Vitodurania.Net
Masterarbeit

Hochschule für Technik Rapperswil

MAS SE 2016

Autor:	Patrick Jezek
Betreuer:	Martin Seelhofer
Auftraggeber:	Vitodurania und Alt Vitodurania
Gegenleser:	Olivier Dahinden

Danksagung

Zunächst möchte ich mich an dieser Stelle bei Allen bedanken, die mich während dieser Arbeit unterstützt und motiviert haben.

- **Martin Seelhofer** für die Betreuung während der Masterarbeit
- **Olivier Dahinden** für das Gegenlesen der Arbeit und Zuhören beim Formulieren von Zusammenhängen und Sammeln von Ideen.
- **Altherren Vorstand** für die Begleitung und Inspiration während der Masterarbeit und Erlaubnis diese Umzusetzen.

Selbstverständlich will ich auch Allen nicht persönlich erwähnten danken, wie meiner Familie, die mich während der gesamten Projektdauer moralisch unterstützt haben.

Abstract

Diese Masterarbeit behandelt die Neuentwicklung der Website und mobilen Applikation des Vereins Vitodurania aus Winterthur. Die Vitodurania ist eine 1863 gegründete Mittelschulverbindung der drei Kantonsschulen Im Lee, Rychenberg und Büelrain in Winterthur in der Schweiz. Ihre Website heisst kurz VitoSite und die mobile Applikation wird VitoApp genannt. Die VitoSite ist in den letzten 14 Jahren technologisch veraltet. Gewünschte neue Features lassen sich deshalb nicht mehr einfach integrieren.

Die neue Software Architektur sollte eine Grundlage für Weiterentwicklungen legen und technologisch für ein Facelifting sorgen. Ein Schwerpunkt sollte auf die Dokumentation gelegt werden. Es ist denkbar, dass andere Vereinsmitglieder ebenfalls die VitoSite oder VitoApp in Zukunft weiterentwickeln. Zu beachten war, dass die bestehenden und neuen Daten auch von der VitoApp verwendet werden. Ein weiteres Ziel sollte die Verwendung von offenen Standards und Technologien sein. Ebenso sollte der Ansatz Offline First verfolgt werden. Teil der Arbeit war auch die Betriebsdokumentation welche beschreibt, wie die benötigten Server-Dienste installiert und konfiguriert wurden. Bis zur Fertigstellung muss ein Parallel Betrieb zur Legacy VitoSite und VitoApp möglich sein.

Als Architektur wurde ein API mit REST Schnittstelle gewählt, welche mit dem .Net Core Framework entwickelt wurde. Die VitoSite wurde mit dem Angular Framework entwickelt. Die VitoApp kann dieselben Schnittstellen wie die VitoSite verwenden. Zur Datenspeicherung wird eine dokumentenorientierte Datenbank CouchBase Server eingesetzt.

Mit dieser Arbeit konnte der Grundstein für zukünftige Weiterentwicklungen gelegt werden. Die Applikation und ihre Umsysteme sind dokumentiert. Die geplanten Use Cases konnten umgesetzt werden. Dank End 2 End Tests wird sichergestellt, dass die umgesetzten Features erhalten bleiben. Das konfigurierte CI/CD ermöglicht statische Code Analyse welche zur Qualitätsverbesserung herangezogen wird und automatisiert den Deployment Prozess.

An der Fertigstellung der neuen VitoSite kann in den nächsten Wochen weitergearbeitet werden. Die bestehende VitoApp kann die neuen oder alten Schnittstellen verwenden. Die Umstellung ist dank Parallel Betrieb nicht dringend. Die bestehenden Daten wurden übernommen. Die Datenübernahme war jedoch nicht Teil dieser Arbeit.

Das Arbeiten mit Angular hat sich als sehr produktiv herausgestellt, da es eine grosse Community hat und im Internet viele Anleitungen zu Best Practices vorhanden sind. Der nötige Zeitaufwand für das Schreiben von Dokumentation wurde unterschätzt aber aufgewendet.

Inhalt

1. Einleitung und Übersicht

1.1 Einleitung

Diese Masterarbeit behandelt das Schaffen eines Grundsteines für die Neuentwicklung der Website und mobilen Applikation der Vereine Vitodurania und Alt-Vitodurania aus Winterthur. Die Vitodurania ist eine 1863 gegründete Mittelschulverbindung der drei Kantonsschulen Im Lee, Rychenberg und Büelrain in Winterthur in der Schweiz. Die Alt-Vitodurania ist der Altherrenverband der Vitodurania.

Ich selbst bin ein Alter Herr im Altherrenverband und beauftragt, ehrenamtlich den Betrieb und die Weiterentwicklung der Website sicher zu stellen.

1.2 Ausgangslage

Die Website der beiden Vereine heisst kurz VitoSite und die mobile Applikation wird VitoApp genannt.

Die VitoSite und VitoApp wird rege von den Vereinsmitgliedern und potentiellen Beitrittskandidaten verwendet. Die Vereinsmitglieder halten auf dieser Plattform ihr Vereinsleben digital fest. Die Inhalte dienen als Archiv oder als Informationskanal für Vereinsmitglieder. Die Inhalte umfassen unter anderem:

- Das Quartalsprogramm: Ein Terminkalender mit den anstehenden Vereinsaktivitäten.
- Den News: aktuelle und archivierte Meldungen über das Vereinsleben
- Der Bildergalerie: Fotosammlung der Vereine
- Den Seiten: Informationen verschiedenster Art. Sammlungen von Protokollen, Informativen Seiten, etc.

Abbildung 1: VitoSite

Die aktuelle Website ist bereits seit 2003 im Betrieb und weist ein Alter von 15 Jahren auf. Entsprechend alt ist die verwendete Technologie. Was sehr hinderlich ist bei Erweiterungen oder im Betrieb. Gewünschte neue Features lassen sich deshalb nicht mehr einfach integrieren.

Die Website ist in PHP 4 [1] entwickelt, was leider nicht mehr unterstützt wird auf modernen Webservern.

Bei der VitoApp handelt es sich um eine Cordova / JavaScript Applikation und wurde vor 5 Jahren ins Leben gerufen. Das letzte Update erschien im letzten Dezember. Entwickelt wurde die App von Dominique Sandoz, ebenfalls ehrenamtlich.

Abbildung 2: VitoApp

Die VitoApp bietet nur einen Teil der Informationen an, welcher in der VitoSite vorhanden sind. Sie wurde auf das wesentliche reduziert und beinhaltet das Quartalsprogramm, das Adressverzeichnis und den Kantusprügel.

Auch die VitoApp könnte

1.3 Gesetzliche Rahmenbedingungen

Es gibt ein Merkblatt über den Umgang mit Mitgliederdaten in einem Verein [2], welches eingehalten werden sollte. Die Personendaten sind vertraulich und nur Mitgliedern zugänglich (Mit wenigen Ausnahmen wo nur Teile der Daten angezeigt werden).

1.4 Einschränkungen

Die Software soll auf einem Linux Server betrieben werden können, da das bestehende Hosting nur diese erlaubt. Linux Server haben sich bei der Legacy VitoSite als sehr robust im Betrieb gezeigt. Ausserdem ist bereits einiges an Know-how vorhanden.

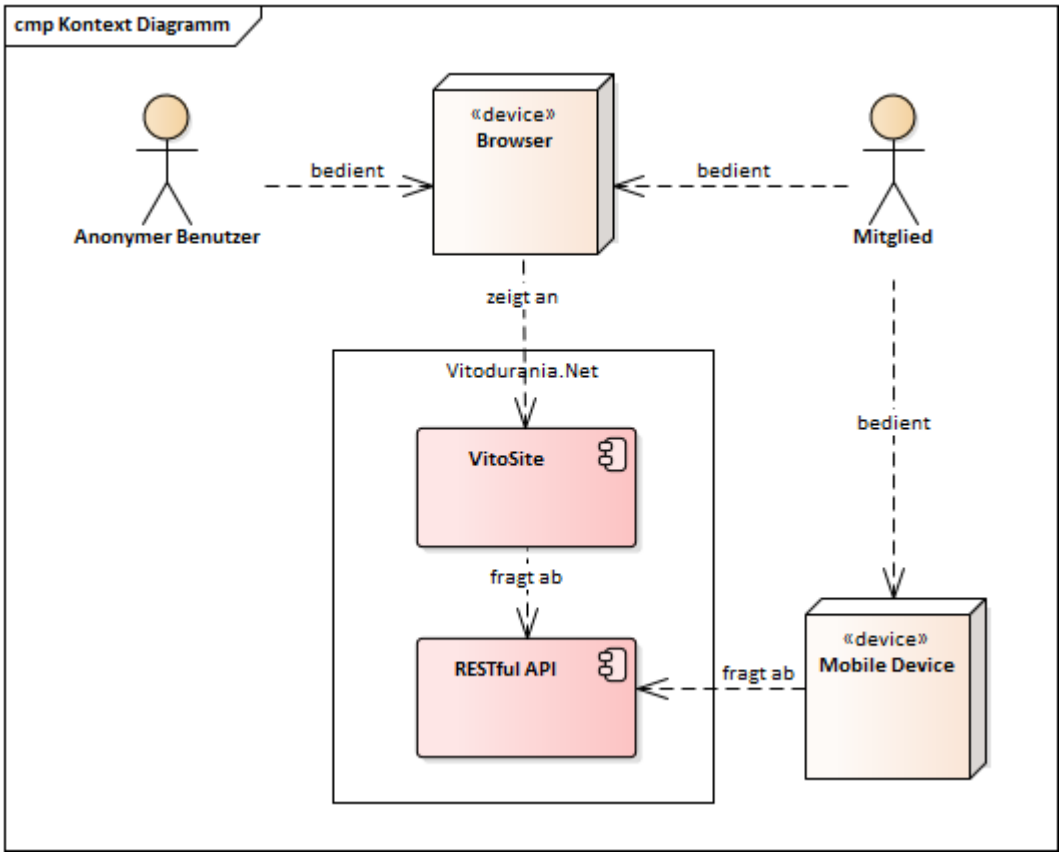
Wann immer möglich sollen Technologien verwendet werden, welche eine breite Unterstützung haben in der Community und auch Teil des Lehrstoffes an den Hochschulen sind. Somit soll sichergestellt sein, dass nicht auf veraltete Technogien gesetzt wird.

Wenn möglich sollen offene Standards eingesetzt werden.

Die Legacy VitoSite hat noch weitere Funktionen welche im Rahmen dieser Masterarbeit noch nicht umgesetzt werden.

Der Importer, welcher die Daten der VitoSite in einer konsumierbaren Schnittstelle exportiert hat, wurde im Vorprojekt fertig gestellt. Im Vorprojekt wurde auch sichergestellt, dass eine neue RESTful API diese konsumieren kann.

1.5 System Kontext



Die VitoSite beinhaltet alle Inhalte und ermöglicht das Bearbeiten dieser. Damit Inhalte auch von der VitoApp benutzbar werden wurde ein RESTful API davorgestellt.

Es wird nur zwischen 2 Benutzer Typen unterschieden:

Anonyme Benutzer

Die Gruppe die anonymen Benutzer enthält alle möglichen Benutzer welche die Website oder App besuchen und anzeigen. Sie sind dem System nicht bekannt und bekommen deshalb nur eingeschränkten Zugriff auf die Inhalte. Vertrauenswürdige Inhalten bleiben ihnen verborgen. Diese Gruppe enthält z.B. interessierte Mittelschüler welche sich über den Verein informieren wollen. Auch Lehrer oder Suchmaschinen oder auch Mitglieder anderer Verbindungen gehören zu dieser Gruppe.

Mitglieder

Logins für die VitoSite und VitoApp werden nur Vereinsmitgliedern vergeben. Abhängig von der der Funktion im Verein werden verschiedene Rechte über Rollen den einzelnen Mitgliedern vergeben.

Benutzer	Funktion / Rolle	Beschreibung
Anonymer Benutzer	Gast	Eingeschränkter lesender Zugriff auf Inhalte
Mitglied	Mitglied	Vollständiger lesender Zugriff auf Inhalte. Kein verändern von Daten erlaubt, mit Ausnahme von Teilnahme and Veranstaltungen und zurücksetzten des eigenen Passwortes
Mitglied	Aktiver	Vollständiger lesender Zugriff auf Inhalte. Rechte Inhalte zu Ändern mit Ausnahme von Adressen, Seitenstrukturen, Berechtigungen und Einstellungen des Systems.
Mitglied	Administrator	Voller Zugriff auf alle Daten. Kann Adressen ändern und Einstellungen am System vornehmen.

1.6 Ziele

Die vorliegende Masterarbeit soll den Grundstein legen, für die Neuentwicklung der bestehenden Website und mobile Applikation. Das Fundament soll auch zukünftige Weiterentwicklungen ermöglichen. Es soll eine Dokumentation entstehen, damit auch andere Mitglieder ehrenamtlich einen leichten Einstieg finden, die beiden Applikationen zu erweitern oder zu betreiben. Denkbar ist das zukünftige Mitglieder im Rahmen einer Studien Arbeit ebenfalls gewisse Erweiterungen aufgreifen.

Da der Leistungsumfang für die Masterarbeit beschränkt ist, werden jedoch nur Teile für eine Neuentwicklung herausgegriffen. Ziel ist es jedoch zu einem späteren Zeitpunkt alles komplett neu zu entwickeln. Da die Website die älteste Komponente ist, soll mit Ihr begonnen werden. Dies wurde gleich zum Projektstart vom Kunden gewünscht.

Diese Masterarbeit beinhaltet die Konzipierung und Erstellung einer Website welche die Aspekte der mobilen Applikation beinhaltet (Quartalsprogramm, Mitgliederverzeichnis und den Kantusprügel). Da die Inhalte der Website auch von einer mobilen Applikation verwendet werden, sollen alle Daten über ein RESTful API bezogen werden.

1.7 Lieferumfang

Folgende Lieferobjekte sollen am Ende des Projekts zur Verfügung stehen:

- Lauffähige Software.
  Die Website und das RESTful API wird auf dem Vereinsserver installiert
- Quelltext
  Der Quelltext ist im VSTS abgelegt. Die Rechte am Quelltext für Weiterentwicklungen wird dem Verein übertragen.
- Softwaredokumentation
  Folgende Dokumente gelten als Lieferobjekte:
    - Anforderungsspezifikation
    - Architektur und Design
    - Betriebsdokumentation

2. Planung

2.1 Projektorganisation

2.1.1 Organisationsstruktur

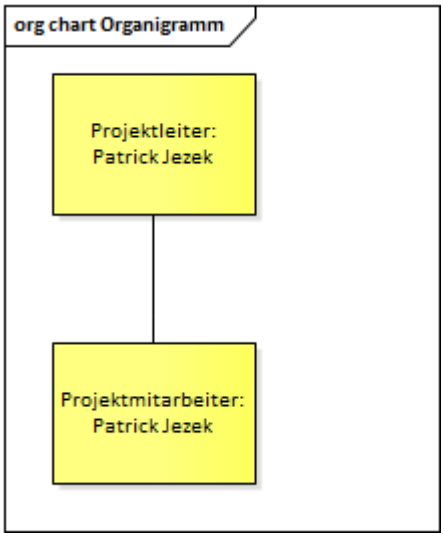


Abbildung 3: Organigramm

2.1.2 Externe Schnittstellen

Name	Funktion
Altherren Vorstand	Auftraggeber, Bewilligung des Projektes **Relevanz**: Stakeholder für neue Funktionalität
Aktivitas	Inhalt Verantwortliche der Website **Relevanz**: Anwender der Software
Florian Schneider v/o Samba	Altherren Präsident **Relevanz**: Ansprechpartner während des Vorprojektes
Remo Berger v/o Gnuss	Quästor des Altherrenverbands **Relevanz**: Ansprechpartner im Zahlungswesen
Dominique Sandoz v/o Schiller	Programmierer der aktuellen mobilen Applikation, Technischer Projektleiter **Relevanz**: Technischer Ansprechpartner
Martin Seelhofer	Betreuer der Master Arbeit: **Relevanz**: Betreuer
Olivier Dahinden	Gegenleser der Master Arbeit **Relevanz**: Lektor

2.2 Controlling

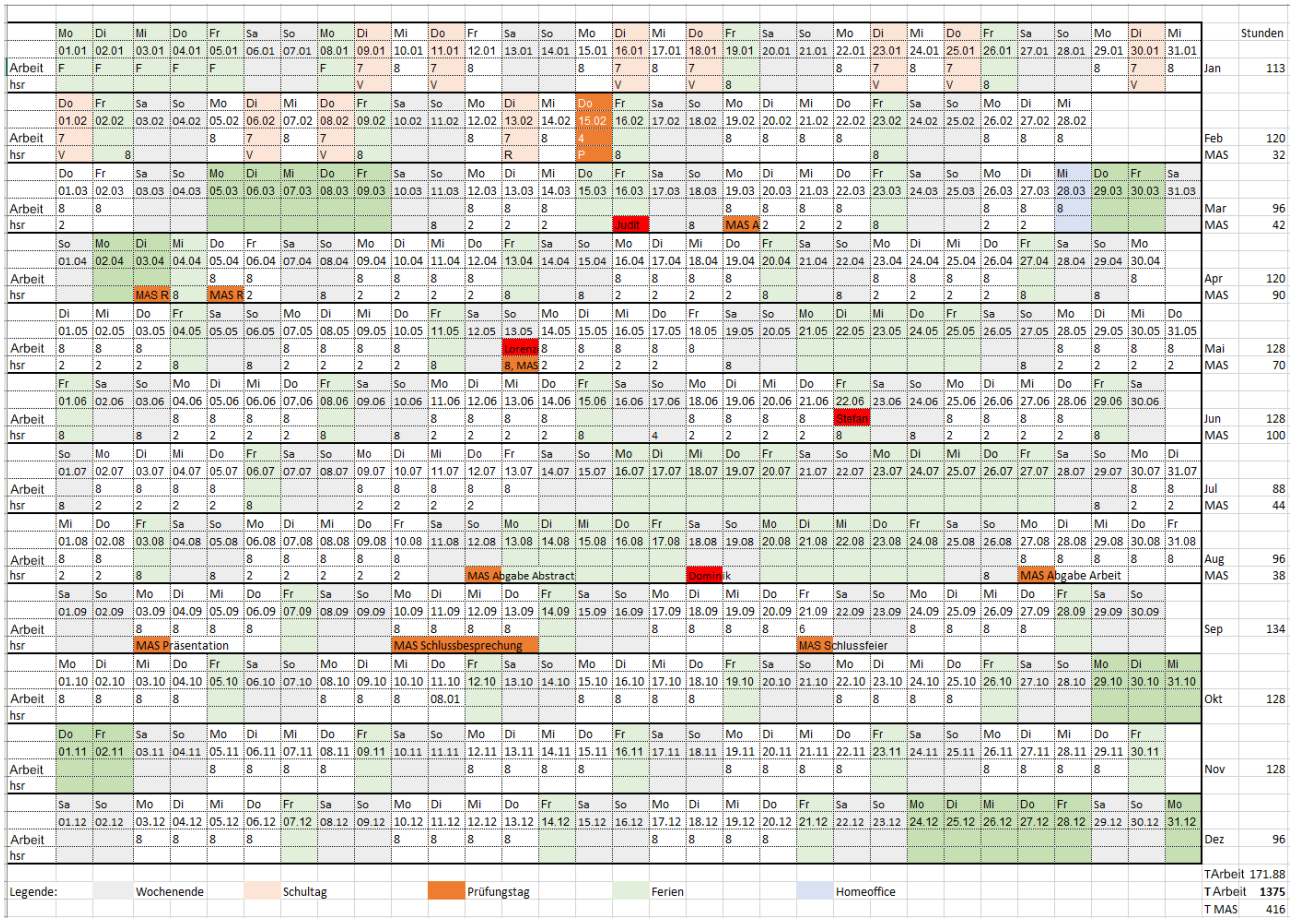
Dem Auftraggeber, der Verein Alt-Vitodurania, stellvertretend durch den Altherrenvorstand wird an den dessen Sitzungen über den Fortschritt berichtet.

Zusammen mit dem Betreuer werden Sprint Reviews am Ende eines Sprints durchgeführt.

Werden geplante Zeiten nicht eingehalten, muss der Vorstand informiert werden.

2.3 Zeitliche Planung

2.3.1 Verfügbarkeiten im Jahr 2018



	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi		Stunden
	01.01	02.01	03.01	04.01	05.01	06.01	07.01	08.01	09.01	10.01	11.01	12.01	13.01	14.01	15.01	16.01	17.01	18.01	19.01	20.01	21.01	22.01	23.01	24.01	25.01	26.01	27.01	28.01	29.01	30.01	31.01		
Arbeit	F	F	F	F	F			F	7	8	7	8			8	7	8	7				8	7	8	7				8	7	8	Jan	113
hsr									V		V					V		V	8				V		V	8				V			
	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi					
	01.02	02.02	03.02	04.02	05.02	06.02	07.02	08.02	09.02	10.02	11.02	12.02	13.02	14.02	15.02	16.02	17.02	18.02	19.02	20.02	21.02	22.02	23.02	24.02	25.02	26.02	27.02	28.02					
Arbeit	7				8	7	8	7				8	7	8	4				8	8	8	8				8	8	8				Feb	120
hsr	V	8				V		V	8				R		P	8								8								MAS	32
	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa				
	01.03	02.03	03.03	04.03	05.03	06.03	07.03	08.03	09.03	10.03	11.03	12.03	13.03	14.03	15.03	16.03	17.03	18.03	19.03	20.03	21.03	22.03	23.03	24.03	25.03	26.03	27.03	28.03	29.03	30.03	31.03		
Arbeit	8	8										8	8	8					8	8	8	8					8	8	8				Mar	96
hsr	2										8	2	2	2		Judit		8	MAS A	2	2	2	8				2	2					MAS	42
	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo					
	01.04	02.04	03.04	04.04	05.04	06.04	07.04	08.04	09.04	10.04	11.04	12.04	13.04	14.04	15.04	16.04	17.04	18.04	19.04	20.04	21.04	22.04	23.04	24.04	25.04	26.04	27.04	28.04	29.04	30.04					
Arbeit					8	8			8	8	8	8				8	8	8	8				8	8	8	8				8			Apr	120
hsr			MAS R	8	MAS R	2		8	2	2	2	2	8		8	2	2	2	2	8		8	2	2	2	2	8		8					MAS	90
	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do					
	01.05	02.05	03.05	04.05	05.05	06.05	07.05	08.05	09.05	10.05	11.05	12.05	13.05	14.05	15.05	16.05	17.05	18.05	19.05	20.05	21.05	22.05	23.05	24.05	25.05	26.05	27.05	28.05	29.05	30.05	31.05				
Arbeit	8	8	8			8	8	8	8			Lorena	8	8	8	8	8											8	8	8	8			Mai	128
hsr	2	2	2	8		8	2	2	2	2	8		8, MAS	2	2	2	2		8								8	2	2	2	2			MAS	70
	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa					
	01.06	02.06	03.06	04.06	05.06	06.06	07.06	08.06	09.06	10.06	11.06	12.06	13.06	14.06	15.06	16.06	17.06	18.06	19.06	20.06	21.06	22.06	23.06	24.06	25.06	26.06	27.06	28.06	29.06	30.06					
Arbeit			8	8	8	8			8	8	8	8	8					8	8	8	8	Stefan			8	8	8	8					Jun	128
hsr	8		8	2	2	2	2	8		8	2	2	2	2	8		4	2	2	2	2	8		8	2	2	2	2	8					MAS	100
	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di					
	01.07	02.07	03.07	04.07	05.07	06.07	07.07	08.07	09.07	10.07	11.07	12.07	13.07	14.07	15.07	16.07	17.07	18.07	19.07	20.07	21.07	22.07	23.07	24.07	25.07	26.07	27.07	28.07	29.07	30.07	31.07				
Arbeit		8	8	8	8				8	8	8	8	8																	8	8			Jul	88
hsr	8	2	2	2	2	8			2	2	2	2																	8	2	2			MAS	44
	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr					
	01.08	02.08	03.08	04.08	05.08	06.08	07.08	08.08	09.08	10.08	11.08	12.08	13.08	14.08	15.08	16.08	17.08	18.08	19.08	20.08	21.08	22.08	23.08	24.08	25.08	26.08	27.08	28.08	29.08	30.08	31.08				
Arbeit	8	8				8	8	8	8	8																	8	8	8	8	8			Aug	96
hsr	2	2	8		8	2	2	2	2	2		MAS Abgabe Abstract						Domin	k							8	MAS Abgabe Arbeit							MAS	38
	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So					
	01.09	02.09	03.09	04.09	05.09	06.09	07.09	08.09	09.09	10.09	11.09	12.09	13.09	14.09	15.09	16.09	17.09	18.09	19.09	20.09	21.09	22.09	23.09	24.09	25.09	26.09	27.09	28.09	29.09	30.09					
Arbeit			8	8	8	8				8	8	8	8				8	8	8	8	6			8	8	8	8						Sep	134
hsr			MAS Präsentation							MAS Schlussbesprechung											MAS Schlussfeier														
	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi					
	01.10	02.10	03.10	04.10	05.10	06.10	07.10	08.10	09.10	10.10	11.10	12.10	13.10	14.10	15.10	16.10	17.10	18.10	19.10	20.10	21.10	22.10	23.10	24.10	25.10	26.10	27.10	28.10	29.10	30.10	31.10				
Arbeit	8	8	8	8				8	8	8	08.01				8	8	8	8				8	8	8	8							Okt	128
hsr																																			
	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr					
	01.11	02.11	03.11	04.11	05.11	06.11	07.11	08.11	09.11	10.11	11.11	12.11	13.11	14.11	15.11	16.11	17.11	18.11	19.11	20.11	21.11	22.11	23.11	24.11	25.11	26.11	27.11	28.11	29.11	30.11					
Arbeit					8	8	8	8				8	8	8	8				8	8	8	8				8	8	8	8					Nov	128
hsr																																			
	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo					
	01.12	02.12	03.12	04.12	05.12	06.12	07.12	08.12	09.12	10.12	11.12	12.12	13.12	14.12	15.12	16.12	17.12	18.12	19.12	20.12	21.12	22.12	23.12	24.12	25.12	26.12	27.12	28.12	29.12	30.12	31.12				
Arbeit			8	8	8					8	8	8	8				8	8	8	8													Dez	96
hsr																																			
																																	TArbeit	171.88
Legende:			Wochenende			Schultag				Prüfungstag						Ferien						Homeoffice											TArbeit	1375
																																	T MAS	416

Abbildung 4: Verfügbarkeiten im Jahr 2018

Diese Planung zeigt die Arbeits-, Schul- und Freizeitplanung an. Wichtige Termine während des Projektes sind auch erfasst.

