und es gibt immer ein noch besseres oder schnelleres. Meistens ist aber mit einem Framework noch nicht alles getan, denn der Code soll dann auch noch minified und uglified werden, auf mehreren Browsern laufen usw. Wir haben uns für Ember.js entschieden, da es alle diese Webtechnologien beinhaltet:

- Templating (Handlebars)
- Data Store
- State Handling
- Routing
- Package Manager (npm)
- Bundler

Zudem stellt es eine praktisches CLI zur Verfügung um schnell bei einem Projekt Ergebnisse zu erzielen. Somit mussten wir uns im Frontend nicht lange mit dem Projekt Aufsetzen beschäftigen und konnten uns mehr aufs Programmieren konzentrieren. Ember.js ist auch online sehr gut dokumentiert. Somit kann unser Projekt da ohne grosses Einlesen weiterentwickelt werden.

2.4 Zusammenarbeit

Auch bei einem Projekt mit nur zwei Entwicklern ist die Zusammenarbeit nicht zu unterschätzen. Um nicht aneinander vorbei zu Entwickeln oder zu entscheiden, haben wir uns jeden Montag getroffen und besprochen.
Die Absprache beinhaltete:

- Rückblick was in der letzten Woche gemacht wurde
- Besprechung was als nächstes ansteht
- Aufteilung der Arbeiten für die folgende Woche
- Ein kurzes Status Update an unseren Betreuer J.Vogel

Falls Fragen oder Probleme auftauchten, sind wir auch zwischendurch zusammengesessen.
Vor allem in der Anfangsphase haben wir viel zusammen am gleichen gearbeitet bis wir die Grundstruktur hatten und aufteilen konnten.

Unsere “Sprints” haben wir in einem gemeinsamen OneNote dokumentiert:

Abbildung 2: Sprint Dokumentation

Beispiel eines Sprints:

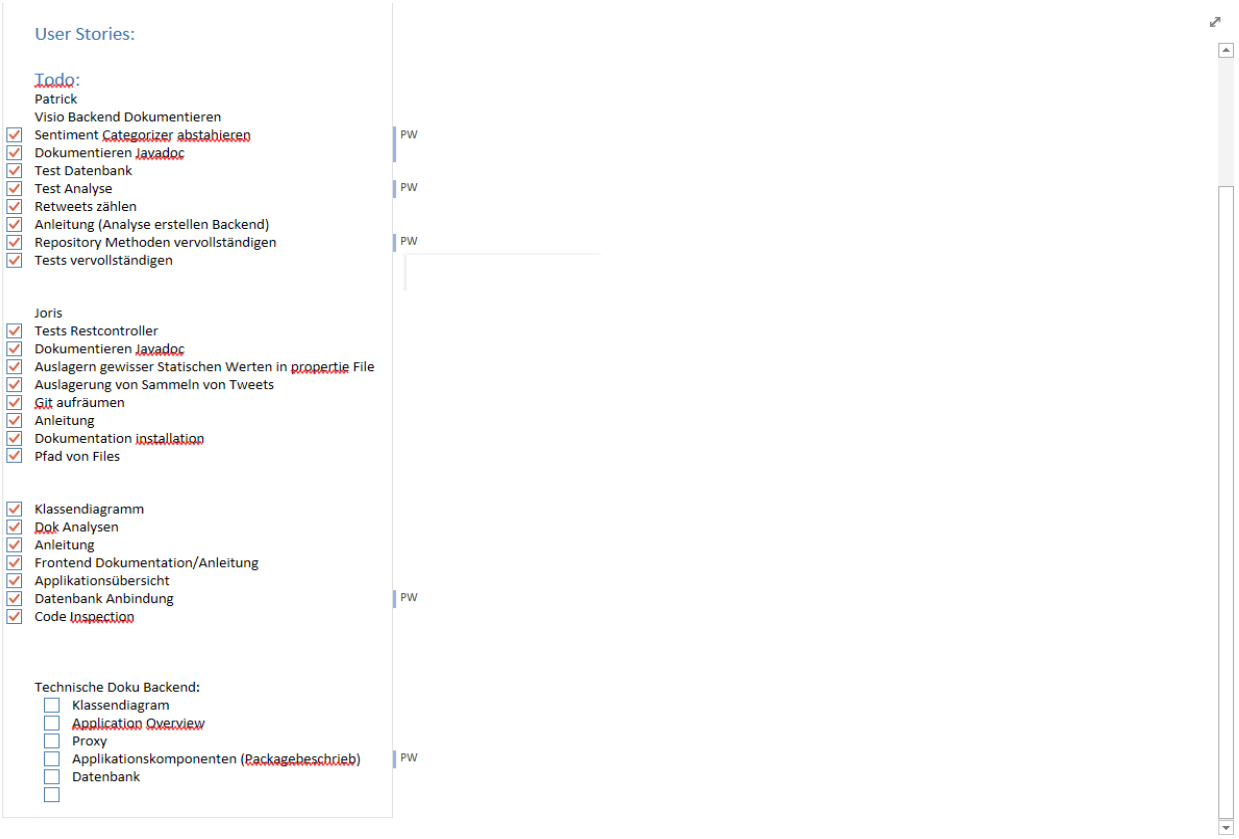

Abbildung 3: Beispiel eines Sprints

2.4.1 Version Control

Um uns mit dem Code nicht in die Quere zu kommen, haben wir mit einem Git Repository auf Github gearbeitet. Für jedes neue Feature haben wir einen neuen Branch erstellt, welcher dann nach dem Testing mit dem Master gemerged wurde. Da wir teilweise an den gleichen Klassen gearbeitet haben, gab es aber trotzdem immer wieder Konflikte.

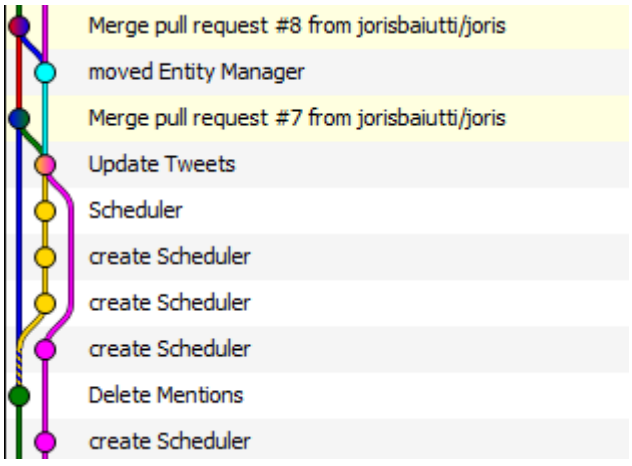

Abbildung 4: Version Control Git Repository

2.5 Testing

Wir haben für alle Analysen und Controller einen Unit Test erstellt um sicherzustellen, dass unser "Framework" die Daten im richtigen Format ausgibt. Für das Testing haben wir uns auf den Hauptteil "das Backend" konzentriert. Mit den Repository-Tests haben wir die Verbindung und die Befehle zur MSSQL Datenbank getestet.