Legende:

- Total: Anzahl Arbeitsstunden (für Master Arbeit nicht relevant)
- HSR: Anzahl Masterarbeit Projekt Stunden (geplant: 416 Stunden)
- grün: nicht Arbeitszeit
- grün (dunkel): Ferien

In der Jahresplanung finden die mindestens erwarteten 375 Stunden Platz. Die Reserve oder Mehrstunden sind aber nur ca. 10%!

2.3.2 Vorbereitung Phase

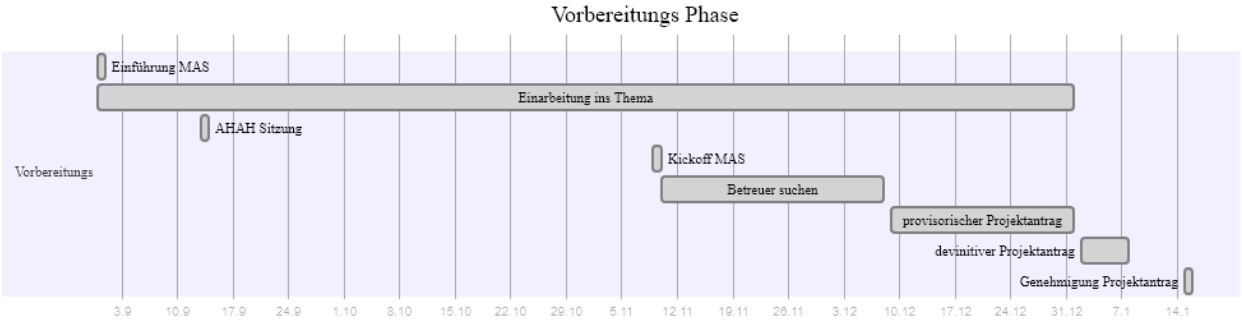


Abbildung 5: Vorbereitung Phase

Diese Phase galt der Technologiefindung. Es wurde mit der gewählten Technologie ein lauffähiger Prototyp erstellt. Ausserdem wurden die Daten aus der alten Website in ein neues Datenformat überführt.

- Einarbeitung in das Tool Enterprise Architekt, mit Hilfe welchem viele Diagramme in dieser Arbeit gezeichnet wurden.
- Einarbeitung in .Net Core. Es entstand ein API Prototyp.
- Einarbeitung in Angular. Es entstand ein Website Prototyp.
- Einarbeitung in Xamarin. Es entstand ein Mobile App Prototyp. (Wird aber in diesem Projekt noch nicht angewandt).
- Einarbeitung in Visual Studio Team Services: Es wurde ein simples Contineous Integration aufgesetzt, welcher die Dokumentation deployet und den Quelltext im SonarCube prüft.
- Einarbeitung in Azure: Es wurde ein SonarCube (Code Smell Analysis Tool) installiert.
- Infrastruktur: Die Legacy Applikation wurde auf den neuen Server migriert.
- Infrastruktur: eigener Laptop mit einer Entwicklungsumgebung aufgesetzt. In einer Virtuellen Maschine läuft die gleiche Software wie auf dem Ziel Server. Somit kann offline entwickelt werden.

2.3.3 Durchführung Phase

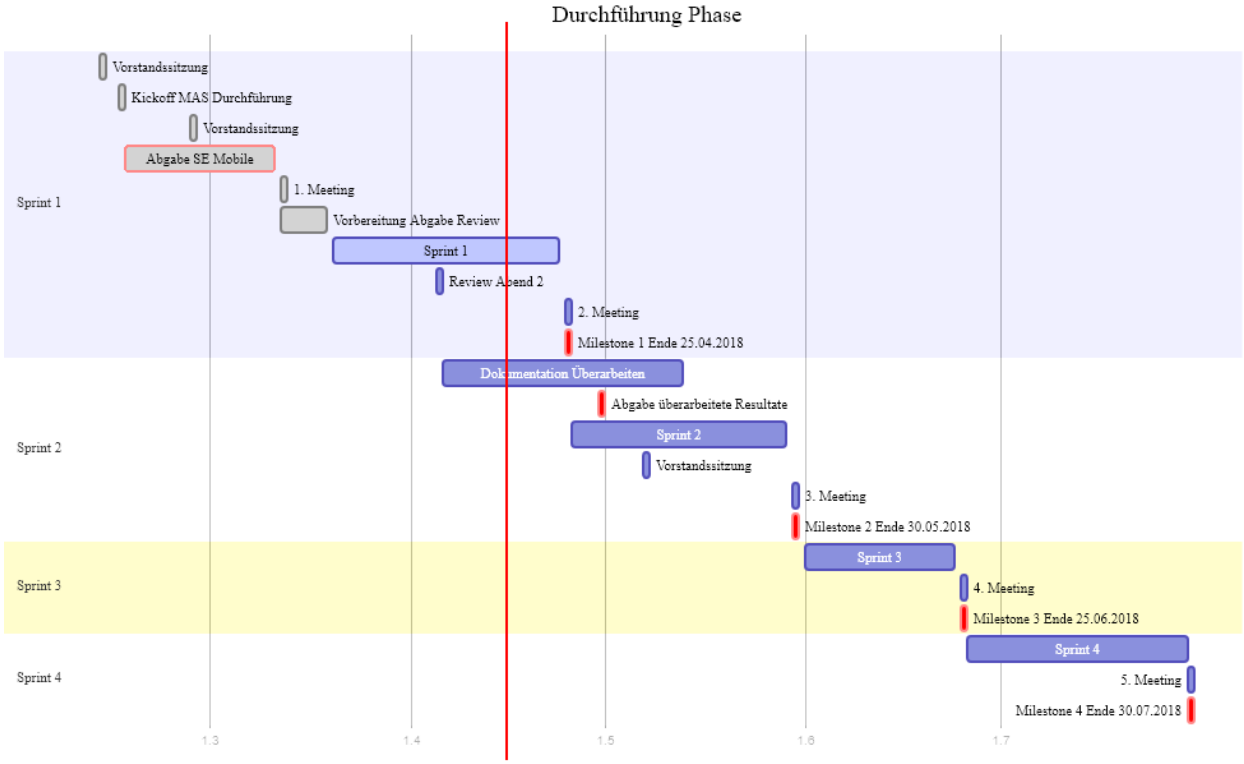


Abbildung 6: Durchführung Phase

Die rote vertikale Linie ist das aktuelle Datum zum Zeitpunkt der Bilderstellung.

Leider wurde gleich zu Beginn des Projektes ein Verzug von 24 Kalendertagen (26 Plan Stunden) gemacht, aufgrund einer bewerteten Arbeit für den 3. CAS. Im Plan ist dieser Verzug als rot umrandeter Balken ausgewiesen.

Der Vorliegende Plan wurde wegen dieser Verzögerung angepasst.

Darauffolgend wurde 1 Woche (16 Planstunden) in das Erstellen der Abgabe für das Review eingeplant.

Die verbleibende Zeit wurde in 4 Sprints aufgeteilt welche je ca. einen Monat dauern.

Treffen mit dem Betreuer wurden an deren Ende gelegt, damit eine Sprint Retrospektive gemacht werden kann.

Am Ende jeder Phase ist ein Meilenstein gesetzt (roter Balken).

2.3.4 Abschluss Phase

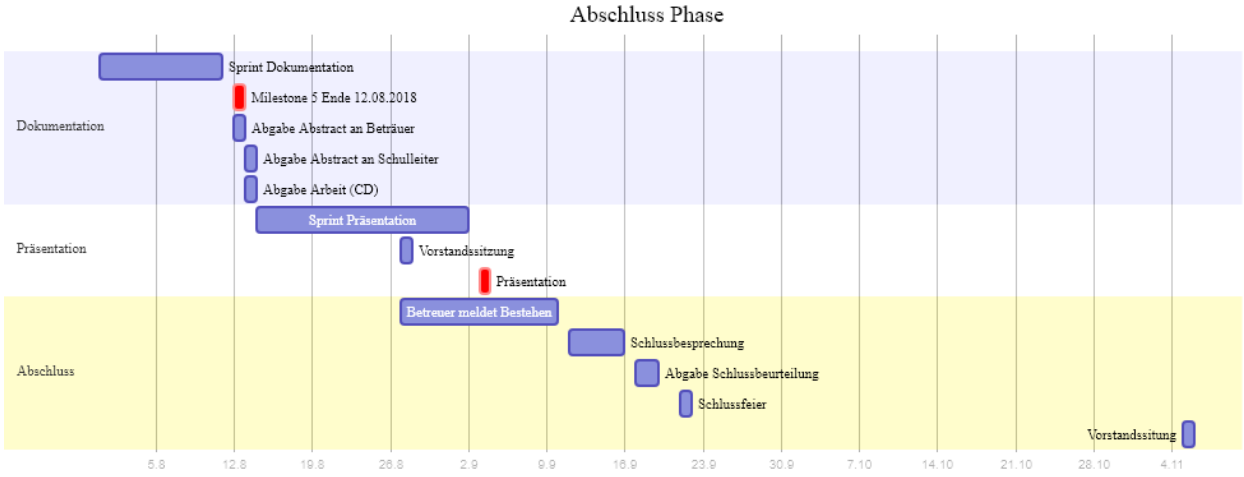


Abbildung 7: Abschluss Phase

Die Abschluss Phase ist der Dokumentation, Abgabe des Projektes und der Schlusspräsentation gewidmet.

Ein Deployment Datum steht noch nicht fest. Es findet vermutlich im Oktober statt.

2.3.5 Meilensteine

Meilenstein Datum	Beschreibung Ergebnis
Meilenstein 1 25.04.2018	**API / Infrastruktur / Betriebsdokumentation** Ergebnis: - Test und Produktive Umgebung können automatisch deployet werden. Betriebsdokumentation enthält die nötigen Abschnitte. - API aus dem Prototyp werden übernommen. Swagger Dokumentation wird vom Quelltext erstellt. - Website aus dem Prototyp wird übernommen. Es steht ein Grundgerüst für die Website bereit. - Dokumentation Projektplan und Anforderungsspezifikation sind fertig gestellt.
Meilenstein 2 30.05.2018	**Inhalte** Ergebnis: - UC6: An einem Anlass im Quartalsprogramm teilnehmen ist fertiggestellt. - UC7: Quartalsprogramm anzeigen ist fertiggestellt. - UC8: Quartalsprogramm als ICS anzeigen ist fertiggestellt. - UC9: Kantusprügel anzeigen ist fertiggestellt. - Inhalte Kantusprügel ist fertiggestellt.
Meilenstein 3 25.06.2018	**Integration API und Website** Ergebnis: - Member Use Cases sind fertiggestellt. - UC1: User kann sich einloggen. - UC2: User kann sich ausloggen. - UC3: Passwort vergessen Formular funktioniert - UC4: Adresslisten können angezeigt werden (Kontaktseiten). - UC5: Adressänderung melden Formular funktioniert. - User erhält die zugewiesene Rolle und kann für Schreibende Zugriffe ins API identifiziert werden.
Meilenstein 4 30.07.2018	**Optionale Use Cases** Möglichst viele der optionalen Use Cases sollten umgesetzt werden. Da diese jedoch optional sind, können diese auch ausgelassen werden. Ergebnis: - UC10: Mitglieder lassen sich verwalten. - UC11: Adressen lassen sich suchen. - UC12: Inhalte Quartalsprogramm lassen sich verwalten - UC13: Inhalte Kantusprügel lassen sich verwalten.
Meilenstein 5 12.08.2018	Ziel ist es die **Dokumentation** fertig zu stellen Ergebnis: - Dokumentation wurde fertig gestellt und ist eingereicht.

Meilenstein Datum	Beschreibung Ergebnis
Präsentation 03.09.2018	Ziel ist es die **Präsentation** erfolgreich abzuhalten. Ergebnis: - Präsentation wurde fertig gestellt. - Projekt wurde vorgestellt. - MAS ist bestanden.

2.4 Besprechungen

Besprechungen mit dem Betreuer wurde jeweils auf Ende eines Sprints Reviews gelegt.

Teilnehmende Personen: Betreuer und Projektteam

Besprechungen mit dem Kunden wurden auf die AH-Sitzungen festgelegt.

Teilnehmende Personen: AH-Vorstand und Projektteam

2.5 Risikomanagement

2.5.1 Risiken

Nr.	Titel	Beschreibung	Max. Schaden [h]	Eintritts-wahrschein-lichkeit	Gewichteter Schaden	Vorbeugung	Verhalten beim Eintreten
R1	Lieferobjekte Zeitverzug	Die geplanten Lieferobjekte können nicht fristgerecht geliefert werden.	10	5%	0.5	Zeitplanüberwachung der Meilensteine und den enthaltenen Arbeits-paketen. Deployment ist noch nicht festgelegt und kann noch verschoben werden	Verzug kommunizieren und das Lieferobjekt fertigstellen.
R2	Krankheit	Planung kann Kranktage nicht einplanen.	24	20%	4	Genügend Schlaf und Freizeit.	Kompensation mit Ferien Zeiten
R3	Technische Probleme	Kein Vorwärtskommen in der Lösungsfindung	5	10%	2	Betreuer Support, Vorprojekt durchführen um technische Risiken zu minimieren.	Rat holen bei Arbeitskollegen welche mit der Technologie vertraut sind.

2.5.2 Umgang mit Risiken

Um das Risiko **R1** zu vermindern wurden die Lieferobjekte in zwei Gruppen unterteilt und somit optionale Ziele gesetzt, welche bei Zeitreserven umgesetzt werden. Kommt das Projekt in Verzug können diese gestrichen werden.

Wie in Punkt 2.3.1 aufgezeigt sind nur 10% Reserven eingeplant. Vor allem das Risiko **R2** kann zu massiven Verzögerungen führen. Es können noch Zeiten von Ferien Tagen bezogen werden, welche aber wieder das Risiko **R2** fördern. Dies muss bei Einträten erneut abgewogen werden.

Das Risiko **R3** wird tief eingeschätzt, da im Bereich Webtechnologien dem Projektteam ein grosser Erfahrungsschatz vorliegt. Ebenso verfügt es um ein grosses Netzwerk von Kollegen, welchen im selben Bereich tätig sind. Es kann damit gerechnet werden, dass in sehr kurzer Zeit Hilfestellung bei offenen Fragen bezogen werden kann.

Glücklicherweise ist ein Verzug nicht kritisch, da die alte VitoSite und VitoApp noch weiterhin betrieben werden können.

Da aber die Lieferung versprochen wurde, muss diese ehrenamtlich fertig gestellt werden. Es resultiert nur eine Verzögerung, welche keinen Wirtschaftlichen Schaden für den Auftraggeber aufweist.

2.6 Arbeitspakete

ID	Work Item...	Title	State	Board Column	Area Path	Iteration Path	Tags
17	Epic	API	Active	Analyze	Vitodurania.Net	Vitodurania.Net	MAS
22	Feature	API - Inhalte	Active	Analyze	Vitodurania.Net	Vitodurania.Net	MAS
19	Feature	API - JWT	Active	Analyze	Vitodurania.Net	Vitodurania.Net	MAS
20	Feature	API - Login	New	Backlog	Vitodurania.Net	Vitodurania.Net	MAS
21	Feature	API - Quartalsprogramm teilnehmen	New	Backlog	Vitodurania.Net	Vitodurania.Net	MAS
18	Feature	API - Swagger	Active	Analyze	Vitodurania.Net	Vitodurania.Net	MAS
11	Epic	Infrastruktur	Active	Analyze	Vitodurania.Net	Vitodurania.Net	MAS
16	Feature	Infrastruktur - Backup Strategie	Active	Analyze	Vitodurania.Net	Vitodurania.Net	MAS
15	Feature	Infrastruktur - Couchbase	Active	Analyze	Vitodurania.Net	Vitodurania.Net	MAS
119	Feature	Infrastruktur - Datenübernahme	New	Backlog	Vitodurania.Net	Vitodurania.Net	MAS
118	Feature	Infrastruktur - Deployment	New	Backlog	Vitodurania.Net	Vitodurania.Net	MAS
12	Feature	Infrastruktur - Legacy System	Closed	Done	Vitodurania.Net	Vitodurania.Net	MAS
13	Feature	Infrastruktur - Nginx Konfiguration	Active	Analyze	Vitodurania.Net	Vitodurania.Net	MAS
45	Epic	Website	Active	Analyze	Vitodurania.Net	Vitodurania.Net	MAS
78	Feature	Website - Kantusprügel	Active	Analyze	Vitodurania.Net	Vitodurania.Net	MAS
51	Feature	Website - Mitglieder	Active	Analyze	Vitodurania.Net	Vitodurania.Net	MAS
66	Feature	Website - Quartalsprogramm	Active	Analyze	Vitodurania.Net	Vitodurania.Net	MAS

Abbildung 8: Arbeitspakete

2.7 Infrastruktur

Bereich	Werkzeug	Version
Terminplanung, Dokumentation, Protokolle	MS Office	2016
Source Code Verwaltung	VSTS [3]	visualstudio.com
Bugtracking	VSTS [3]	visualstudio.com
Buildserver	VSTS [3]	visualstudio.com
Code Quality	SonarQube [4] auf Ubuntu Linux [5]	16.04
Datenbank	Couchbase [6]	5.0
Webserver	Nginx [7] auf Ubuntu Linux [5]	14.04
Diagramme	Enterprise Architekt [8]	13.5
Gantt Diagramme	mermaid [9] [10]	Aktueller Master
Entwicklungsumgebung	Rider [11]	2017.3

2.8 Qualitätsmassnahmen

Massnahme	Zeitraum	Ziel
Continuous Integration	Gesamte Projektlaufzeit	Qualität des Quelltextes steigern, frühzeitige Erkennung von nicht buildbarer produktiver Umgebung. Automatisches Deployment.
Komponententest	Gesamte Projektlaufzeit	Qualität des Quelltextes steigern.
Integrationstest	Gesamte Projektlaufzeit	Erstellte Komponenten verhalten sich auf dem Zielsystem in erwarteter Qualität.
Systemtest	Gesamte Projektlaufzeit	Einhaltung der erwarteten Features.
Abnahmetest	Am Ende des Projektes	Erlaubnis die alte Software mit dem neuen zu ersetzten.
Anforderungsreview	Gesamte Projektlaufzeit	Einhaltung der erwarteten Qualität.

Massnahme	Zeitraum	Ziel
Sprint Review	Am Ende einer Iteration	Qualität des gesamten Produktes steigern. Verbessern des Prozesses.
Präsentationen	An AH-Sitzungen	Zwischenstand dem Kunden zeigen.

2.8.1 Dokumentation

Die Dokumentation befindet sich beim Quelltext und ist somit auch in der Versionsverwaltung enthalten.

Die Dokumentation wird unter der URL: https://docs.vitodurania.ch automatisiert und passwortgeschützt publiziert.

Die Qualität wird durch das Review sichergestellt. Zusätzlich zum geplanten Review Abend im Rahmen der Masterarbeit finden Reviews mit dem Betreuer und dem Kunden stat. Für die Lesbarkeit hilft ein freiwilliger Lektor.

2.8.2 Projektmanagement

Als Projektmanagement Tool wird das VSTS eingesetzt. Es erlaubt Epics, Features und Stories zu verwalten. Hier werden auch aktuelle Work Items auf Sprints zugeteilt und deren Fortschritt nachgepflegt. Der URL lautet: https://pjezek.visualstudio.com/Vitodurania.Net

Verfügbarkeiten werden im MS Excel visualisiert und liegt diesem Dokument bei.

Die Sprint Planung (Dauer) und Meilensteine wurden mit Hilfe von Mermeid erstellt und sind in diesem Dokument enthalten.

2.8.3 Entwicklung

Der Quelltext ist im VSTS enthalten. Die URL lautet: https://pjezek.visualstudio.com/Vitodurania.Net

2.8.3.1 Tätigkeiten im Software Engineering:

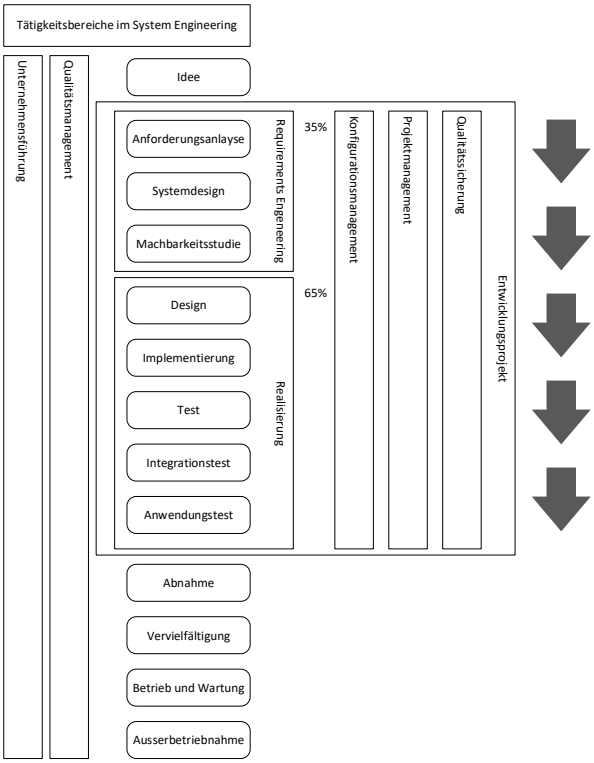


Abbildung 9: Tätigkeiten im Software Engineering

Hier kurz eine Skizze der allgemeinen Tätigkeiten im Software Engineering. Es zeigt, Tätigkeiten welche durchlaufen werden.

2.8.3.2 Gewähltes Vorgehen

Als Vorgehens Model wurde **Kanban** gewählt. Es ist ein sehr schlankes Model welches aus nur 4 Prinzipien

- Start with existing process
- Agree to pursue incremental, evolutionary change
- Respect the current process, roles, responsibilities and titles
- Encourage acts of leadership at all levels

und 5 Praktiken besteht.

- Visualize the workflow
- Limit Work-in-Progress
- Monitor, measure and optimize workflow
- Make Process Policies Explicit
- Improve Collaboratively

Die letzte Praktik gestaltet sich schwierig bei einer Teamgrösse von 1. Es zählt die Zusammenarbeit mit den Stakeholdern und dem Betreuer.

In Absprache mit dem Betreuer wurde ebenfalls eine Sprint Retrospektive - im Sinne einer Sprint Retrospektive aus der Agilen Methode SCRUM - nach jeder abgeschlossen Iteration abgemacht.

Mehr über Kanban kann im Internet [12] nachgelesen werden.

Die zentrale Rolle nimmt das Kanban Board selbst ein. Auf ihm sind die Arbeitspakete als Karten visualisiert. Das Board ist in Bereiche (welche durch Swimmlines getrennt werden) unterteilt. Der Übergang von einem Bereich in den nächsten hat eine Definition of Done (DoD) welche hier kurz beschrieben wird.

- **Backlog**
  Hier liegen Arbeitspakete welche in dem Anforderungsdokument beschrieben wurden. Sie sind in der Reihenfolge ihrer Priorität angeordnet. Es deckt die Tätigkeiten der Anforderungsanalyse und Machbarkeitsstudie ab.
- **Analyze**
  Arbeitspakete angeschaut und in die bestehende Architektur eingegliedert werden. Das Arbeitspacket soll diesen Bereich erst verlassen, wenn es im Architektur und Design Dokument eingearbeitet wurde. Es deckt die Tätigkeiten des Systemdesigns und Software Architektur (Design) ab.
- **Develop**
  Wenn aus den vorausgegangen Bereichen klar ist was wie umgesetzt werden soll, kann es implementiert werden. Es deckt die Tätigkeiten der Implementierung ab.
- **Test**
  In diesem Bereich wird verifiziert, dass die geforderten Anforderungen umgesetzt wurden. Spätestens hier sollten auch die Integrations Tests automatisiert werden, damit sichergestellt wird, dass die Gelieferte Software auf dem Zielsystem integrierbar ist ohne Regressionen aufzuweisen.
- **Done**

2.8.3.3 Code Reviews

Leider kann mit einer Teamgrösse von einer Person nicht ein klassisches Code Review durchgeführt werden.

Es werden jedoch zwei Tools eingesetzt welche Guidelines und bekannte Probleme in der Qualität von Quelltexten finden können.

Die Entwicklungsumgebung beinhaltet Code Analyse [13], welches den Quelltext bereits bei der Eingabe prüft.

Der Buildserver kann automatisch den Quelltext auf selbst gehosteten SonarQube prüfen.

2.8.3.4 Code Style Guidelines

Zwei Code Style Guidelines sind für dieses Projekt wichtig:

Für c# gelten diejenigen von Microsoft:
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions

Für JavaScript diejenigen von Google:
Der URL lautet: https://angular.io/guide/styleguide

Beide Coding Guidelines sind in der verwendeten IDE enthalten und können über den Menüpunkt Code -> Reformat Code aufgerufen werden.

2.8.4 Testen

2.8.4.1 Komponententest

Komponenten werden, wenn sinnvoll, getestet. Die Ausführung geschieht vollautomatisch.

2.8.4.2 Integrationstest

Der Fokus liegt auf der Integration von Website, API und der persistierenden Schicht. Abdeckungsgrad soll den geforderten Features entsprechen.
So wird sichergestellt, dass die Konfigurationen auf dem Zielsystem richtig gesetzt sind und die Systeme sich richtig ansprechen. Dieser Test soll vollautomatisch ausgeführt werden.

2.8.4.3 Systemtest

Die geforderten Features werden getestet, damit sichergestellt wird, dass die gelieferte Software auf dem Zielsystem läuft Dies soll vollautomatisch funktionieren.

2.8.4.4 Abnahmetest

Hier kann auf den Systemtest zurückgegriffen werden um die Features zu Zeigen. Zusätzlich sollen aber die Inhalte alte VitoSite mit der neuen VitoSite verglichen werden, um zu sehen, dass die Daten übernommen wurden:

- Die nächsten Einträge im Quartalsprogramm stimmen überein.
- Die Kontaktseiten zeigen dieselben Personen / Adressen
- User kann sich mit seinem bekannten Passwort anmelden.
- Anfordern eines neuen Passwortes funktioniert
- Melden einer Neuen Adresse funktioniert
- Anmelden / Abmelden an einen Anlass funktioniert

3. Anforderungen

3.1 Beschreibung

3.1.1 Vision

Die bestehende Website mit zeitgerechter Technologie neu umzusetzen. Dabei Wert legen auf Dokumentation, damit Erweiterungen von anderen Personen möglich werden. Zur Dokumentation gehört auch die Betriebsanleitung, die beschreibt, wie das System reproduzierbar betrieben und konfiguriert wird. Das Design kann von der bestehenden Website übernommen werden. Die bestehenden Daten sollten erhalten oder erweitert werden.

3.1.2 Funktionsblöcke

Das Produkt lässt sich auf drei wesentliche Komponenten Aufteilen:

- **RESTFul API**
  Ein API mit welchem sich die Inhalte verwalten und abfragen lassen.
  Als Kommunikation wird das Hypertext Transfer Protocol (HTTP) für sichere Kommunikation (HTTPS) Protokoll [4] verwendet.
  Das Daten Austauschformat ist JavaScript Object Notation JSON [5]. So wird sichergestellt, dass die Inhalte unabhängig von der Präsentation verwaltet werden können
  Die Inhalte Kantus, Adressen und Veranstaltungen sollen im API abrufbar sein.
  Mitglieder können am API Authentifiziert und Autorisiert werden.
- **VitoSite** (Webseite)
  Dies ist die Benutzerschnittstelle welcher der Benutzer über einen Browser sehen wird.
  Die Seiten, welche die Adressen anzeigen (Aktivitas Seite, AH-Vorstand Seite, Kontaktadressen) werden im bisherigen Design neu implementiert.
  Das Quartalsprogramm mit dem Widget auf der Startseite werden im bisherigen Design neu implementiert.
  Der Kantusprügel wird um die Lieder Texte erweitert zum bisherigen MP3 streaming / download.
- **Shared Library**
  Mit Hilfe dieser können Programmfunktionen zwischen den verschiedenen Komponenten wiederverwendet werden. Ein Beispiel sind Models welche die Daten repräsentieren und auch vom Importer (s.u.) verwendet werden.

Vom Projekt unabhängige – zum Teil bereits bestehende – Komponenten.

- **Importer**
  Importiert die Daten aus dem Legacy System in JSON Dokumente. Es stellt sicher, dass die Daten auch nach einem initialen Import wieder zu jeder Zeit neu importiert werden können und bereits importierte Daten geupdatet werden.
- **VitoApp** (mobile Applikation)
  Hierbei handelt es sich um eine zweite Benutzerschnittstelle, welche für mobile Geräte optimiert ist.
  Die neusten Veranstaltungen werden im Quartalsprogramm dargestellt.
  Es können die Adressen von Vereinsmitgliedern angezeigt und gesucht werden.
  Im Kantusprügel können die Texte zu den im Verein gesungen Liedern nachgeschaut werden.
- **Legacy VitoSite**
  Die bestehende Website mit allen Inhalten in einer MySQL Datenbank.

3.1.3 System Kontext

Folgendes Kontextdiagramm zeigt die für das Produkt relevanten Umsysteme und Akteure. Die erkannten Funktionsblöcke sind eingezeichnet. Das System wird nicht als Blackbox gezeichnet.

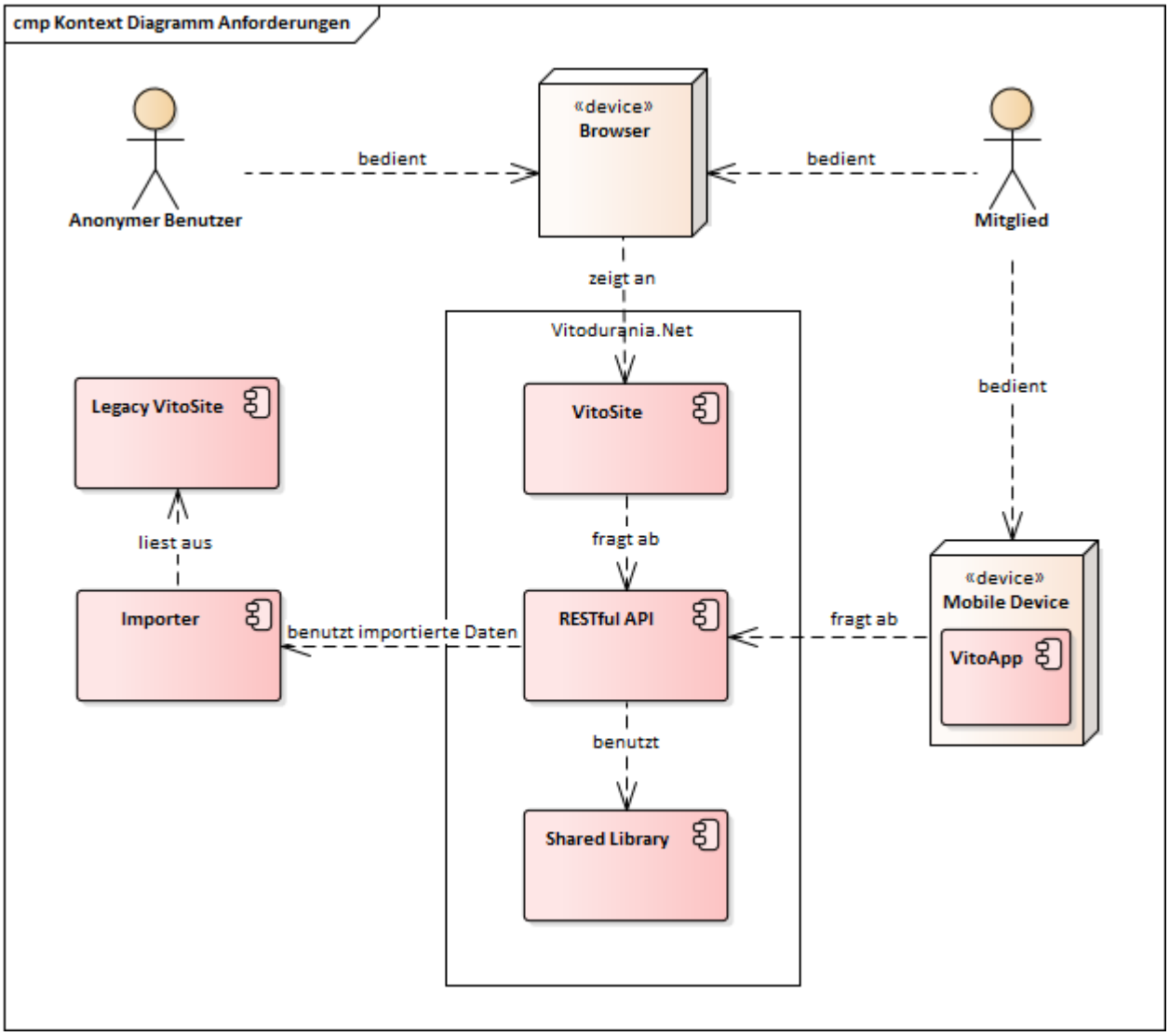


Abbildung 10: Kontextdiagramm

3.2 Übersicht der Use Cases

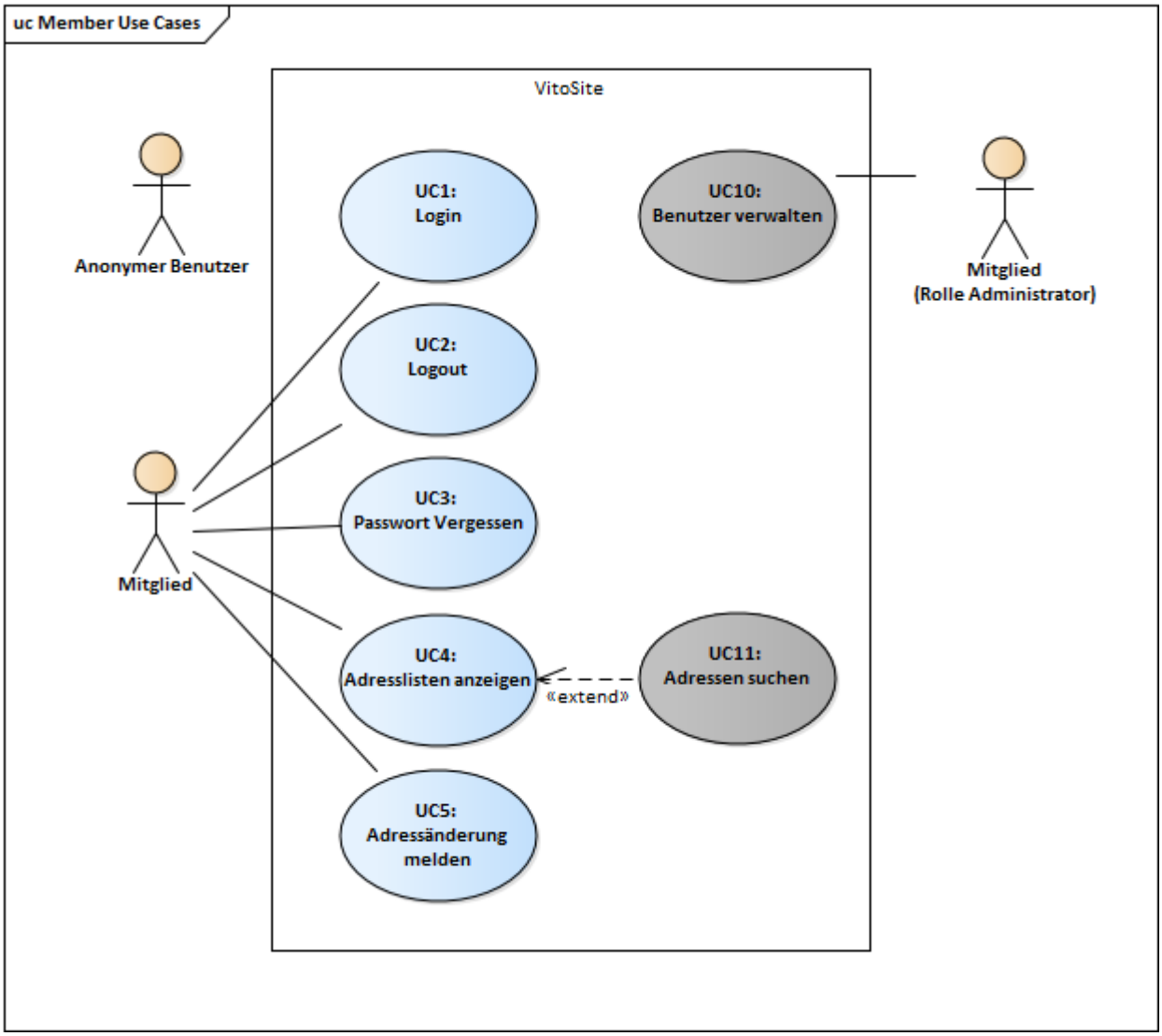


Abbildung 11: Member Use Cases

Im Diagramm ist keine Linie vom Anonymen Benutzer zum UC4 gezeichnet, obwohl er zwar Adresslisten einsehen kann, nicht jedoch nach Adressen suchen kann.

Generell wird in den nachfolgenden Use Case Beschreibungen vom Akteure Anonymer Benutzer gesprochen, wenn er eine Seite an navigieren kann, ohne eingeloggt zu sein und somit der Benutzer dem System nicht bekannt sein muss.

Wir in einer Use Case Beschreibung vom Mitglied gesprochen, wird in den Vorbedingungen hingewiesen, ob der Akteure eingeloggt sein muss.

Generell gilt die Einschränkung aus Kapitel 1.3, dass besonders schützenswerte Daten anonymen Benutzern nicht angezeigt werden dürfen. Das gilt beim vollen Umfang der VitoSite auch, auf gewisse Bereiche die noch nicht in diesem Projekt umgesetzt werden. Beim Akteur Mitglied wird auch der Zugang anhand seiner hinterlegten Rolle eingeschränkt. In den Use Case mit Akteure Mitglied (Rolle Administrator) gekennzeichnet.

Blaue Use Cases sind Pflicht und graue Use Cases sind optionaler Teil dieser Masterarbeit. Die Farben werden auch in den Use Case Beschreibungen angewendet.

Teilweise wurden Screenshots der alten VitoSite beigefügt. Sie dienen als Referenz für das Design. Dieses muss jedoch nicht 100% pixelgenau übernommen werden! Sie dienen auch der Veranschaulichung des Use Cases.

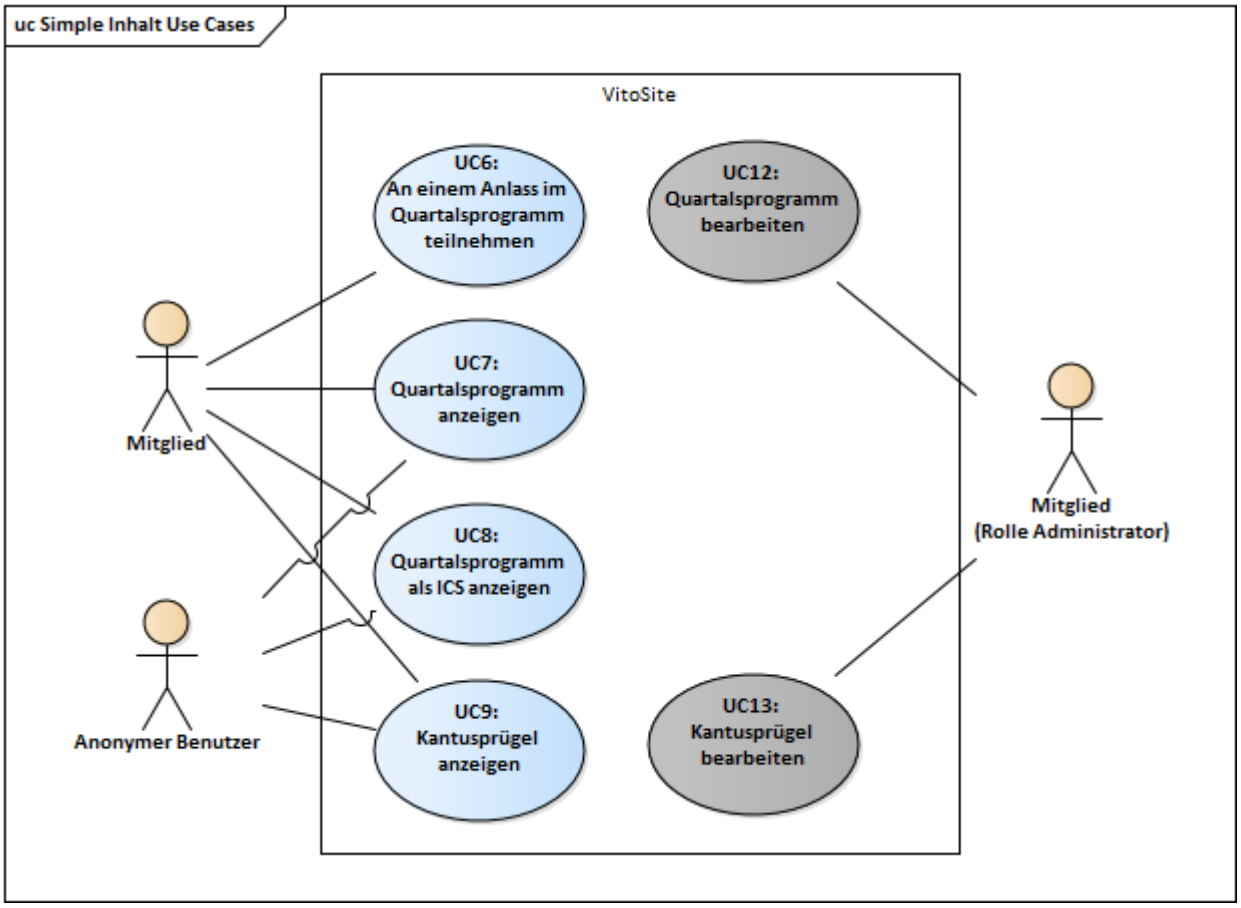
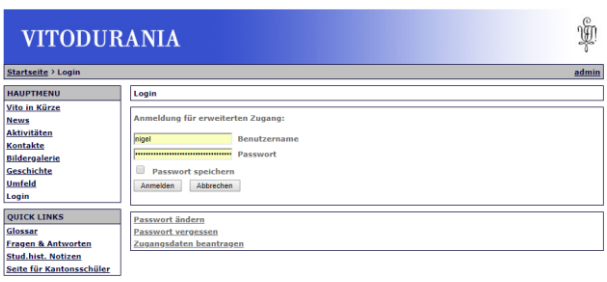


Abbildung 12: Inhalt Use Cases

3.2.1 UC1: Login

Bezeichner	UC1
Use Case Name	Login
Akteure	Mitglied
Vorbedingung	Mitglied ist nicht eingeloggt.
Ereignis Fluss	1. Das Mitglied klickt auf den Login Button 2. Das System zeigt den Login Dialog an. 3. Das Mitglied authentifiziert sich mit seinem Login Namen und seinem Passwort. 4. Das System erstellt eine neue Session für das Mitglied.  **Abbildung 13: Seite Login** https://www.vitodurania.ch/index/1000

Nachbedingung	Das Mitglied ist angemeldet und erhält die Ihm zugewiesene Rolle und den dazugehörigen Rechten. Eine neue Session wurde für das Mitglied erstellt.

3.2.2 UC2: Logout

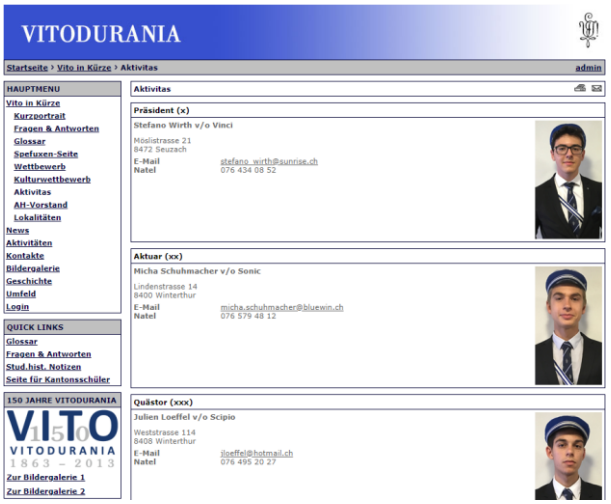
Bezeichner	UC2
Use Case Name	Logout
Akteure	Mitglied
Vorbedingung	Mitglied ist eingeloggt.
Ereignis Fluss	1. Das Mitglied klickt auf den Logout Button 2. Das Mitglied wird vom System abgemeldet 3. Das System löscht die vom Benutzer verwendete Session
Nachbedingung	Mitglied ist ausgeloggt. Die Session wurde vom System gelöscht.

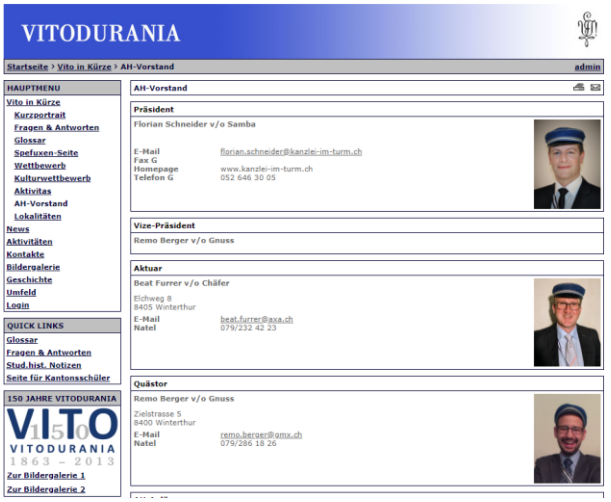
3.2.3 UC3: Passwort vergessen

Bezeichner	UC3
Use Case Name	Passwort vergessen
Akteure	Mitglied
Vorbedingung	Mitglied hat ein Passwort gesetzt (resp. dem System ist dessen Hash bekannt). Das System kennt vom Mitglied eine E-Mail-Adresse und sein Geburtsdatum.
Ereignis Fluss	1. Das Mitglied klickt im Login Dialog auf Passwort vergessen. 2. Das Mitglied wird nach seinem Cerevis und Geburtsdatum gefragt 3. Das System generiert ein E-Mail mit einem einmal gültigen Link zum Passwort Zurücksetzten Dialog. 4. Das E-Mail wird verschickt. 5. Das Mitglied klickt auf den im E-Mail enthaltenen Passwort Zurücksetzten Dialog Link 6. Das System prüft das der Link gültig ist und zum ersten Mal aufgerufen wird. 7. Das System zeigt den Passwort Zurücksetzten Dialog an. 8. Das Mitglied setzt sich ein neues Passwort. 9. Das System merkt sich einen kryptographischen Hash zum Passwort. 10. Das System zeigt den Login Dialog an.
Nachbedingung	Das Mitglied hat ein neues Passwort und kann sich gleich damit anmelden.