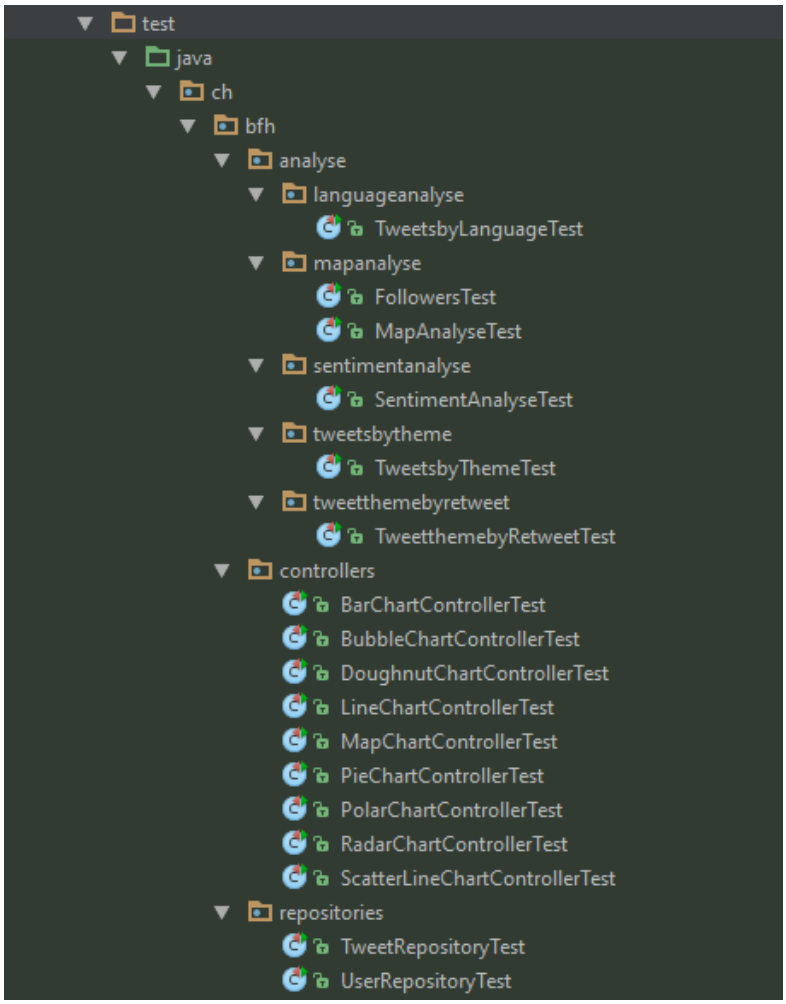

Abbildung 5: Unit Tests

Abgesehen von den Unit Tests haben wir die Installationsanleitung https://github.com/jorisbaiutti/Projekt1TwitterAnalysis/wiki/Installation getestet und durchgespielt

2.6 Dokumentation

Es war von Anfang an eine Anforderung, dass das Projekt erweiterbar sein muss. Damit ein Entwickler nicht den ganzen Code reverse engineeren muss, ist eine gute Dokumentation essentiell.
Folgendes wurde während dem Projekt dokumentiert:

- Anforderungen
- Technische Dokumentation
- Javadoc
- API
- Projektdokumentation

2.6.1 Anforderungen

Die Anforderungen wurden in einem Anforderungsdokument aus einer Vorlage von Herrn Schwab dokumentiert. Die Anforderungen beinhalten eine Priorisierung und sind unterteilt in

- Funktionale Anforderungen
- Technische Anforderungen
- Qualitätsanforderungen
- Prozessanforderungen

2.6.2 Technische Dokumentation

Die technische Dokumentation haben wir als Wiki gestaltet um die unterschiedlichen Komponenten einfacher zu strukturieren. Das Wiki liegt auf dem gleichen Github Repository wie der Code https://github.com/jorisbaiutti/Projekt1TwitterAnalysis/wiki.
Sie beinhaltet neben der Beschreibung der Lösung auch wie man das Backend und Frontend installiert und einen kleinen getting started teil um einen einfachen Einstieg in das Framework zu erhalten.
Alle Libraries und Frameworks welche wir eingesetzt haben sind schon von den Entwicklern dokumentiert. Wir haben uns auf unseren Teil konzentriert aber verlinken immer wieder auf die Dokumentation der Libraries.

2.6.3 Javadoc

An den wichtigen Stellen im Code wo nicht klar ist, was passiert haben wir die Methoden oder die Klassen dokumentiert. Methoden welche für sich selbst sprechen, haben wir nicht ausführlich dokumentiert.

2.6.4 API

Die Schnittstelle haben wir nicht selber dokumentiert, sondern wir nahmen uns eine Library zu Hilfe welche alle Endpoints ausliest und dem Benutzer eine Weboberfläche zur Verfügung stellt um das API zu untersuchen und zu testen.

2.6.5 Projektdokumentation

Das Projekt wird eigentlich durch alle unterschiedlichen Dokumentation dokumentiert. Aber um aufzuzeigen warum wir so vorgegangen sind und was unser Fazit ist, haben wir noch dieses Dokument erstellt.

3 Herausforderungen/Erkenntnisse

3.1 Hibernate/Datenbank

Für das Persistieren der Twitter-Daten haben wir eine MSSQL Datenbank gewählt.
Diese Twitter-Daten haben wir anhand unserer Objektmodelle Tweet, User, HashTag vereinfacht und mit Hibernate/JPA in die Datenbank geschrieben. Es sind div. Probleme mit Hibernate aufgetreten, z.B. beim automatischen erstellen der Datenbank, beim Update einer Entität oder auch sobald derselbe User mehrere Tweets erstellt hatte. Nach geraumer Zeit konnten wir unser Problem mit merge und persist lösen.

Anfänglich hatten wir die Datenbank lokal. Der Abgleich der Daten war jedoch nur mühsam mit Export/Import zu bewältigen, daher haben wir uns für Azure Cloud entschieden. Von Anfang an alle IP's für den Zugriff öffnen hätte uns auch einige Stunden erspart. Als dann noch die Testlizenz abgelaufen war konnten wir die Datenbank erneut mit der Schülerlizenz anlegen. Da wir sowieso noch Anpassungen an den Modellen geplant hatten, war uns der Verlust der bisher gesammelten Daten egal.