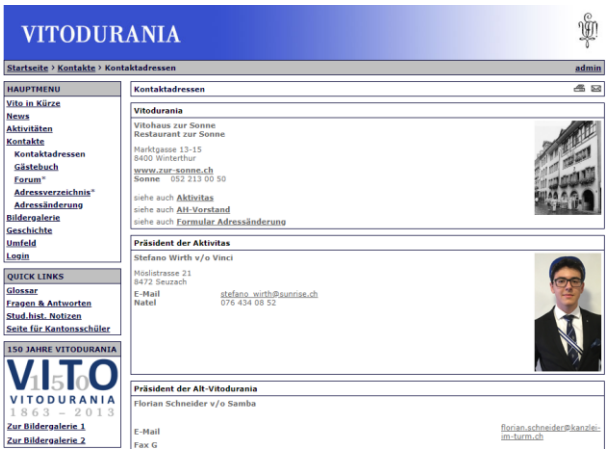
3.2.4 UC4: Adresslisten anzeigen

Bezeichner	UC4
Use Case Name	Adresslisten anzeigen
Akteure	Anonymer Benutzer
Vorbedingung	Adressen sind Schlagworten der Gruppe Aktivitas verschlagwortet.
Ereignis Fluss	1. Mitglied klickt auf die Seite Vito in Kürze -> Aktivitas 2. Seite zeigt die Kontakt Adressen der Mitglieder der Aktivitas. Adressen sind nach der Rolle im Verein sortiert.

	 **Abbildung 14: Aktivitas Seite https://www.vitodurania.ch/index/20/20/**
Nachbedingung	-

Bezeichner	UC4
Use Case Name	Adresslisten anzeigen (Alternative)
Akteure	Anonymer Benutzer
Vorbedingung	Adressen sind Schlagworten der Gruppe AH-Vorstand verschlagwortet.
Ereignis Fluss	1. Mitglied klickt auf die Seite Vito in Kürze -> AH-Vorstand 2. Seite zeigt die Kontakt Adressen der Mitglieder der Aktivitas. Adressen sind nach der Rolle im Verein sortiert.  **Abbildung 15: AH-Vorstand https://www.vitodurania.ch/index/20/22/**
Nachbedingung	-

Bezeichner	UC4
Use Case Name	Adresslisten anzeigen (Alternative)
Akteure	Anonymer Benutzer

Vorbedingung	Adressen sind Schlagworten der Gruppe Kontaktadressen verschlagwortet.
Ereignis Fluss	1. Mitglied klickt auf die Seite Vito in Kontakte -> Kontaktadressen 2. Seite zeigt die Kontakt Adressen der Vitodurania an. Die Liste enthält auch Einträge die nicht aus dem Adressverzeichnis kommen (Statischer Inhalt)  **Abbildung 16: Seite Kontakte https://www.vitodurania.ch/index/50/1/**
Nachbedingung	-

3.2.5 UC5: Adressänderung melden

Bezeichner	UC5
Use Case Name	Adressänderung melden
Akteure	Mitglied
Vorbedingung	Mitglied ist angemeldet
Ereignis Fluss	1. Mitglied klickt auf die Seite Vito in Kontakte -> Adressänderung 2. Das Mitglied kann die Felder für die Adressänderung ausfüllen 3. Pflichtfelder sind gekennzeichnet. 4. Das System prüft die Eingaben während der Eingabe und nach dem Absenden 5. Das Mitglied klickt auf absenden. 6. Das System sendet ein E-Mail an konfigurierte E-Mail-Adresse für Adressänderungen für die manuelle Prüfung der Eingabe.  **Abbildung 17: Seite Adressänderung https://www.vitodurania.ch/index/50/15/**
Nachbedingung	Email wurde zur manuellen Prüfung verschickt. **Option**: Daten werden bereits neu angezeigt, da sie von einer vertraulichen Person stammen.

3.2.6 UC6: An einem Anlass im Quartalsprogramm teilnehmen

Bezeichner	UC6
Use Case Name	An einem Anlass im Quartalsprogramm teilnehmen
Akteure	Mitglied
Vorbedingung	Mitglied ist nicht eingeloggt und befindet sich auf einem Anlass im Quartalsprogramm
Ereignis Fluss	1. Das Mitglied gibt seinen Vulgo und Geburtstag an und bestätigt mit dem Anmelden Knopf  **Abbildung 18: Anmeldung an einen Anlass im Quartalsprogramm**
Nachbedingung	Das Mitglied wird im Anlass als angemeldet geführt. Sein Name erscheint in der Teilnehmerliste.

Bezeichner	UC6
Use Case Name	An einem Anlass im Quartalsprogramm teilnehmen (Alternative)
Akteure	Mitglied
Vorbedingung	Mitglied ist eingeloggt und befindet sich auf einem Anlass im Quartalsprogramm
Ereignis Fluss	1. Das System erkennt das Mitglied und zeigt die Felder Vulgo und Geburtsdatum nicht an. 2. Bestätigt mit dem Anmelden Knopf seine Teilnahme am Anlass.
Nachbedingung	Das Mitglied wird im Anlass als angemeldet geführt. Sein Name erscheint in der Teilnehmerliste.

3.2.7 UC7: Quartalsprogramm anzeigen

Bezeichner	UC7
Use Case Name	Quartalsprogramm anzeigen
Akteure	Anonymer Benutzer
Vorbedingung	Anlässe für das aktuelle Semester wurden eingepflegt. Das Semester ist nicht zu Ende, oder es wurden bereits Anlässe für das folgende Semester eingepflegt.
Ereignis Fluss	1. Der anonyme Benutzer navigiert auf die Seite Aktivitäten -> Quartalsprogramm 2. Das System zeigt eine Liste von den nächsten Anlässen an (vergangene Anlässe werden nicht mehr angezeigt) Es gibt einen Hinweis auf den AH-Stamm Die Anlässe lassen sich nach den Kategorien: Alle Kategorien, Wandergruppe der Vitodurania, Alt-Vitodurania und Aktivitas filtern

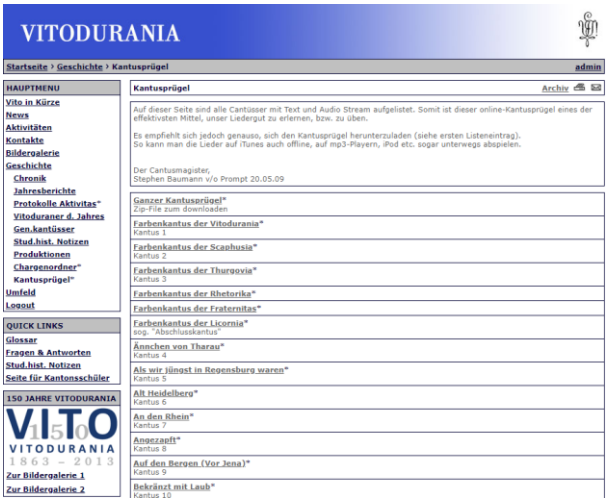
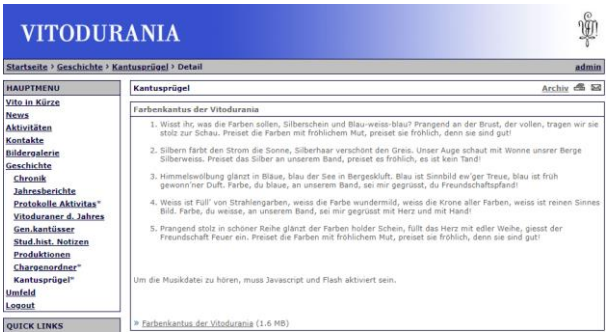
	 **Abbildung 19: Seite Quartalsprogramm https://www.vitodurania.ch/index/40/1/**
Nachbedingung	-

3.2.8 UC8: Quartalsprogramm als ICS anzeigen

Bezeichner	UC8
Use Case Name	Quartalsprogramm als ICS anzeigen
Akteure	Anonymer Benutzer
Vorbedingung	Anlässe für das aktuelle Semester wurden eingepflegt. Das Semester ist nicht zu Ende, oder es wurden bereits Anlässe für das folgende Semester eingepflegt.
Ereignis Fluss	1. Der anonyme Benutzer navigiert auf die Seite Aktivitäten -> Quartalsprogramm 2. Der anonyme Benutzer klickt auf den Link ICS-Datei. 3. Das System wandelt das Quartalsprogramm ins ICS Format um und bietet die Datei zum Download an. (Je nach System kommt entweder der Speichern unter… Dialog oder der Öffnen mit… Dialog) Die folgende Datei kann in Kalenderprogramme (iPhone, Google Kalender, Outlook) importiert werden: ICS-Datei **Abbildung 20: ICS-Datei Download Link**
Nachbedingung	-

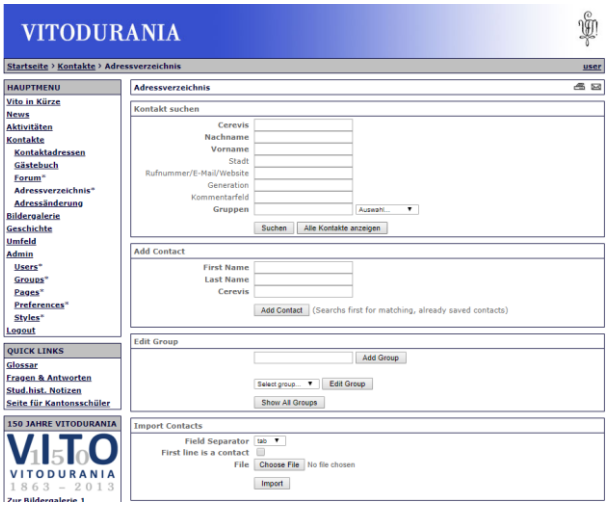
3.2.9 UC9: Kantusprügel anzeigen

Bezeichner	UC9
Use Case Name	Kantusprügel anzeigen
Akteure	Anonymer Benutzer
Vorbedingung	Die Kantusse wurden ins System eingepflegt.
Ereignis Fluss	1. Der anonyme Benutzer navigiert auf die Seite Geschichte -> Kantusprügel 2. Das System zeigt eine Liste aller erfassten Kantusse an sortiert nach der Nummerierung der gedruckten Version des Kantusprügels. 3. Der anonyme Benutzer kann einen Kantus auswählen um dessen Detail Seite anzuzeigen. 4. Das System zeigt den Lieder Text an und bietet (wenn möglich) das Lied als MP3 als Audio Stream und Download an.

	Technische Notiz: moderne Browser haben das Flash Plugin nicht mehr installiert. Es soll ein moderner Audio Player verwendet werden, welcher ohne Flash Technologie auskommt.  **Abbildung 21: Seite Kantusprügel https://www.vitodurania.ch/index/60/80/**  **Abbildung 22: Seite Kantusprügel Detail** **https://www.vitodurania.ch/index/60/80/0/0/0/0/showSingle/5/**
Nachbedingung	-

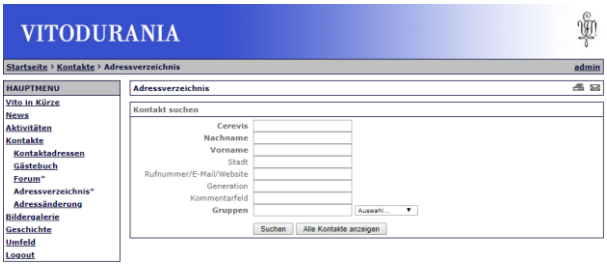
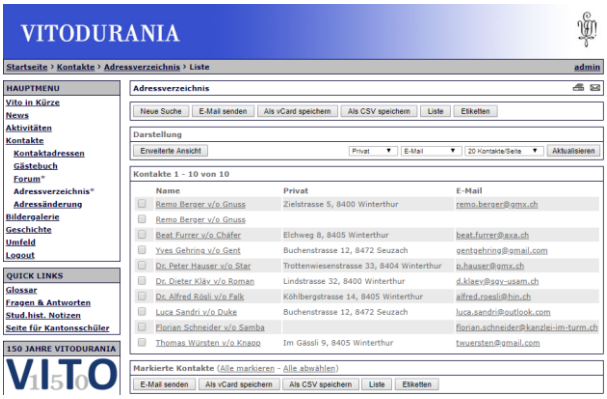
3.2.10 UC10: Benutzer verwalten

Bezeichner	UC10
Use Case Name	Benutzer verwalten
Akteure	Mitglied (Rolle Administrator)
Vorbedingung	Das Mitglied ist eingeloggt und wurde mit der Rolle Administrator autorisiert.
Ereignis Fluss	1. Das Mitglied navigiert auf die Seite Kontakte -> Adressverzeichnis 2. Das System erkennt die Rolle Administrator und zeigt unterhalb des Suchfeldes Eingabefelder für die Verwaltung von Adressen an. 3. Nach einer Suche werden im Resultat auch Köpfe für die Verwaltung von Adressdaten angezeigt.

	 **Abbildung 23: Adressverzeichnis Suche Administrator https://www.vitodurania.ch/index/50/10/**  **Abbildung 24: Adressverzeichnis Resultat Administrator**
Nachbedingung	Benutzerdaten wurden geändert und werden im System verfügbar. Das System protokoliert den Benutzer, die Zeit und die Änderung im Veränderungslog.

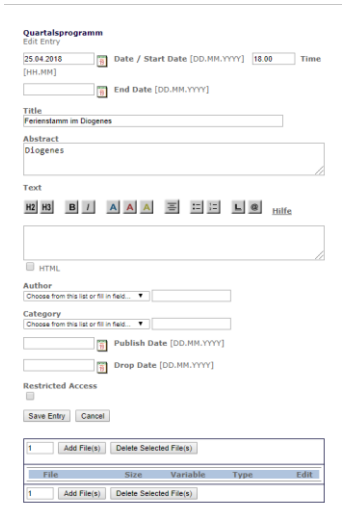
3.2.11 UC11: Adressen suchen

Bezeichner	UC11
Use Case Name	Adressen suchen
Akteure	Mitglied
Vorbedingung	Adressdaten der Mitglieder sind im System eingepflegt. Das Mitglied ist eingeloggt.
Ereignis Fluss	1. Das Mitglied navigiert auf die Seite Kontakte -> Adressverzeichnis 2. Das System zeigt eine Suchmaske an. 3. Das Mitglied füllt beliebige Felder aus und bestätigt die Suche mit dem Suche Knopf. 4. Das System sucht nach treffenden Adressen und zeigt diese an.

	 **Abbildung 25: Adressverzeichnis Suche https://www.vitodurania.ch/index/50/10/**  **Abbildung 26: Adressverzeichnis Resultat**
Nachbedingung	-

3.2.12 UC12: Quartalsprogramm bearbeiten

Bezeichner	UC12
Use Case Name	Quartalsprogramm bearbeiten
Akteure	Mitglied (Rolle Administrator)
Vorbedingung	Das Mitglied ist eingeloggt und wurde mit der Rolle Administrator autorisiert.
Ereignis Fluss	1. Das Mitglied navigiert auf die Seite Aktivitäten -> Quartalsprogramm 2. Das System erkennt die Rolle des Administratoren und zeigt einen Admin Knopf an. 3. Das Mitglied klickt auf den Admin Knopf und wechselt in den Bearbeitungsmodus. 4. Das Mitglied erstellt einen neuen Anlass mittels Add Entry Knopf. 5. Das Mitglied füllt das Formular aus. 6. Das System prüft während der Eingabe und nach dem absenden des Formulars die Eingaben 7. Das System speichert den Anlass nach erfolgreicher Prüfung. **Technischer Hinweis**: Die alte VitoSite hat hier den Ansatz von Formularen in einem neu geöffneten Fenster gewählt. Schöner und benutzerfreundlicher ist der Ansatz die Daten inline zu bearbeiten, so dass bereits während der Eingabe dem Mitglied angezeigt wird, wie der Anlass aussehen wird.

	 **Abbildung 27: Quartalsprogramm Bearbeitungsmodus**  **Abbildung 28: Quartalsprogramm Bearbeitungsmodus Formular**
Nachbedingung	Der Anlass wurde neu angelegt und steht im System zur Verfügung. Das System protokoliert den Benutzer, die Zeit und die Änderung im Veränderungslog.

3.2.13 UC13: Kantusprügel bearbeiten

Bezeichner	UC13
Use Case Name	Kantusprügel bearbeiten
Akteure	Mitglied (Rolle Administrator)
Vorbedingung	Das Mitglied ist eingeloggt und wurde mit der Rolle Administrator autorisiert.
Ereignis Fluss	1. Das Mitglied navigiert auf die Seite Geschichte -> Kantusprügel 2. Das System erkennt die Rolle des Administratoren und zeigt einen Admin Knopf an. 3. Das Mitglied klickt auf den Admin Knopf und wechselt in den Bearbeitungsmodus. 4. Das Mitglied erstellt einen neuen Kantus mittels Add Entry Knopf. 5. Das Mitglied füllt das Formular aus. 6. Das System prüft während der Eingabe und nach dem absenden des Formulars die Eingaben 7. Das System speichert den Kantus nach erfolgreicher Prüfung.

Abbildung 29: Kantusprügel Bearbeitungsmodus

Abbildung 30: Kantusprügel Bearbeitungsmodus Formular

Nachbedingung	Der Kantus wurde neu angelegt und steht im System zur Verfügung. Das System protokoliert den Benutzer, die Zeit und die Änderung im Veränderungslog.

3.3 Qualitätsmerkmale - nicht-funktionale Anforderungen.

Qualitätsmerkmale welche nicht in den Use Cases abgebildet sind nach ISO 9126 [14].

3.3.1 NFR1: API Efficiency

Die Schnittstelle soll einer Last von 10 lesenden Anfragen von 400 Mitgliedern innerhalb einer Minute aushalten und auch nicht bei 1 stündiger Dauerlast stabil laufen.
(Dies entspricht der doppelten Anzahl der maximalen Teilnehmer an der Generalversammlung)

3.3.2 NFR2: Website Efficiency

Die Website soll einer Last von 10 lesenden Anfragen von 200 Mitgliedern innerhalb einer Minute aushalten und auch nicht bei 1 Stündiger Dauerlast stabil laufen.
(Dies entspricht der Anzahl der maximalen Teilnehmer an der Generalversammlung)

3.3.3 NFR3: Wartbarkeit

Es entsteht nebst der Dokumentation auch eine Betriebsdokumentation in welcher Festgehalten ist was und wie auf dem Server deployet wurde.
Die Schnittstellen sind Dokumentiert, so dass Erweiterungen auch von anderen Personen – welche mit der Technologie vertraut ist – durchgeführt werden können.

3.3.4 NFR4: Erweiterbarkeit

Die eingesetzten Programmiersprachen, Frameworks und Server Komponenten sind frei verfügbar und deren Dokumentation ist einsehbar. Sie haben eine breite Community welche diese Produkte aktiv einsetzt.

3.3.5 NFR5: Verständlichkeit

Die Website und Mobile App lehnt sich an das Design des Vorgängers an oder verwendet im Aussehen ein in der Öffentlichkeit des WWW bekanntem CSS Framework.
Die Mitglieder finden sich somit in der neuen Version leicht zurecht.

3.4 Domain Model

Das Domain Model wurde dem System vorgegeben durch den Import der Daten aus der alten VitoSite extrahierte.

3.4.1 User Model

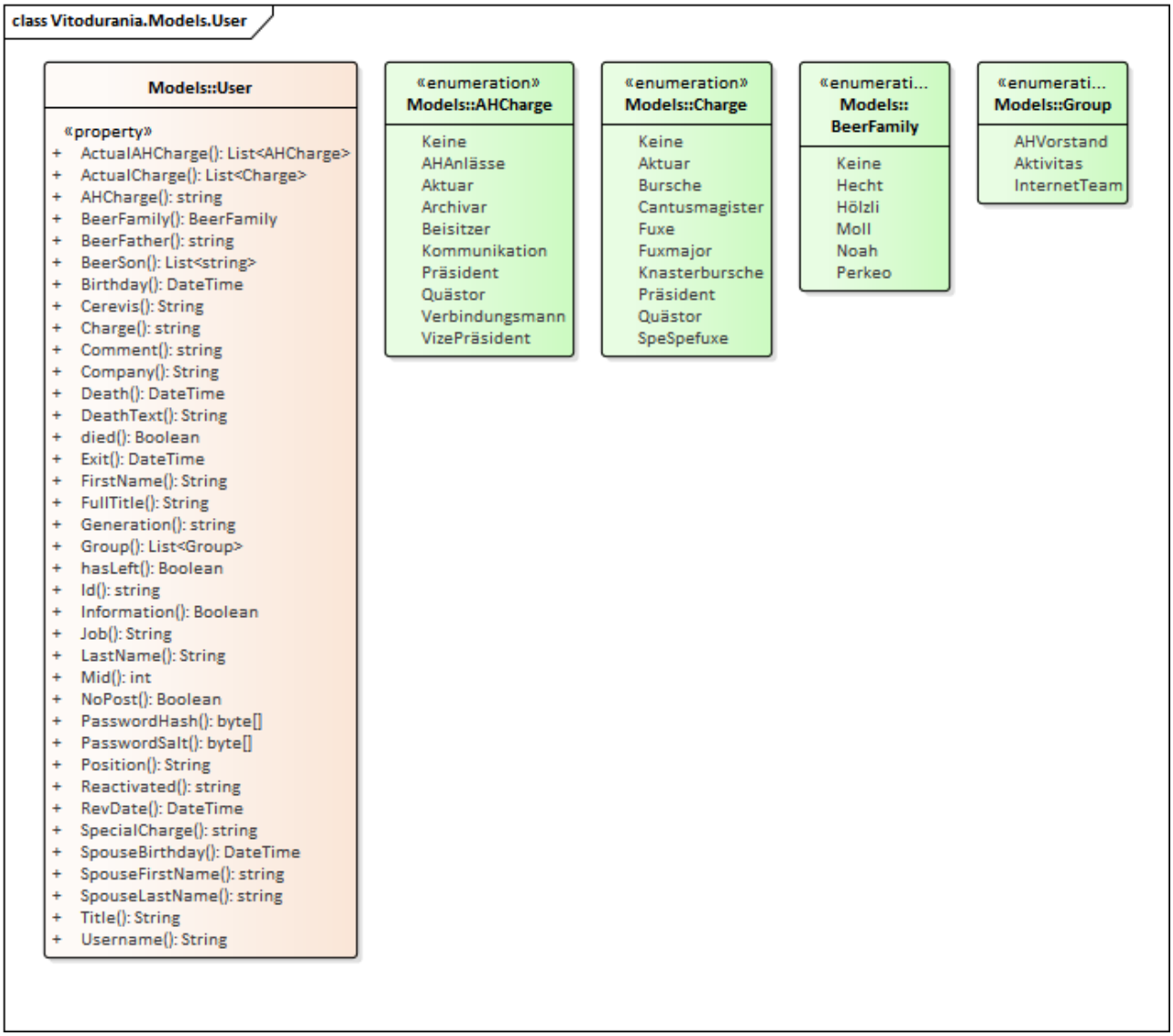


Abbildung 31: Models\User

Dieses Diagramm zeigt das importierte `User` Entity, welches ein Mitglied repräsentiert, mit seinen Abhängigkeiten. Es resultieren 4 Enumerationen:

- `AHCharge`: Zeigt die Funktion / Rolle im AH Vorstand.
- `Charge`: Zeigt die Funktion / Rolle in der Aktivitas
- `BeerFamily`: Zeigt die Zugehörigkeit zur Bierfamilie
- `Group`: Zeigt an welcher optionalen Kontaktegruppe das Mitglied angehört. Ist ein Mitglied nicht einer Gruppe angehörig, hat es die Rolle Mitglied.

3.4.2 Event Model

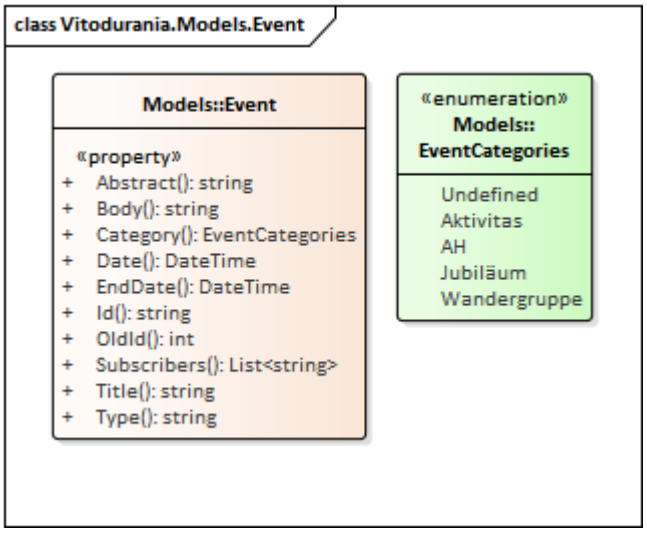


Abbildung 32: Model\Event

Dieses Diagramm zeigt das importierte `Event` Entity, welches einen Anlass im Quartalsprogramm repräsentiert. Es hat nur eine Enumeration `EventCategories`, welches den Typ des Anlasses repräsentiert. Im Feld `Subscribers` wird eine Liste mit Vulgos geführt von Mitgliedern welche sich an den Anlass angemeldet haben.

3.4.3 Song Model

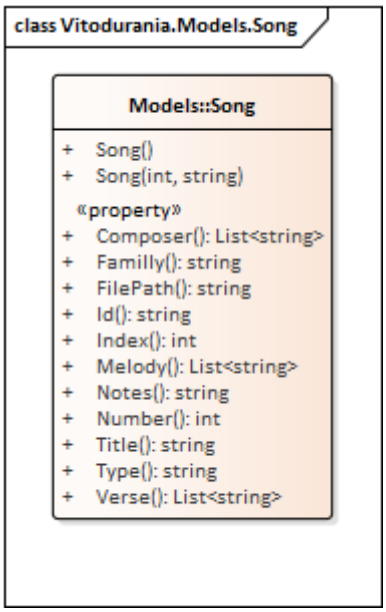


Abbildung 33: Model\Song

Dieses Diagramm zeigt das importierte `Song` Entity, welches ein Lied im Kantusprügel repräsentiert. Die Lieder sind sortiert nach `Index`.

3.4.4 MenuNavigationItem Model

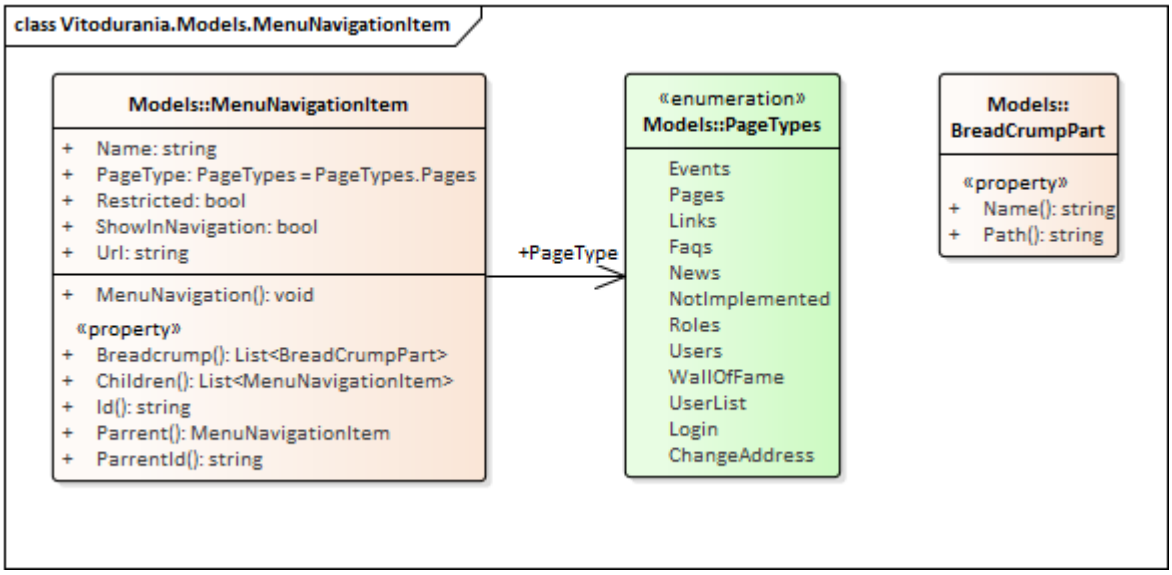


Abbildung 34: Model\MenuNavigationItem

Dieses Model ermöglicht die Darstellung einer hierarchischen Seiten Navigation.

Das Property `Breadcrump` wird verwendet um die gleich benannte Komponente zu rendern. Das Property `Children` enthält die Kindseiten welche über `ParrentId` mit dem Elter verbunden sind.

Die Enumeration `PageType` enthält die möglichen Seiten Typen. Dieser entscheidet welche Komponente den Inhalt darstellt.

3.4.5 Page

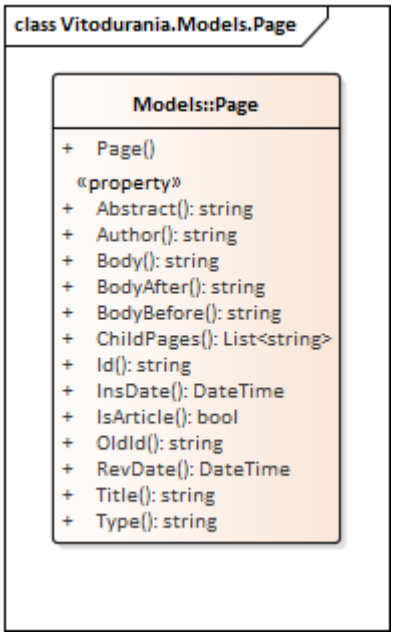


Abbildung 35: Model\Page

Eine Inhaltsseite. Das Property `ChildPages` ist eine Liste von Kindseiten Ids.

3.5 Schnittstellen

Zu den neu benötigten Schnittstellen wurden keine genauen Anforderungen formuliert. Die Schnittstelle soll aber folgende Bedingungen erfüllen.

Das API soll mittels Swagger dokumentiert werden. Somit wird sichergestellt, dass die Dokumentation mit dem Quelltext verbunden ist und bei Anpassungen ebenfalls mit verändert wird.

Das API soll eine REST Schnittstelle sein, welche JSON Dokumente als Antwort liefert.

4. Architektur

4.1 Ziele

Die Architektur soll einfach gegliedert sein. Einzelne Komponenten sollen austauschbar sein. Zur Kommunikation zwischen Komponenten sollen standardisierte Protokolle (z.B. HTTP) eingesetzt werden.

Die Architektur soll ausreichend Dokumentiert sein, damit auch andere Vereinsmitglieder Erweiterungen dieser schreiben können. Die Dokumentation soll die Architektur aufzeigen um einem neuen Entwickler einen schnellen Einstieg in die Weiterentwicklung zu ermöglichen.

Für dieses Projekt wird nur ein Server verwendet. Die Architektur soll jedoch eine horizontale Skalierung aller Komponenten (ausser der mobilen Applikation) sicherstellen.

Die Performance soll der doppelten Anzahl (400) aller Vereinsmitglieder für 10 gleichzeitig lesende Seitenaufrufen innerhalb einer Minute über einen Zeitraum von einer Stunde standhalten.

4.2 Einschränkungen

Die Software soll auf einem Linux Server betrieben werden können, da das Hosting nur diese erlaubt. Linux Server haben sich bei der Legacy VitoSite als sehr robust im Betrieb gezeigt. Ausserdem ist bereits einiges an Know-how vorhanden.

Wann immer möglich sollen Technologien verwendet werden, welche eine breite Unterstützung haben in der Community und auch Teil des Lehrstoffes an den Hochschulen ist. Somit soll sichergestellt sein, dass nicht auf veraltete Technogien gesetzt wird.

Da die VitoSite personenbezogene Daten speichert, muss sichergestellt werden, dass nur autorisierte Anfragen im API für schützenswerte Daten erlaubt werden.

4.3 Logische Architektur

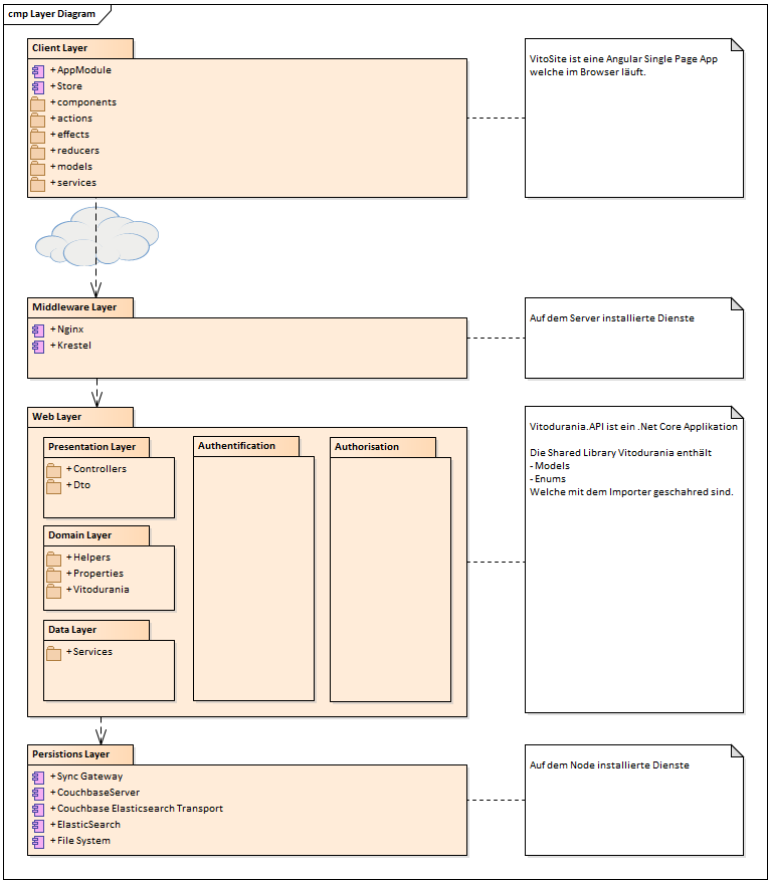


Abbildung 36: Layer Diagramm

Dieses Layer Diagramm zeigt die logische Architektur des Projektes. Ein Teil der Software läuft im Browser des Benutzers ab. Der Middleware Layer sind auf dem Server installierte Dienste, welche aber im Datenfluss durchlaufen werden. Es ist auch die Schnittstelle und Einstiegspunkt zur auf dem Server installierten Software. Zur Kommunikation wird das HTTP Protokoll eingesetzt.

Der Web Layer entspricht dem API und zeigt dessen logische Aufteilung.

Beim Persistenz Layer handelt es ich wieder um auf dem Server installierte Dienste.

4.3.1 Client Layer

Der Inhalt der VitoSite wurde in Angular Komponenten zerstückelt. Jede Komponente kümmert sich um das Rendering eines kleinen Teils der Webseite. Dank dem Angular Framework ist das DOM der Seite in Zonen aufgeteilt und es werden nur Zonen gerendert oder geupdatet, wenn dies nötig wird. Es können somit einzelne Komponenten nachgeladen werden ohne die anderen Komponenten zu beeinflussen.

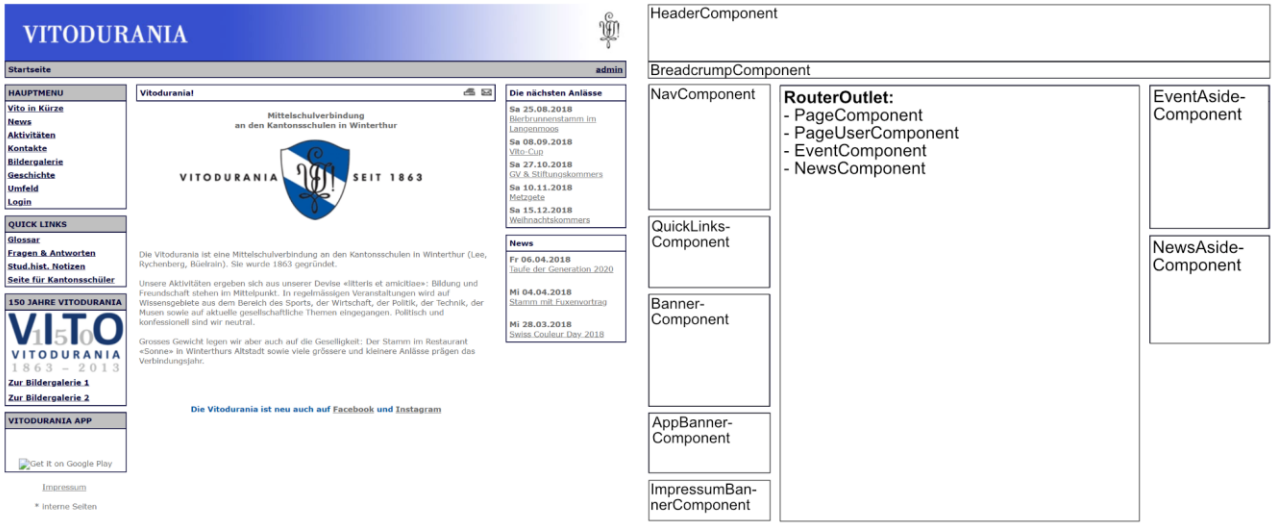


Abbildung 37: Angular Komponenten

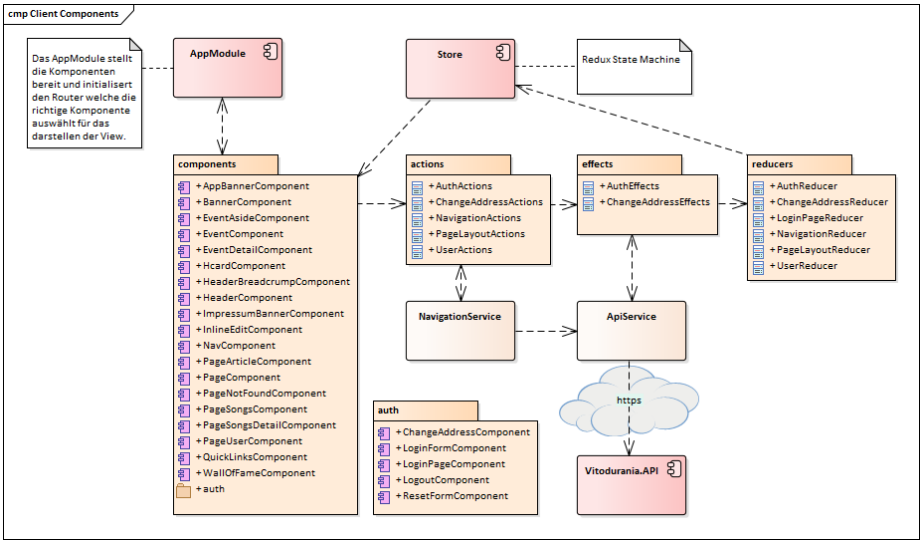


Abbildung 38: Client Components

Komponenten beziehen Ihre Daten über einen globalen RxJS State [15] welcher Store genannt wird. Er basiert auf Ideen von Redux [16].

Store: Der Store kann mit einer Datenbank auf dem Client verglichen werden, welche die «single source of truth» [17] darstellt. Zu jedem Zeitpunkt enthält sie einen Zustand des globalen Applikationszustandes [englisch: *state*]. Der Applikationszustand ist zentral und unwandelbar. Da Komponenten nur Scheiben des Applikationszustandes konsumieren, kann Angular die Komponenten nur bei Änderungen dieser anweisen zu reagieren, was eine wesentliche Performance Optimierung darstellt.

Aktionen: Der Applikationszustand kann nur mittels versenden [englisch: *dispatching*] von Aktionen [englisch: *actions*] geändert werden. Eine Aktion ist eine Nachricht, welche aus einem Aktionstypen [englisch: *action types*] besteht. Ein Aktionstyp besteht aus einem einfachen String der eindeutig sein muss. Optional kann einer Aktion noch eine Nutzlast [englisch: *payload*] mitgegeben werden.

Reduzierer: Während der Anwendungsspeicher den Zustand beibehält, sind Reduzierer [englisch: *reducers*] das Arbeitspferd für die Manipulation und Ausgabe neuer Zustandsdarstellungen, wenn Aktionen ausgelöst werden. Jeder Reduzierer sollte auf einen bestimmten Abschnitt oder ein bestimmtes Statussegment ausgerichtet sein, ähnlich einer Tabelle in einer Datenbank. Ein Reduzierer ist eine reine Funktion [englisch: *pure function*] ohne Seiteneffekte, welche 2 Argumente annimmt: Das erste Argument entspricht dem vorgängigen Zustand und das zweite Argument ist eine Aktion.

Seiteneffekte: Seiteneffekte können auch entstehen, wenn an der Datenbank eine Änderung vorgenommen wird. Effekte [englisch: *effects*] Isolieren Komponenten von Seiteneffekten umso reinere Komponenten zu ermöglichen. Effekte hören [englisch: *listen*] auf Aktionen welche vom Store versendet wurden. Effekte sollen immer bei externen Interaktionen (Netzwerkanfragen, Websocketanfragen, etc.) verwendet werden, da der Zustand im externen System geändert werden könnte.

API Service: Diese Klasse kapselt die Anfragen an das API an einer zentralen Stelle. Vereinfacht kann festgestellt werden, dass das HTTP Verb GET und OPTOINS keine Seiteneffekte auslösen. POST, PUT und PATCH jedoch schon.

Navigation Service: Diese Klasse bündelt Funktionalität um die Navigation zu konsumieren.

4.3.2 API Layer

Der API Layer ist eine .Net core MVC Applikation. Folgende Middlewares sind konfiguriert:

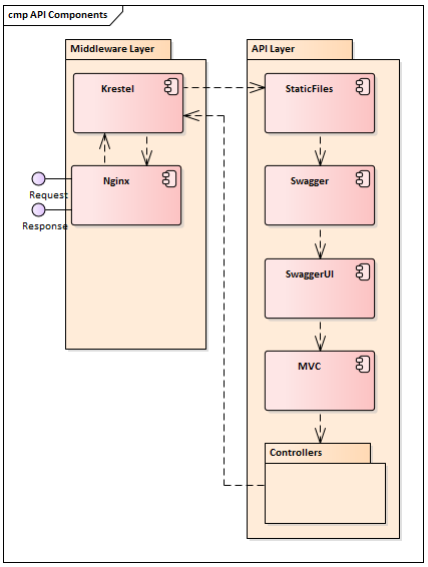


Abbildung 39: HTTP request pipeline / Middleware config

Die Konfiguration zu Nginx [7] und Krestel [18] befinden sich in der Betriebsdokumentation.

Die MVC Komponente ist zusätzlich mit einem Output Formatter `CalendarOutputFormatter` konfiguriert, welcher iCal [mime-type: *text/calendar*] unterstützt, da diese Erweiterung sonst dem Framework nicht bekannt wäre.

Da Angular CORS [19] voraussetzt, hat die MVC Komponente noch in der Klasse `Startup` einen Cors Authorization Filter konfiguriert.

Die Swagger Konfiguration ist ebenfalls in der Klasse `Startup` konfiguriert.

4.4 Schnittstellen

Das API dokumentiert die Schnittstellen mit Hilfe von «swagger ui» [20].

Die aktuellen Endpunkte sehen wie folgt aus:

4.4.1 User

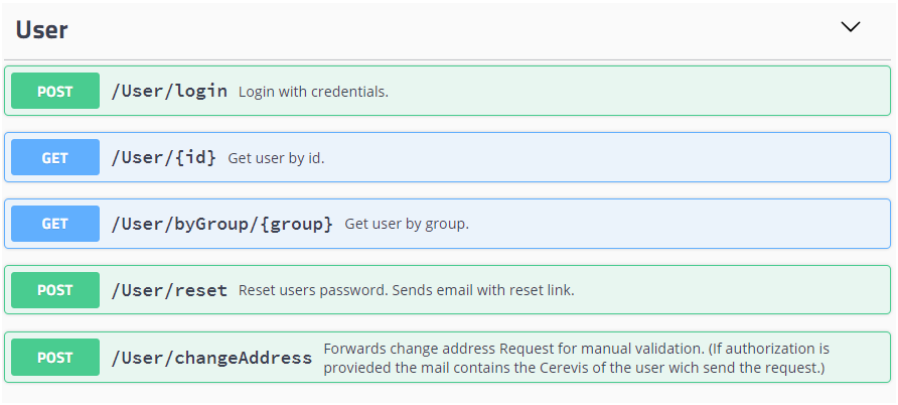


Abbildung 40: API User

4.4.1.1 POST /User/login

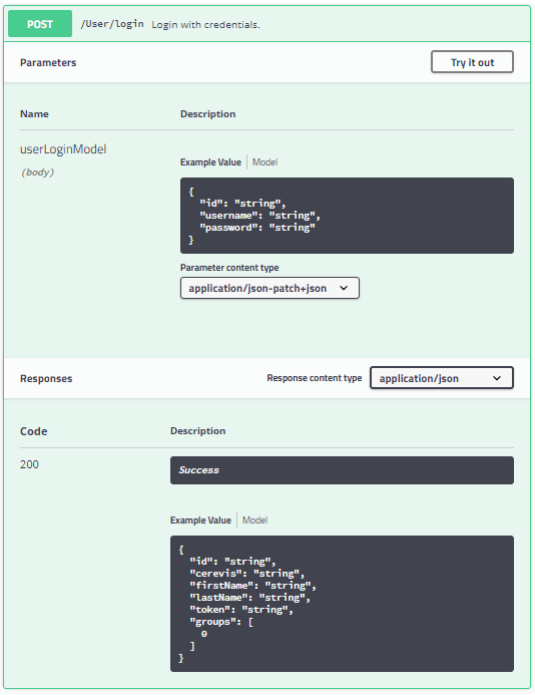


Abbildung 41: API User - POST /User/login

Wird verwendet um sich einzuloggen.

4.4.1.2 GET /User/{id}

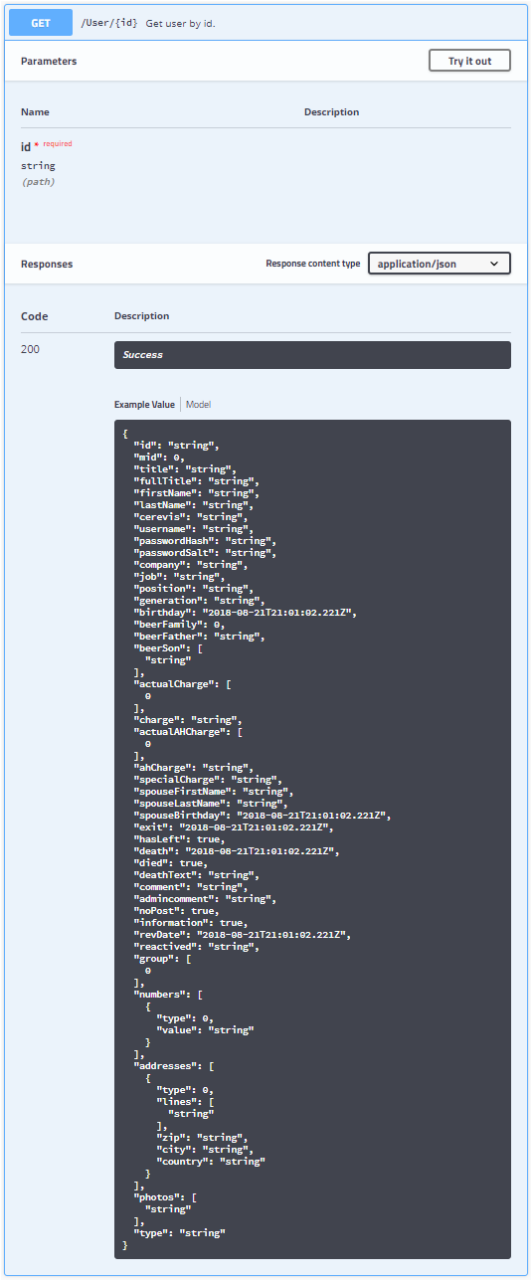


Abbildung 42: API User - GET /User/{id}

Wird verwendet um Detail Informationen eines Benutzers abzufragen. Wird noch nicht verwendet. Dient als Vorbereitung von *UC10 Benutzer verwalten* welcher noch nicht umgesetzt wurde.