3.2 Vorkenntnisse

Wir beide sind Beruflich nicht oder nur teilweise als Entwickler unterwegs und haben noch nicht so viel Erfahrung in der professionellen Software Entwicklung. Wir brauchten daher am Anfang viel Zeit uns einzulesen und hatten nicht von Anfang an das Gesamtbild wie die Lösung dann aussehen sollte. Wir konnten mit unserem Wissen keine Architektur entwerfen die wir dann umsetzen können, somit war unser Vorgehen “try->error” bis wir wussten wie es funktionieren könnte.

3.3 Trennung von Backend und Frontend

Uns war von Anfang an klar, dass wir Frontend und Backend trennen wollen, aber was wo abgebildet wird, war uns nicht klar. In den ersten Versuchen haben wir das Model für die Grafiken auf dem Frontend gehabt. Dies widersprach sich dann mit der Anforderung “einfach Analysen zu erstellen”, da das Model auf dem Backend erstellt wurde und dann noch auf dem Frontend für das entsprechende Chart umgebaut werden musste. Wir haben uns dann dafür entschieden das Model auf dem Backend zu erstellen und dem Frontend die fertigen Daten für das Chart zu übergeben.

4 Ergebnis

Wir haben ein Framework erstellt, welches erlaubt nach gewünschten Livetweets zu suchen, diese zu Daten zu persistieren, inklusive laufendem Update eben dieser Daten. Via JSON Rest-API Schnittstelle ermöglichen wir, die erstellten und leicht erweiterbaren grafischen Analysen abzugreifen und mit JavaScript zu interpretieren. Wir haben das Frontend und das Backend völlig getrennt um einen möglichst breiten Einsatz des Frameworks zu ermöglichen.

5 Fazit

Das Projekt erlaubte uns mit vielen neuen Technologien und Frameworks in Kontakt zu kommen und das gelernte aus dem Software Engineering umzusetzen. Es gab einige Steine auf dem Weg zum jetzigen Ergebnis, aber gerade diese sind wertvolle Erfahrungen, welche uns bei nächsten Projekten helfen werden. Die Lösung welche aus dem Projekt entstanden ist, hat bestimmt Potential für weiterführende Projekte. Wahrscheinlich haben wir uns nicht an alle Conventions und Best Practices gehalten, welche vielleicht ein hauptberuflicher Entwickler automatisch umsetzt, aber wir haben das Projekt nach bestem Gewissen umgesetzt wie wir es für sinnvoll hielten. Wir denken, dass unsere Arbeit eine gute Basis für ergänzende Projekte bildet. In dieser doch relativ kurzen Zeit des Moduls Projekt1 haben wir unser Bestes gegeben und dabei sehr viel gelernt.

6 Weiterentwicklung

Mit diesem Projekt haben wir ein Fundament erstellt, welches viele Weiterentwicklungen erlaubt. Vor allem das Frontend hat noch viel Potential:

- Admin Bereich um Query Strings für die Tweetsuche zu definieren
- Charts als Code Widgets generieren.
- Analysen mit einer Grafischen Oberfläche erstellen und parametrisieren

Auch der Backend Bereich kann mit weiteren und spezifischen Analysen erweitert werden. Anpassung der Sentiment Analyse und eine themenbezogene Anpassung des Goldstandards.

7 Glossar

ID	Wort	Beschreibung
7.1	Rest API	Eine Schnittstelle, welche die Daten vom Backend (Java) dem Frontend (JavaScript) bereitstellt
7.2	JSON	JSON (JavaScript Object Notation) ist ein Datenformat zum Datenaustausch zwischen Anwendungen
7.3	Sentiment Analyse	Eine automatisierte Auswertung mit dem Zweck die positive oder negative Haltung zu erkennen

Tabelle 2: Glossar

8 Abbildungsverzeichnis

9 Tabellenverzeichnis

10Versionskontrolle

Version	Datum	Beschreibung	Autor
0.1	30.12.2017	Dokument erstellt und bearbeitet	Joris Baiutti & Patrick Wyss
0.2	07.01.2018	Dokument überarbeitet	Joris Baiutti & Patrick Wyss
0.3	08.01.2018	Dokument überarbeitet	Joris Baiutti & Patrick Wyss
1.0	19.01.2018	Dokument finalisiert	Joris Baiutti & Patrick Wyss

Tabelle 3: Versionskontrolle