4.4.1.3 GET /User/byGroup/{group}

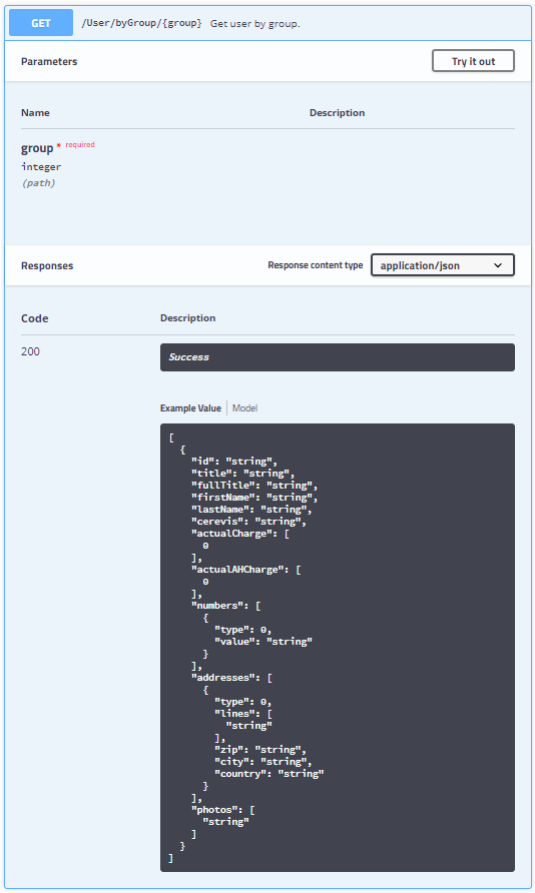


Abbildung 43: API User - GET /User/byGroup/{group}

Wird im *UC4 Adresslisten anzeigen* verwendet um Adresslisten der verschiedenen Gruppen (Aktive, AH Vorstand, etc.) anzuzeigen.

4.4.1.4 POST /User/reset

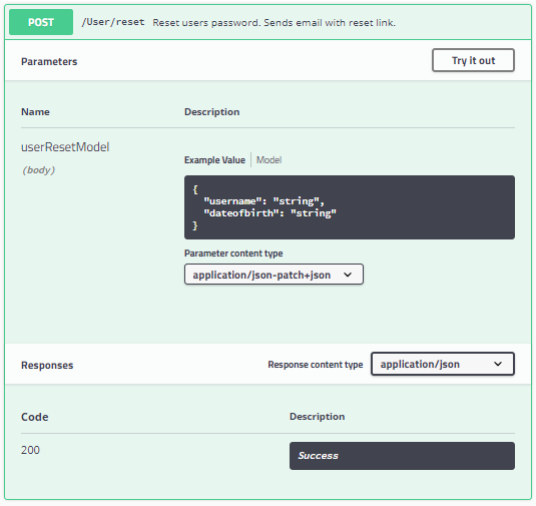


Abbildung 44: API User - POST /User/reset

Wird beim *UC3 Passwort vergessen* verwendet. Überprüft wird, ob ein Benutzer mit dem Geburtsdatum gefunden wird. Anschliessend wird dem User hinterlegten E-Mail-Adresse ein Link geschickt, mit welchem sich der Benutzer einloggen kann um sein Passwort zurückzusetzten.

4.4.1.5 POST /User/changeAddress

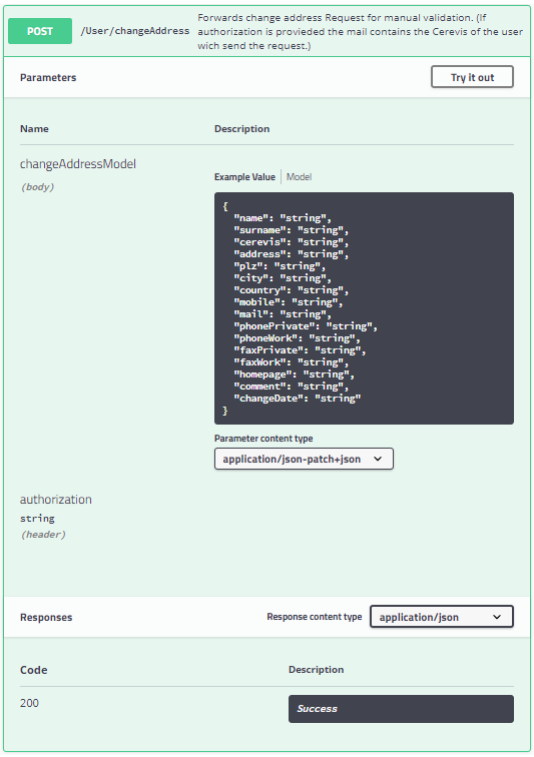


Abbildung 45: API User - POST /User/changeAddress

Wird im *UC5 Adressänderung melden* verwendet, damit ein anonymer oder eingeloggter Benutzer seine Adresse ändern kann. Es wird ein E-Mail an die für Adressänderungen zuständige Person verschickt zwecks Prüfung. War der Benutzer eingeloggt wird dies im E-Mail erwähnt. Ansonsten muss die Person prüfen, ob Adressänderung plausibel ist. (Kombination von Cerevis und Vor- und Nachname sind nicht öffentlich bekannt).

4.4.2 Event

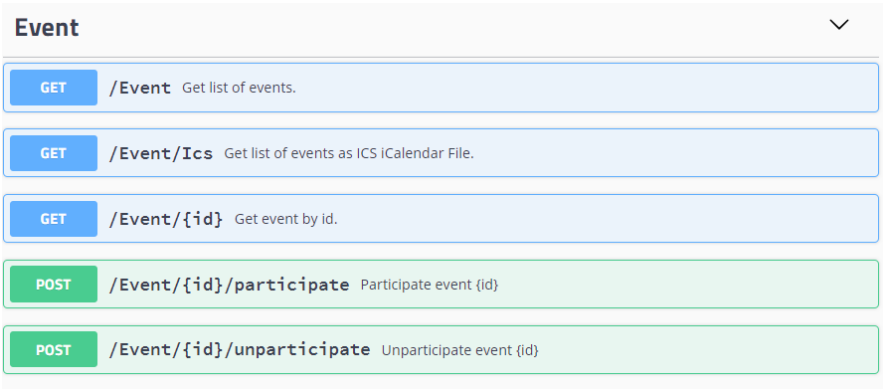


Abbildung 46: API Event

4.4.2.1 GET /Event

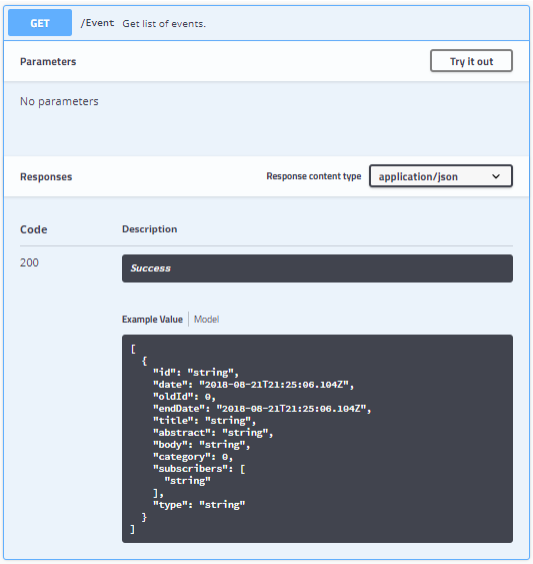


Abbildung 47: API Event - GET /Event

Wird in *UC7 Quartalsprogramm anzeigen* verwendet um eine Liste der Events zu erhalten.

4.4.2.2 GET /Event/Ics

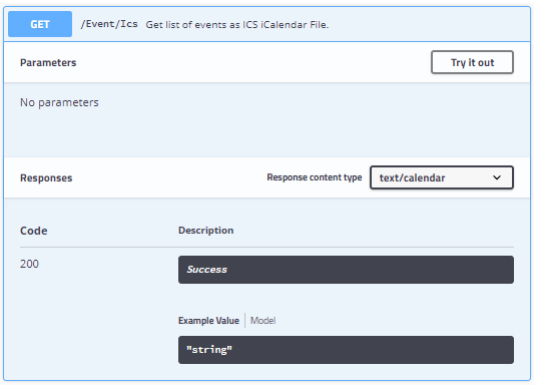


Abbildung 48: API Event - GET /Event/Ics

Wird in *UC8 Quartalsprogramm als ICS anzeigen* verwendet um die Anlässe als iCal ICS Datei herunterzuladen.

4.4.2.3 GET /Event/{id}

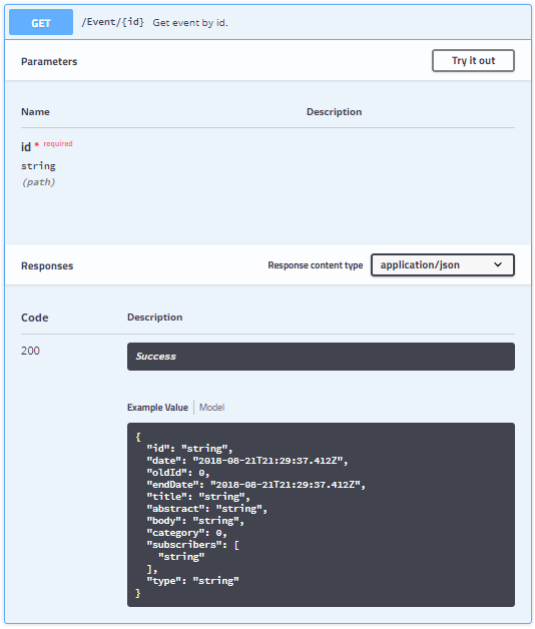


Abbildung 49: API Event - GET /Event/{id}

Wird in *UC7 Quartalsprogramm anzeigen* verwendet um einen Events in der Detailansicht darzustellen.

4.4.2.4 POST /Event/{id}/participate

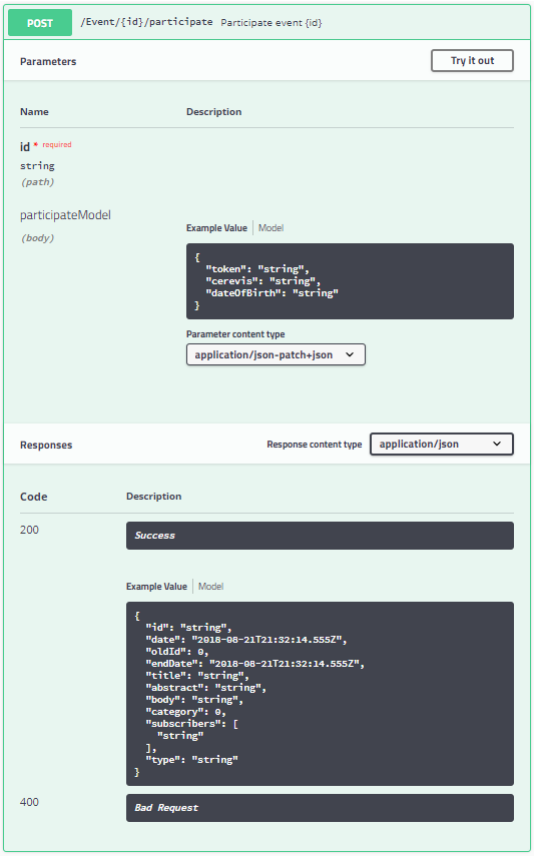


Abbildung 50: API Event - POST /Event/{id}/participate

Wird in *UC6 An einem Anlass im Quartalsprogramm* teilnehmen verwendet um einem Anlass teilzunehmen. Anonym muss das Cerevis mit gültigem Geburtsdatum übermittelt werden, damit eine Überprüfung der Person vorgenommen werden kann. Ansonsten wird nur das Token übermittelt.

4.4.2.5 POST /Event/{id}/unparticipate

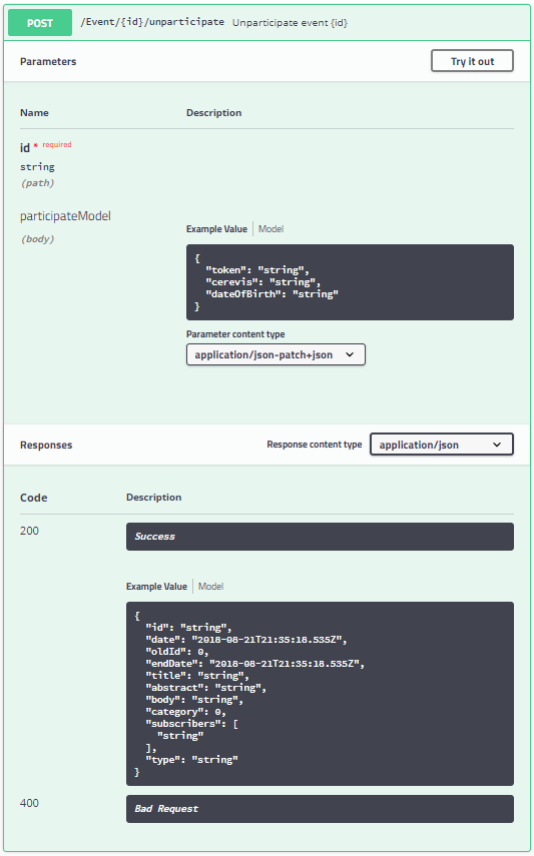


Abbildung 51: API Event - POST /Event/{id}/unparticipate

Wird in *UC6 An einem Anlass im Quartalsprogramm* teilnehmen verwendet um sich einem Anlass abzumelden. Anonym muss das Cerevis mit gültigem Geburtsdatum übermittelt werden, damit eine Überprüfung der Person vorgenommen werden kann. Ansonsten wird nur das Token übermittelt.

4.4.3 Song

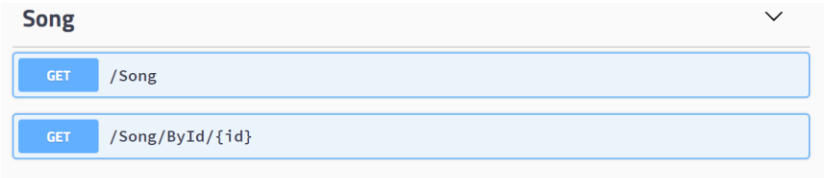


Abbildung 52: API Song

4.4.3.1 GET /Song

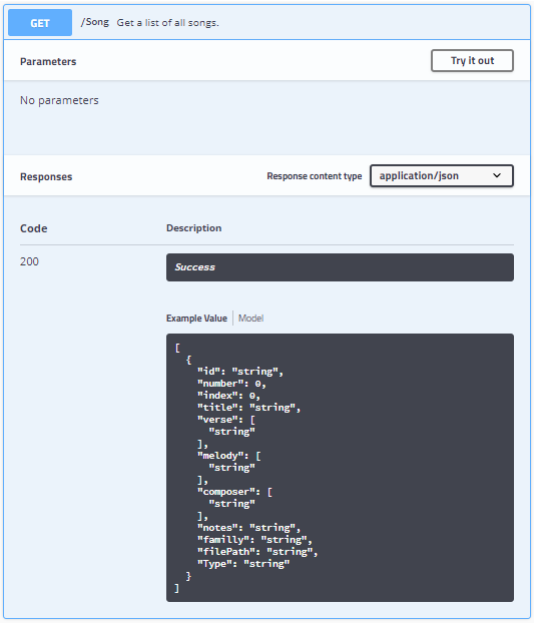


Abbildung 53: API Song - GET /Song

Wird in *UC9 Kantusprügel anzeigen* verwendet um die Inhaltsseite aller Lieder darzustellen.

4.4.3.2 GET /Song/{id}

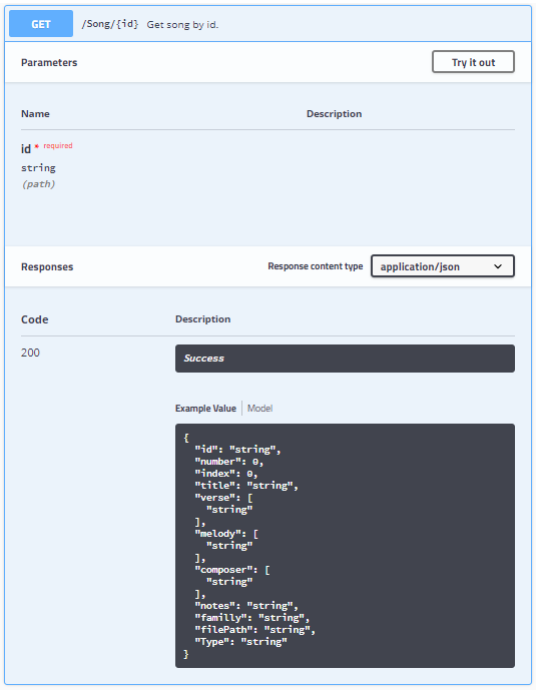


Abbildung 54: API Song - GET /Song/{id}

Wird in *UC9 Kantusprügel anzeigen* verwendet um die Detailseite eines Liedes darzustellen.

4.4.4 Navigation

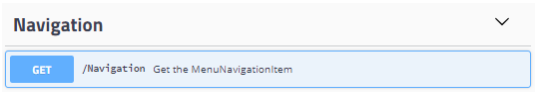


Abbildung 55: API Navigation

4.4.4.1 GET /Navigation

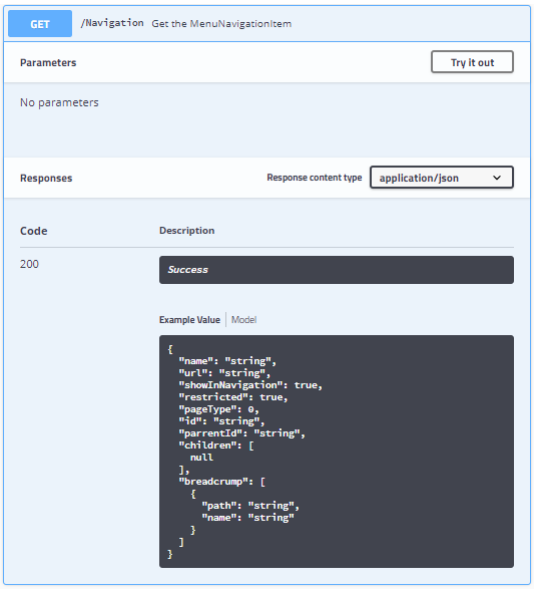


Abbildung 56: API Navigation - GET /Navigation

Nicht Bestandteil der Masterarbeit, aber nötig um die verschiedenen Seiten hierarchisch navigierbar zu machen. Es wird in der Komponente Breadcrump und Navigation verwendet.

4.4.5 Page

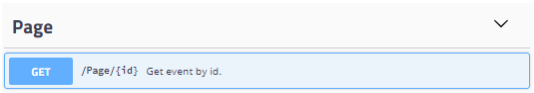


Abbildung 57: API Page

4.4.5.1 GET /Page/{id}

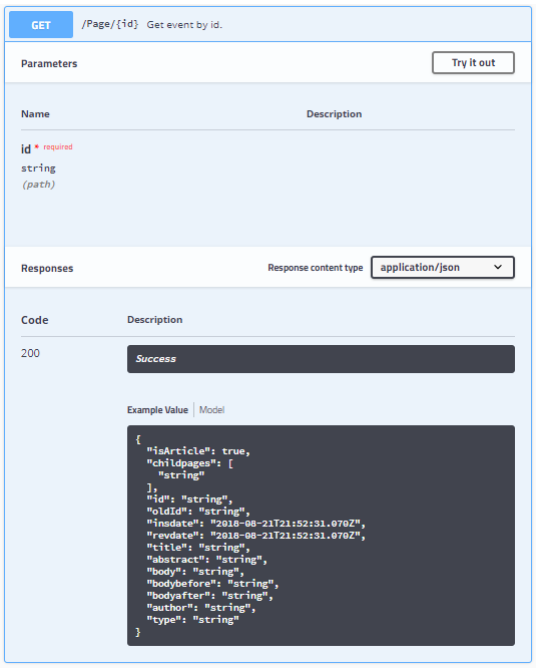


Abbildung 58: API Page - GET /Page/{id}

Nicht Bestandteil der Masterarbeit. Wird verwendet um Inhaltsseiten darzustellen. Wurde benötigt, als verschiedene Texteditoren ausprobiert wurden. Es können zwar Seiten bearbeitet werden, es fehlen jedoch noch die Routen diese zu speichern.

4.5 Wichtige Abläufe

4.5.1 Organisation des Quelltextes

Der Quelltext aller beteiligten Komponenten war zu Anfang des Projektes in einem Monorepo. Das heiss, alles war in einem einzigen Git Repository. Da jedoch nicht leicht festgestellt werden konnte, welche Komponente sich bei einer Quelltextänderung ändert, wurde mit Hilfe von Git Submodulen das Monorepo erweitert und stell jeder Komponente ein eigenes Git Repository zur Verfügung.

Um diesen Umstand zu dokumentieren wurde im Hauptrepository eine ReadMe.md Datei hinzugefügt, die den Aufbau des Git Repository erklärt:

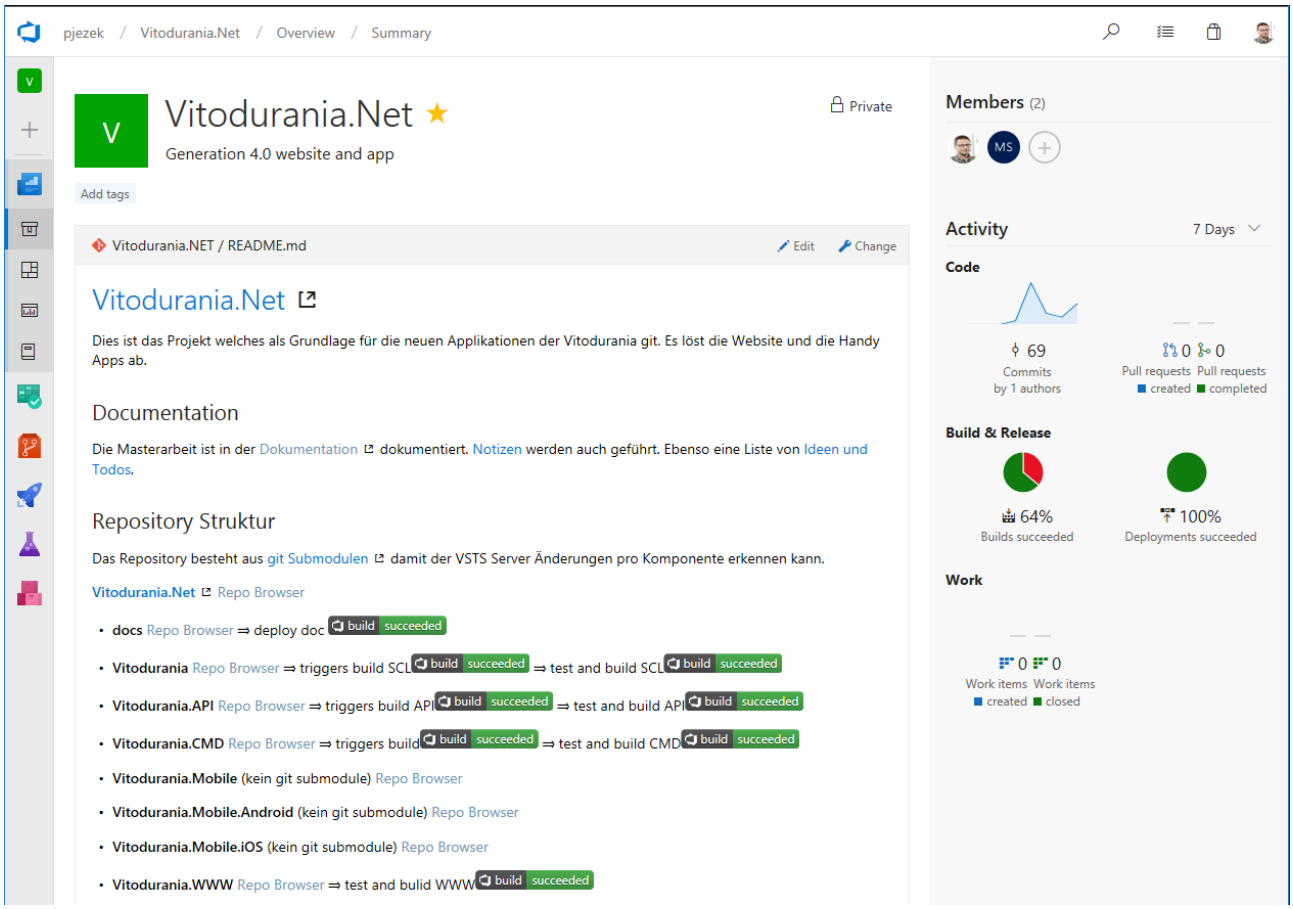


Abbildung 59: VSTS Übersichtsseite Vitodurania.Net

4.5.2 VitoSite Generell

Die Applikation hat keine komplexen Vorgänge. Im Prinzip ist es immer das Anfrage / Antwort Pattern [21], wie es für Webseiten typisch ist. Weil es sich bei der Applikation um eine SPA (Single Page App) handelt wird meistens Anstelle von Anfrage / Antwort das Ereignis / Zustandsänderung Pattern [22] verwendet. Die Seite wird nicht mehr verlassen und neue Inhalte werden per XHR Request nachgeladen. Jede Benutzer Interaktion löst somit eine Zustandsänderung aus.

Die erwähnten Klassen, Module und Komponenten sind in *Abbildung 4: Client Components* auf Seite 9 dargestellt.

4.5.3 Zustandsmaschine

Die im Kapitel 4.1.1 vorgestellte NGRX Komponente stellt eine zentrale Zustandsmaschine bereit und jede Angular Komponente reagiert auf Änderungen innerhalb des Zustandes. Da der Zustand zentral gehalten wird und der Zugriff über klar definierte Schnittstellen erfolgt, hilft die Zustandsmaschine die Angular Komponenten unter sich zu entkoppeln.

4.5.4 User Session

Wird die Seite neu aufgerufen, z.B. mit dem Anklicken eines Links in einem E-Mail versucht die Applikation den zuvor eingeloggten User zu ermitteln. Dies passiert in der `AppComponent`. Damit das funktioniert wird ein Objekt vom Typ `LoginResponse` im lokalen Speichers des Browsers gelesen. Ist das Objekt vorhanden wird ein `LoginSuccess` ausgelöst, das genau selbe Ereignis, welches auch nach einem erfolgreichen Login ausgelöst wird. Die Logik wird wiederverwendet.

Möglich ist das, da die Session ein JWT Token ist. Dieser ist kryptografisch signiert und kann nicht verändert werden, ohne dass eine Überprüfung dies nicht feststellen kann. Das Token hat ein Ablaufdatum. Ist das Token nicht mehr gültig, wird der Benutzer nicht eingeloggt und benützt die Website als anonymer Benutzer. Die Session ist dann abgelaufen.

4.5.5 APIService

Der `APIService` übernimmt zentral die Kommunikation zum API mittels XHR Request. Muss das API wissen um welchen Benutzer es sich handelt, kann das JWT Token übermittelt werden. Das API kann verifizieren, ob das Token unverändert ist und noch nicht abgelaufen.

Das übermitteln des Tokens ist dank dem HTTPS Protokoll kryptographisch geschützt.

Die Einzige Ausnahme ist das Versenden des Login Links in UC3 Passwort vergessen, da technisch nicht sichergestellt werden kann, dass das E-Mail auch verschlüsselt übermittelt wird.

4.5.6 Routing / NavigationService

Der `NavigationService` stellt bekannte Routen der Angular Applikation – welche ebenfalls über den `APIService` bezogen werden – bereit. Der Router kann Änderungen an der URL feststellen und wählt die hinterlegte Komponente aus, welche den Inhalt darstellen kann.

Einige immer sichtbare Komponenten wie die `HeaderBreadcrumpComponent` oder die `NavComponent` reagieren ebenfalls auf Änderungen an der URL. Damit dieser Zustand Zentral sichtbar ist, stellt der `NavigationReducer` die URL ebenfalls in der Zustandsmaschine dar.

4.5.7 Widgets

Die Website verwendet Widgets aus der Material Bibliothek für Angular. Diese werden über das `MaterialModule` zentral zur Verfügung gestellt. Das Modul exportiert die tatsächlich verwendeten Material Komponenten.

4.5.8 In Place Edit

Für das Bearbeiten von Inhalten wird die Content Tools Bibliothe [23] verwendet.

Während der Arbeit wurden verschiedene Bibliotheken angeschaut. Es stellte sich jedoch heraus, dass die Content Tools Bibliothek sich am besten eignet. Die Gründe sind die einfache Erweiterbarkeit und dass die Bibliothek mit einer freien Lizenz verfügbar ist.

Die Bibliothek wird über die `InlineEditComponent` bereitgestellt. Der `ContentToolsService` abstrahiert die Bibliothek und macht diese für Angular nutzbar. Diese Komponente wurde nicht von Grund auf neu Entwickelt, jedoch stark modifiziert. Die `ContentToolsDirective` Implementiert einen `ControlValueAccessor` damit Angular mit dem nativen HTML Element kommunizieren kann.

Es gibt eine Klasse `ImageUploader` welche verwendet wird um hochgeladene Bilder zu Bearbeiten. Zurzeit wird Rotieren und Zuschneiden unterstützt. Das Bild wird anschliessend in `base64` encodiert und kann gespeichert werden.

Der Begriff In Place Edit bedeutet, dass eine Website einen Ansichtsmodus und einen Bearbeitungsmodus hat. Es kann zwischen diesen Modi umgeschaltet werden ohne die Seite neu zu Laden und das Mitglied sieht die Daten so wie sie später auch gerendert werden.

Die im Projekt enthaltene ContentTools Komponente ist nicht selber Entwickelt, sondern basiert auf Kommentaren [24] welche in einer Fehlerbeschreibung gefunden wurden. Diese liessen sich nicht einfach referenzieren, somit wurde die Komponente in den Quelltext übertragen!

Abbildung 60: content tools

Die Abbildung 60 Zeigt die Komponente im Einsatz. Ein Mitglied hat das bearbeiten der Seite aktiviert. Der dargestellte Inhalt hat sich umgewandelt und ist bearbeitbar. Ein Overlay zeigt einen Eigenschaften Dialog welcher je nach markiertem Element verschiedene Einstellungen ermöglicht.

4.5.9 Abläufe im API

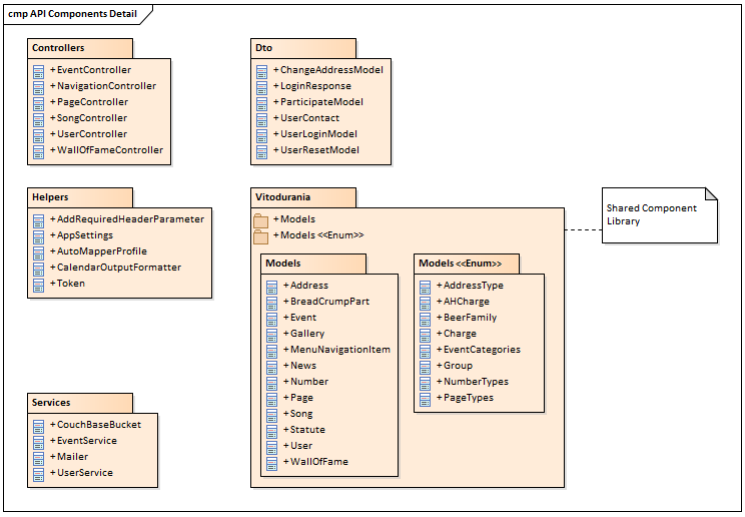


Abbildung 61: Klassen im API

Die Kontroller Klassen implementieren Aktionen welche mit Hilfe des `CouchBaseBucket` Service oder dem `EventService` und `UserService` Abfragen an den Couchbase Server machen und die Resultate als Antwort zurücksenden.

Die Service Klassen `EventService` und `UserService` lagern NQL1 Abfragen gegen den Couchbase Server aus, damit diese wiederverwendet werden können. Teilweise sind nur einmal verwendete NQL1 Abfragen noch in den Kontroller Klassen enthalten. Diese sollten ebenfalls ausgelagert werden.

Die Service Klasse `Mailer` kann E-Mails via SMTP Protokoll versenden.

Die DTO Klassen sorgen dafür, dass das API nur Daten übermittelt welche auch benötigt werden.

Die Klasse `AutoMapperProfile` kümmert sich darum, dass vom DTO `UserLoginModel` zum Model `User` gemappt und umgekehrt werde kann. Gleiches gilt für das DTO `UserContact` und dem Model `User`.

Die Hilfsklasse `Token` bündelt das Generieren und Verifizieren von JWT Tokens. Das Token selbst ist keine Klasse, da es sich um einen normalen `String` handelt, welcher aus drei `base64` enkodierten Teilen besteht. Die genaue Spezifikation wurde aus der JWT Dokumentation [25] entnommen werden. JWT steht für JSON Web Tokens was ein Industrie Standard nach RFC 7519 [26] ist.

Die Enumerationen und Models sind in die Shared Component Bibliothek `Vitodurania` ausgelagert, damit diese auch vom Import Prozess wiederverwendet werden können.

Die Hilfsklasse `AppSettings` kann Einstellungen für das eingesetzte Environment auslesen und mach das API Anpassbar an die Umgebung. Das API kann somit für Entwicklungszwecke anders konfiguriert werden als für die Produktion.

4.6 Prozesse und Threads

Da die VitoSite als JavaScript Applikation läuft, ist diese ein einziger Thread. Es wurden keine Service Worker selbst geschrieben. Jedoch kann die VitoSite in verschiedenen Browsern gleichzeitig geöffnet werden, wobei jede Instanz für sich selbst unabhängig von den anderen Instanzen ist.

Das API wurde ebenfalls nicht Asynchron ausgelegt. Es werden keine Threads oder Queues selbst erstellt.

Das API kann trotzdem mehrere Anfragen gleichzeitig entgegennehmen und verarbeiten, da die Komponente Nginx die Anfragen in einem Event Loop nach Reactor Pattern [27] abarbeitet. Nginx kann so simultan tausende Anfragen verarbeiten. Die Anfragen werden Kestrel [18] über ein Unix Socket von Nginx übergeben. Krestel kann mehrere Event Loops [28] verwenden um Abfragen ebenfalls simultan abzuarbeiten. Der Resultierende API Aufruf läuft dann von Krestel aufgerufen in einem eigenen Thread.

Obwohl die VitoSite und das API somit `single threaded` sind, können dank dieser Middleware Kombination mehrere Anfragen gleichzeitig verarbeitet werden.

Beim Implementieren des API wurde diesem Umstand Rechnung getragen und die verwendeten Service Klassen als Transient eingebunden. Ihre Lebenszeit beschränkt sich somit auf jede Ihrer Verwendung. Werden sie pro Aufruf mehrfach benötigt entstehen sie somit auch mehrfach. In der Regel pro Kontroller Aufruf nur einmal.

4.7 Deployment

Die Konfiguration des Servers befindet sich in der Betriebsdokumentation. Es stehen Deployment Skripte bereits um die einzelnen Applikationen zu deployen.

API:

```
@echo off

REM clean build
rm -rf Vitodurania.CMD\build

REM build project
dotnet publish -c Release -o build Vitodurania.API\Vitodurania.API.csproj

REM clean target
ssh vitodurania@82.220.38.72 "rm -rf /home/vitodurania/api/*"

REM copy files to production
scp -r Vitodurania.API/build/* vitodurania@82.220.38.72:/home/vitodurania/api

REM create symlink to wwwroot
ssh vitodurania@82.220.38.72 "rm -rf /home/vitodurania/api/wwwroot"
ssh vitodurania@82.220.38.72 "ln -s /home/vitodurania/public_html/www_root /home/vitodurania/api/wwwroot"
```

WWW:

```
@echo off

REM in Vitodurania.WWW
cd Vitodurania.WWW

REM clean build
rm -rf dist

REM build package
npm run build

REM clean target
ssh vitodurania@82.220.38.72 "rm -rf /home/vitodurania/public_html/www_dotnet/*"

REM copy files
scp -r dist/* vitodurania@82.220.38.72:/home/vitodurania/public_html/www_dotnet

REM fix permission
REM echo "deploying"
REM ssh parsemall@172.16.11.4 "cd /home/parsemall/public_html/vito_www && find . -type d|xargs chmod 755"
REM ssh parsemall@172.16.11.4 "cd /home/parsemall/public_html/vito_www && find . -type f|xargs chmod 644"

REM back to Vitodurania.Net
cd ..
```

Diese wurden auch auf dem Visual Studio Team Services hinterlegt, damit das Deployment halb automatisiert wurde. Deployments müssen im VSTS manuell angestossen werden, da zwingend die Version der Artefakte gewählt werden müssen. Das Deployment unterscheidet zwischen QA (Qualifikation) und PROD (Produktion). Die genaue Konfiguration kann im VSTS angeschaut werden und die Übersicht dazu sieht wie auf der folgenden Abbildung aus. Die Produktion ist noch nicht konfiguriert!

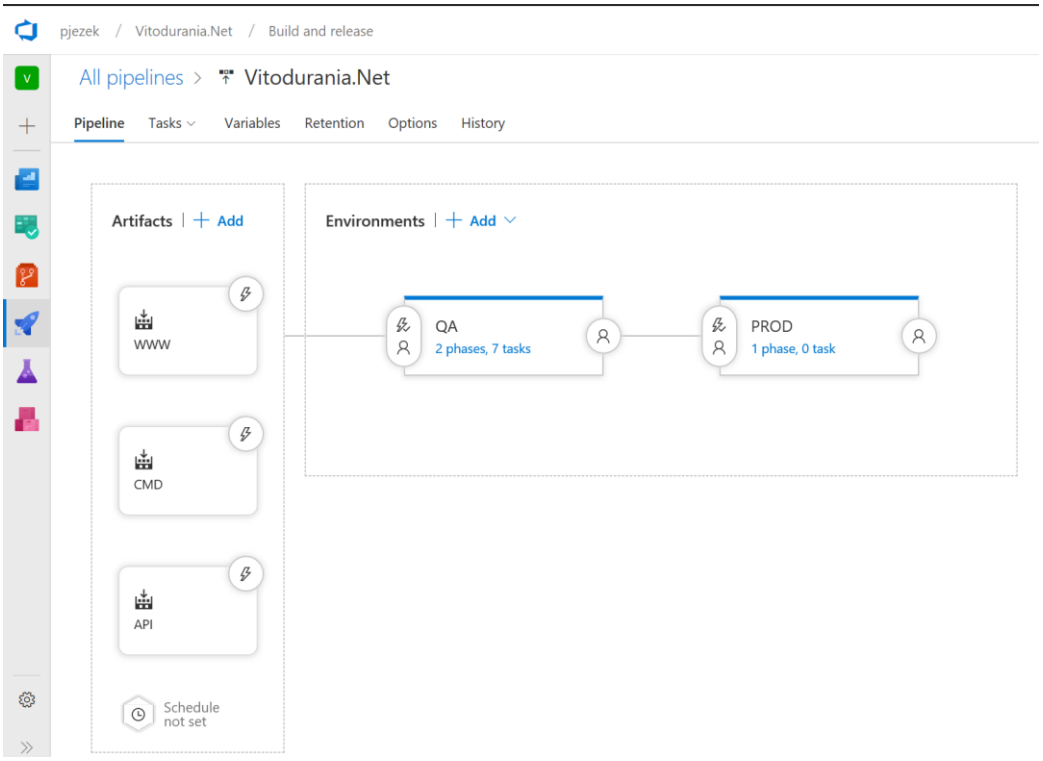


Abbildung 62: Release Management

Die Artefakte werden Vorgängig bei jeder Codeänderung gebaut, wenn die vorgelagerten Tests erfolgreich waren.

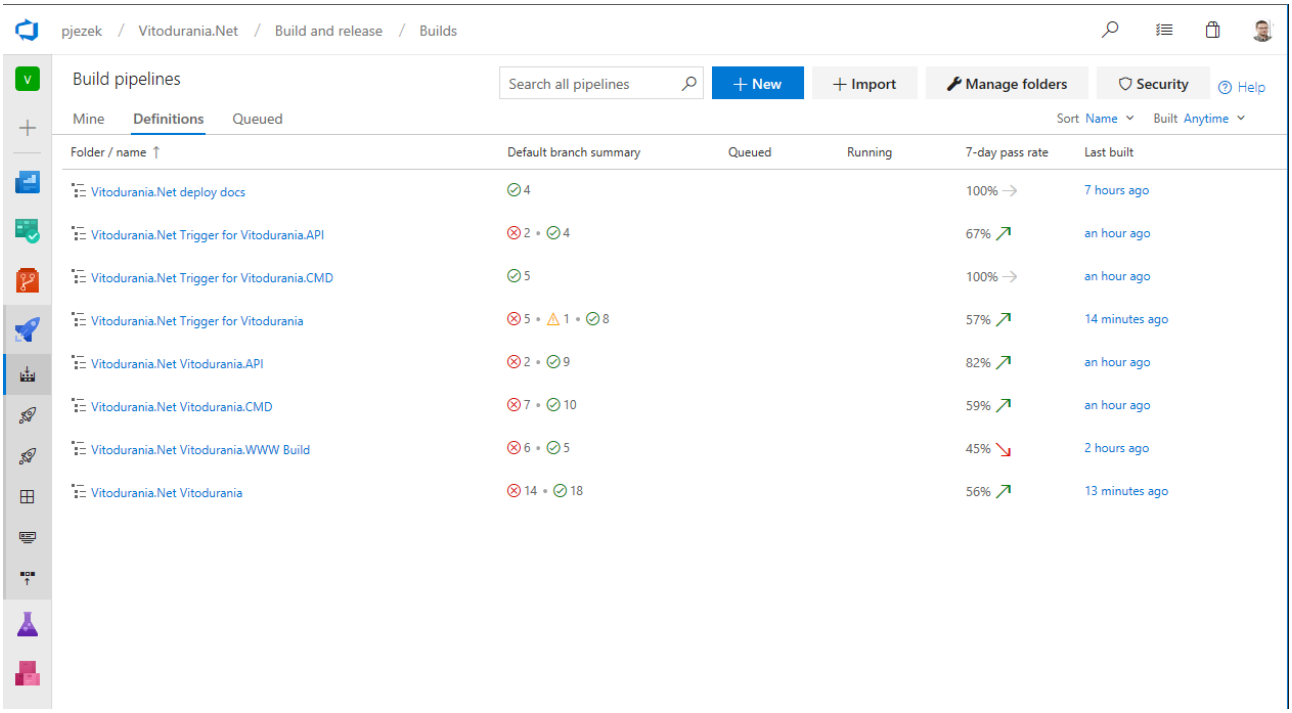


Abbildung 63: Build Pipelines

4.8 Datenspeicherung

4.8.1 Datenbank

Die Datenspeicherung erfolgt in einem Dokumentenspeicher aus einem einfachen Grund: JavaScript konsumiert JSON Objekte welche zwar als Text übermittelt werden, jedoch einfach zurück in Objekte gemappt werden können. Es liegt nahe genau dieses JSON Dokument auch in die Persistierung zu legen. Ebenso einfach kann das im API eingesetzte Framework mit JSON Objekten umgehen. Die JSON Objekte werden in Klassen abgebildet.

Der Couchbase Server bietet eine graphische Oberfläche an.

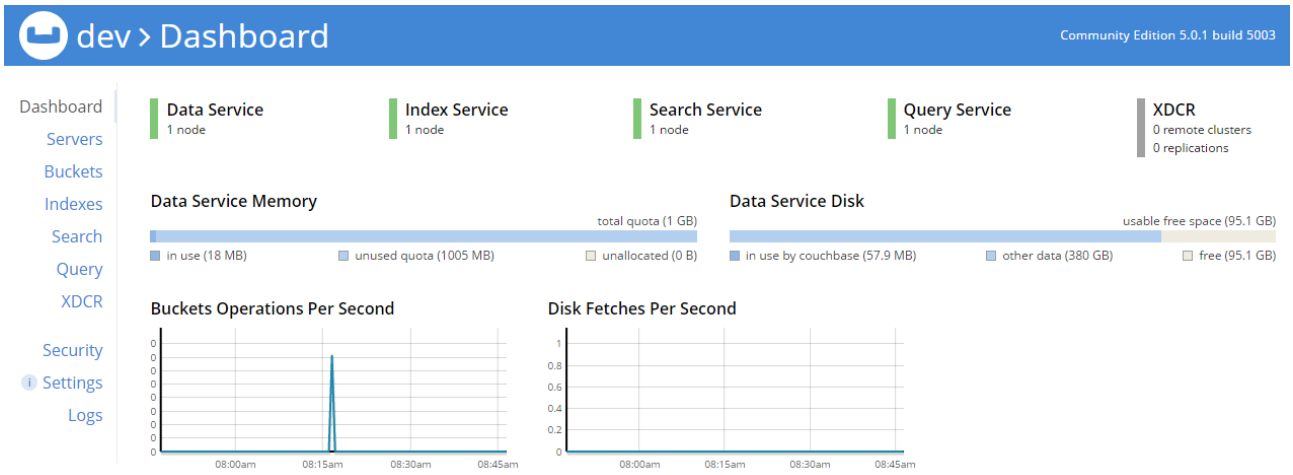


Abbildung 64: Couchbase Server Dashboard

Hier können einfach die Leistungswerte des Servers ausgelesen werden. Es können gespeicherte Dokumente modifiziert werden und NQL1 Abfragen abgesetzt werden.

Der Couchbase Server bietet auch Kommandozeilen Programme an, welche die Verwaltung skriptbar machen. Die Backup- und Restore Strategien sind in der Betriebsdokumentation festgehalten.

4.8.2 Dateien

Nebst (un)strukturieren Daten werden auch Bilder und Musikdateien gespeichert. Diese liegen als gewöhnliche Dateien auf dem Filesystem. Dies bietet den Vorteil, dass die Middleware einfach konfiguriert werden kann, die Auslieferung dieser Dateien mit Cache Headern zu versehen, damit die Browser diese Dateien ebenfalls in den Cache legen. Ein Cache Header kann so aussehen:

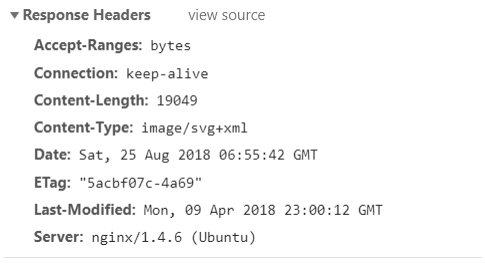


Abbildung 65: Response header with cache information

4.9 Grössen und Leistung

4.9.1 End 2 End Tests

Zwecks Qualitätssicherungen werden e2e [29] Tests eingesetzt. Es handelt sich um `end to end` Tests was Integrationstests entspricht.

Angular setzt für e2e Tests auf Protractor [30]. Es ist ausführlich dokumentiert und lässt sich sehr gut in Angular Applikationen integrieren.

Es werden die Use Cases aus Kapitel 3.1 der *Anforderungsspezifikation* in den Tests abgebildet. Somit kann sichergestellt werden, dass die vom Kunden geforderten Anforderungen abgedeckt sind.

Die e2e Tests laufen auf der Entwicklungsmaschine, da sie einen Webbrowser benötigen. Im Build Prozess des VSTS können die Tests auch auf dem QA System kopflos laufen. Der Browser wird dann in einem Prozess gestartet, ohne dass er sichtbar wird.

Gestartet werden die Tests lokal mittels:

```
ng e2e
```

Im VSTS können die Resultate bei jedem Build im Reiter Tests angeschaut werden

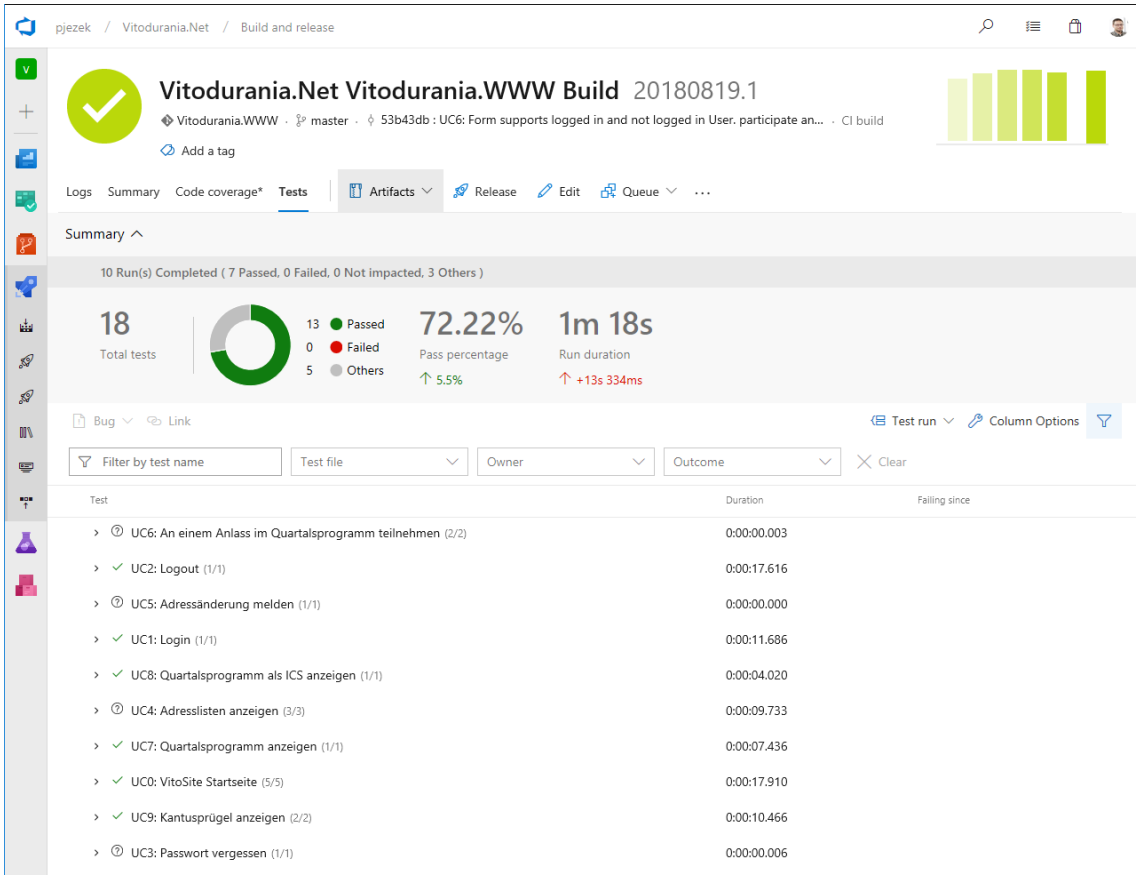


Abbildung 66: VSTS Tests

4.9.1.1 Aufstellung der e2e Tests:

- UC0: VitoSite Startseite
    - sollte den Seiten Titel darstellen
    - sollte den Breadcrump auf Startseite haben
    - sollte News in der dritten Spalte darstellen
    - sollte ein Menu mit 8 Einträgen haben
    - sollte die 4 Quicklinks darstellen
- UC1: Login
    - Ein Mitglied sollte sich einloggen können
- UC2: Logout
    - Ein Mitglied sollte sich ausloggen können
- UC3: Passwort vergessen
    - Ein Mitglied benützt den Passwort vergessen Dialog
- UC4: Adresslisten anzeigen
    - Anonymer Benutzer zeigt die Aktivitas Seite an
    - Anonymer Benutzer zeigt die AH-Vorstand Seite an
    - Anonymer Benutzer zeigt die Kontaktadressen Seite an
- UC5: Adressänderung melden
    - Ein Mitglied meldet seine neue Adresse
- UC6: An einem Anlass im Quartalsprogramm teilnehmen
    - Ein Mitglied meldet sich an einem Anlass an
    - Ein angemeldetes Mitglied meldet sich an einem Anlass an
- UC7: Quartalsprogramm anzeigen
    - Das Quartalsprogramm zeigt die aktuellen 10 Einträge
- UC8: Quartalsprogramm als ICS anzeigen
    - Das Quartalsprogramm zeigt den ICS download Link an
- UC9: Kantusprügel anzeigen
    - Ein Anonymer Benutzer zeigt den Kantusprügel an
    - Ein Anonymer Benutzer zeigt den Farbenkantus an

4.9.2 Unit Tests

Wo es sinnvoll ist wurden auch Unit Tests geschrieben. Diese werden Ebenfalls im VSTS automatisch ausgeführt, sobald eine Quelltext Änderung festgestellt wird.

Im VSTS sieht dies wie folgt aus:

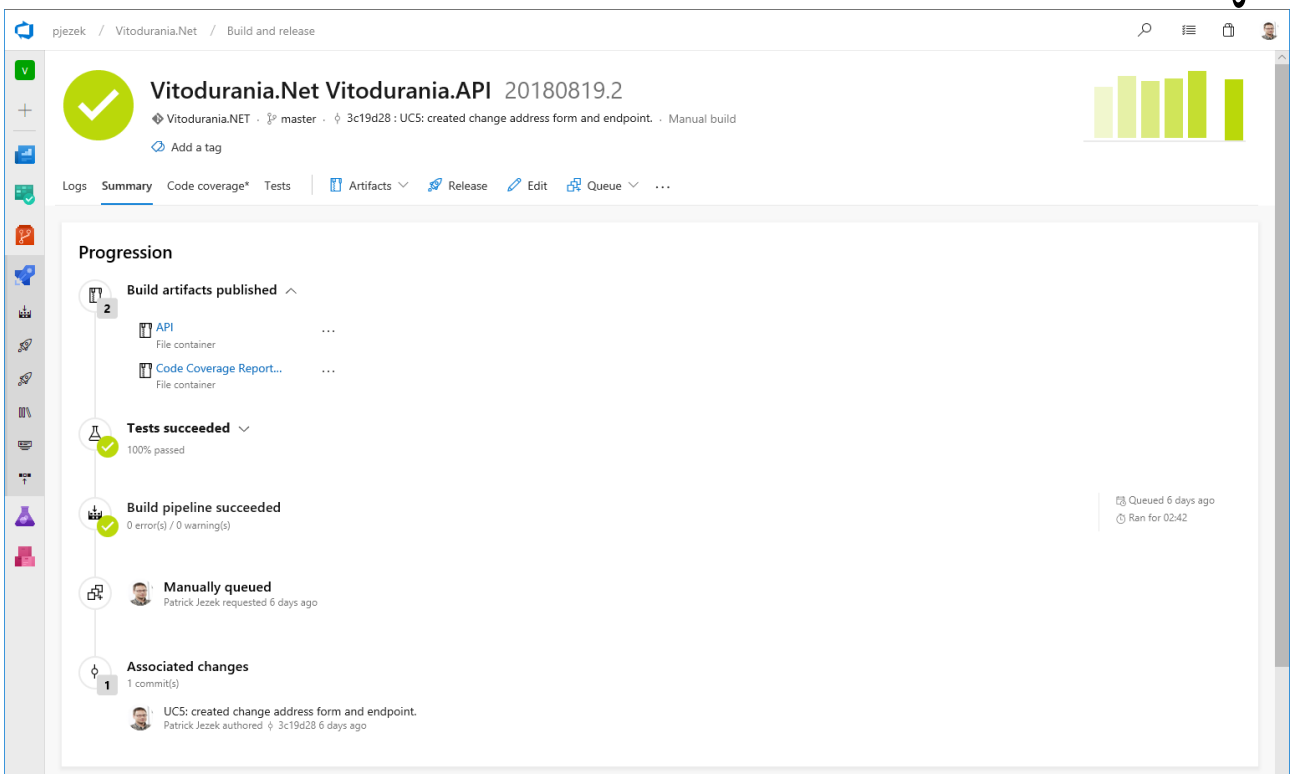


Abbildung 67: triggered build with unittest

4.9.3 Statische Quelltext Analyse

Wenn Quelltext geändert wird, löst dies einen Build Prozess im Ci/CD System aus. Sind die Tests erfolgreich abgeschlossen (siehe Punkt 4.9.1 und 4.9.2) werden die Quelltexte SonarQube für die statische Analyse übergeben.

Leider kann bei E2E Tests keine Coverage ermittelt werden. Es kann aber angenommen, dass eine sehr hohe Abdeckung erreicht wird, da echte Request und Ereignisse auf der VitoSite ausgeführt werden, was alle beteiligten Komponenten mitverwendet.

Es werden dabei keine Mocks verwendet, sondern die auf diesem System ähnlich Aufgebauten Komponenten, wie sie später auf der Produktion auch vorhanden sein werden. Auch die Daten entsprechen denen von der Produktiven Umgebung mit der Ausnahme, dass es sich um eine eigene Instanz der Daten handelt. Diese werden ja durch den Importer hereingeholt.

Das Resultat kann auf der SonarQube Instanz nachgeschaut werden.

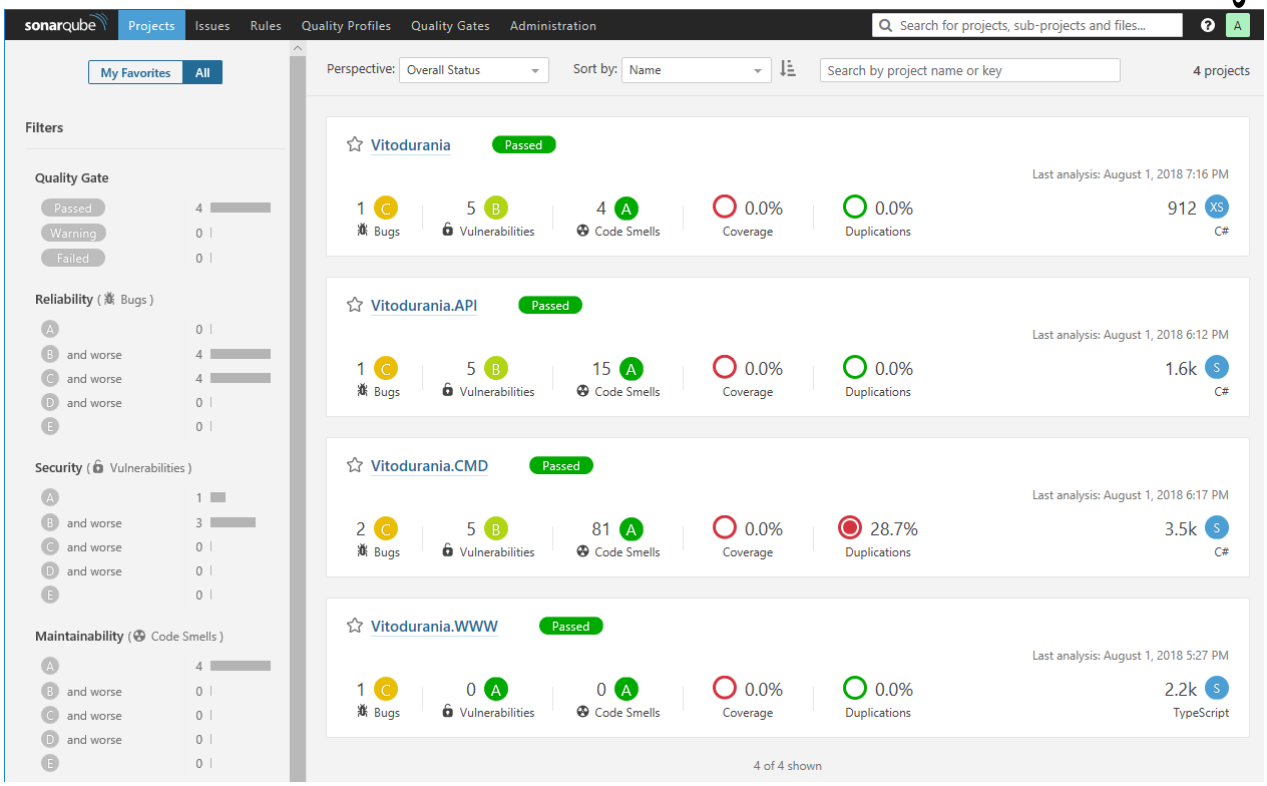


Abbildung 68: SonarQube Dashboard

Dank VSTS Integration lassen sich die Resultate von SonarQube und den Coverage Results ebenfalls im VSTS anschauen:

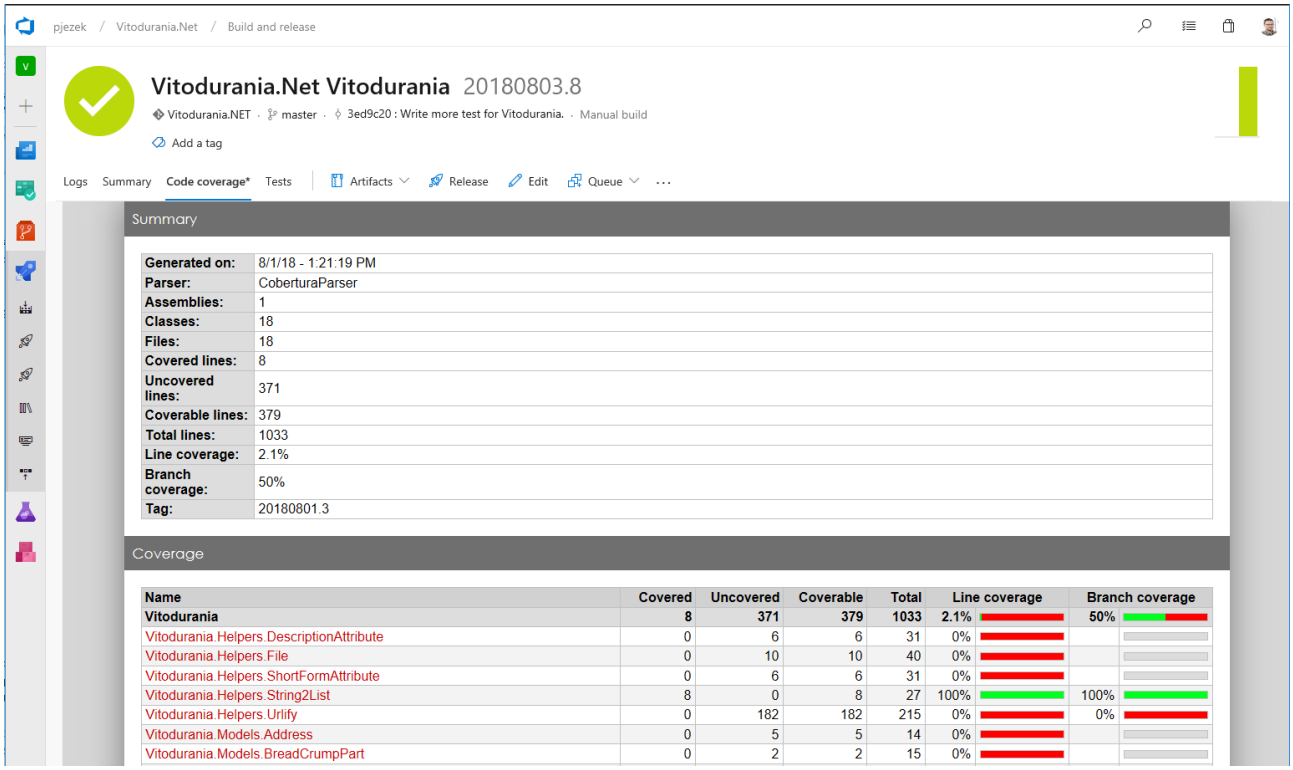


Abbildung 69: code coverage

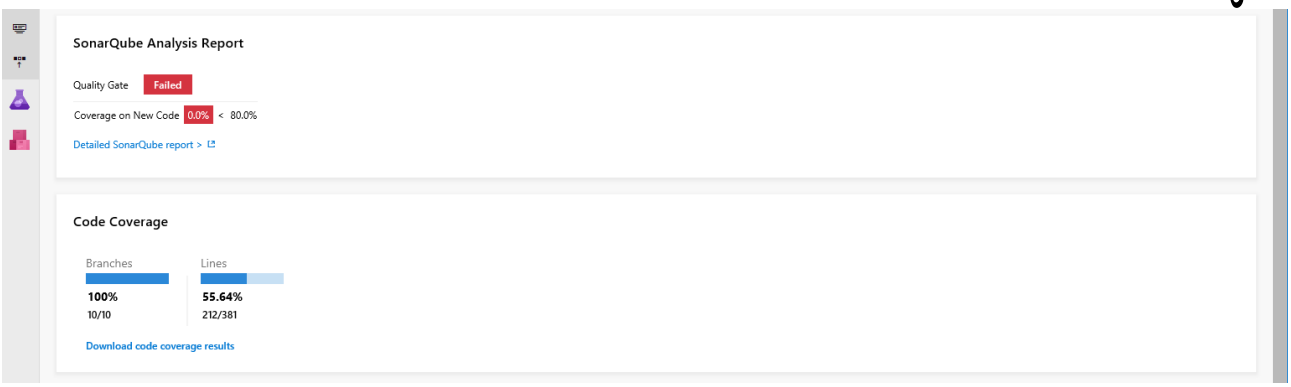


Abbildung 70: VSTS SonarQube integration

4.9.4 Benchmarks

Zu Beginn des Projektes wurde angenommen, dass eine nach Best Pratices entwickelte Applikation dieselbe Leistung erreichen wird, wie von der Beispielsapplikation des verwendeten Frameworks. Eine Beispielsapplikation wurde vom aspnet Team erstellt und nimmt im TechEmpower Web Framework Benchmarks [31] teil.

Im Verlauf des Projektes mussten jedoch Optimierungen vorgenommen werden und anschliessend wurde gemessen, ob die Massnahmen auch Erfolg hatten. Hierzu wurde `ab` [32] verwendet. Ein Aufruf geschieht wie folgt:

```
ab -c 400 -n 4000 http://beta.lo.vitodurania.ch/api/User/byGroup/1
```

Das Flag `c` bedeutet: Anzahl der gleichzeitig auszuführenden Anfragen. Das Flag `n` Anzahl der Anforderungen pro Benchmarking-Sitzung.

Die Einstellung im Beispiel entspricht NFR1 aus dem Kapitel 4.1.1 der *Anforderungsspezifikation*. Das entspricht dem Fall, wenn Neuwahlen sind und die Mitglieder an der Generalversammlung die Adressliste aufrufen, ob die Änderungen bereits auf der VitoSite sind.

```
λ c:\Tools\ab\ab.exe -n 100 -c 10 localhost:5000/User/byGroup/1
This is ApacheBench, Version 2.3 <$Revision: 1826891 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient).....done


Server Software:        Kestrel
Server Hostname:        localhost
Server Port:            5000

Document Path:          /User/byGroup/1
Document Length:        10336 bytes

Concurrency Level:      10
Time taken for tests:   34.038 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      1047500 bytes
HTML transferred:       1033600 bytes
Requests per second:    2.94 [#/sec] (mean)
Time per request:       3403.813 [ms] (mean)
Time per request:       340.381 [ms] (mean, across all concurrent requests)
Transfer rate:          30.05 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.4      0       1
Processing:   405 3277 1249.7   3178    6707
Waiting:      401 3141 1251.4   3053    6665
Total:        405 3277 1249.6   3178    6707

Percentage of the requests served within a certain time (ms)
  50%   3178
  66%   3693
  75%   3967
  80%   4190
  90%   4857
  95%   6054
  98%   6277
  99%   6707
 100%   6707 (longest request)
```

```
λ c:\Tools\ab\ab.exe -n 100 -c 10 localhost:5000/User/byGroup/1
This is ApacheBench, Version 2.3 <$Revision: 1826891 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient).....done


Server Software:        Kestrel
Server Hostname:        localhost
Server Port:            5000

Document Path:          /User/byGroup/1
Document Length:        10336 bytes

Concurrency Level:      10
Time taken for tests:   20.484 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      1047500 bytes
HTML transferred:       1033600 bytes
Requests per second:    4.88 [#/sec] (mean)
Time per request:       2048.422 [ms] (mean)
Time per request:       204.842 [ms] (mean, across all concurrent requests)
Transfer rate:          49.94 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.4      0       1
Processing:   547 1955 747.7   1739    4461
Waiting:      308 1832 719.7   1659    4389
Total:        547 1955 747.8   1739    4461

Percentage of the requests served within a certain time (ms)
  50%   1739
  66%   1998
  75%   2203
  80%   2385
  90%   3225
  95%   3723
  98%   4391
  99%   4461
 100%   4461 (longest request)
```

Abbildung 71: ab test vor (links) und nach (rechts) der Optimierung

5. Umsetzung

5.1 Importer

Der importer kann alle Daten aus der Legacy VitoSite importieren.

```
Run: Vitodurania.CMD\Vitodurania.CMD
"C:\Program Files\dotnet\dotnet.exe" C:/Data/vitodurania/Vitodurania.NET/Vitodurania.CMD/bin/Debug/netcoreapp2.0/Vitodurania.CMD.dll
Use This Program for Maintenance!
Create or update songs
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
Created or updated 82 songs
Create or update statutes
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
Created or updated 53 statues
Create or update WallOfFame
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
Created or updated 36 WallOfFame
```

Abbildung 72: Laufender Import Prozess

Importiert werden die Inhalte:

- Navigationsstruktur
- Seiten und Inhalte
- Benutzer mit Rollen
- News
- Galerien

Erstellt werden folgende Inhalte:

- Lieder
- Statuten
- Vitoduraner des Jahres

Der Importer kann Einträge erstellen (C) oder updaten (U).

Werden Bilder gefunden werden diese auf das Dateisystem geschrieben.

markup	29.07.2018 11:49	File folder	
newmarkup	29.07.2018 11:49	File folder	
pages	29.07.2018 11:50	File folder	
temp_pages	29.07.2018 11:50	File folder	
temp_users	29.07.2018 11:50	File folder	
users	29.07.2018 11:50	File folder	
id_map.json	20.06.2018 07:16	JSON File	96 KB
id_map_flat.json	20.06.2018 07:16	JSON File	8 KB
navigation.json	20.06.2018 07:16	JSON File	84 KB
url_map.json	20.06.2018 07:16	JSON File	96 KB
url_map_flat.json	20.06.2018 07:16	JSON File	8 KB

Abbildung 73: Import Ordner Struktur

Pro Daten Typ wird ein Ordner erstellt auf dem Dateisystem.

0b75fe72-06b8-4f62-bf86-9a52c3886441	29.07.2018 11:50	File folder
0de27132-86e2-44fa-b697-58e7749fbc51	29.07.2018 11:50	File folder
0eab9501-c725-4306-9013-2a0bae51940c	29.07.2018 11:50	File folder
0faffb99-8038-46a8-b83e-03997c6235e7	29.07.2018 11:50	File folder
1a43cc4c-13be-4779-bbd7-ac3267b5b80a	29.07.2018 11:50	File folder
1b5d1839-8461-436a-a83f-851011b6edf8	29.07.2018 11:50	File folder
1d99a1ba-4e91-41b1-a639-502ae6b059c9	29.07.2018 11:50	File folder

Abbildung 74: Import Verzeichnisse nach GUID

Die Dateien werden in einen Ordner mit der GUID des Datensatzes geschrieben, wo sie schliesslich abgelegt werden.

Abbildung 75: Import einzelne Bilder

Bei der Wahl eines Dateinamens wird sichergestellt, dass nur gültige Zeichen verwendet werden. Um Kollisionen von Dateinamen zu vermeiden, wird die alte ID der aus Datenbank vorangestellt. (Hier im Beispiel die ID 57 und 70).

Werden Seiten importiert, kann das Markup ersetzt werden, wenn ein HTML Dokument mit selber ID im Ordner `newmarkup` liegt.

5.2 Datenbank

Als Datenbank wurde ein Couchbase Server gewählt. Es ist ein dokumentbasierter NOSQL Server.

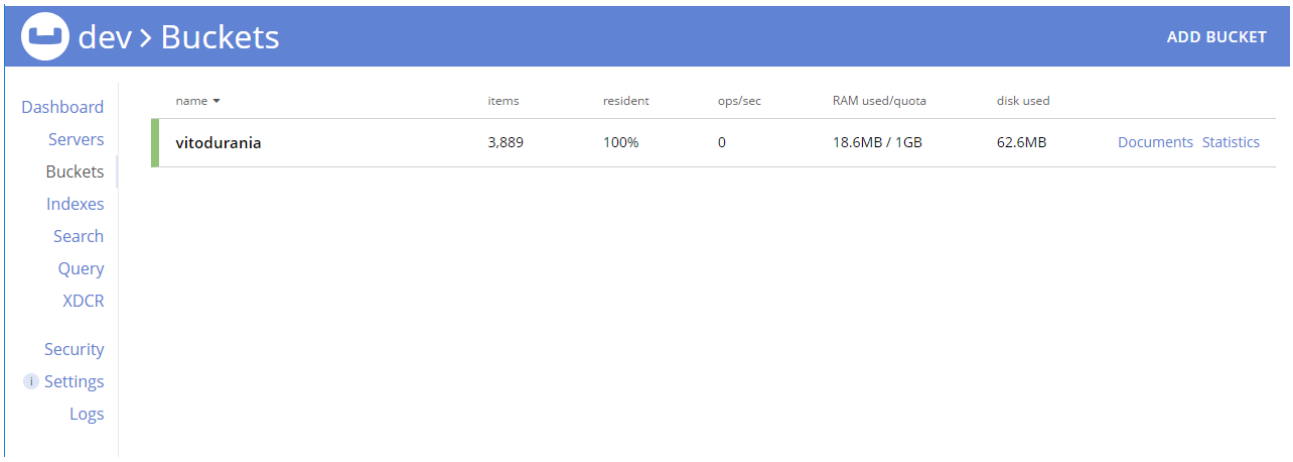


Abbildung 76: couchbase buckets

Die Abbildung zeigt, dass ca. 3900 Dokumente indexiert sind. Das sind die Inhalte, welcher vom Importer geschrieben wurden. Erfreulich ist, dass diese vielen Inhalte wenig Speicher zur Laufzeit benötigen.

Um die Geschwindigkeit der Abfragen zu erhöhen wurden Indices angelegt.

Abbildung 77: couchbase indexes

Das Indices manuell erstellt werden müssen ist etwas ungewohnt, da moderne relationale Datenbanken dies teilweise selbständig machen. Hier liegt noch ein grosses Potenzial an Geschwindigkeitsoptimierungen.

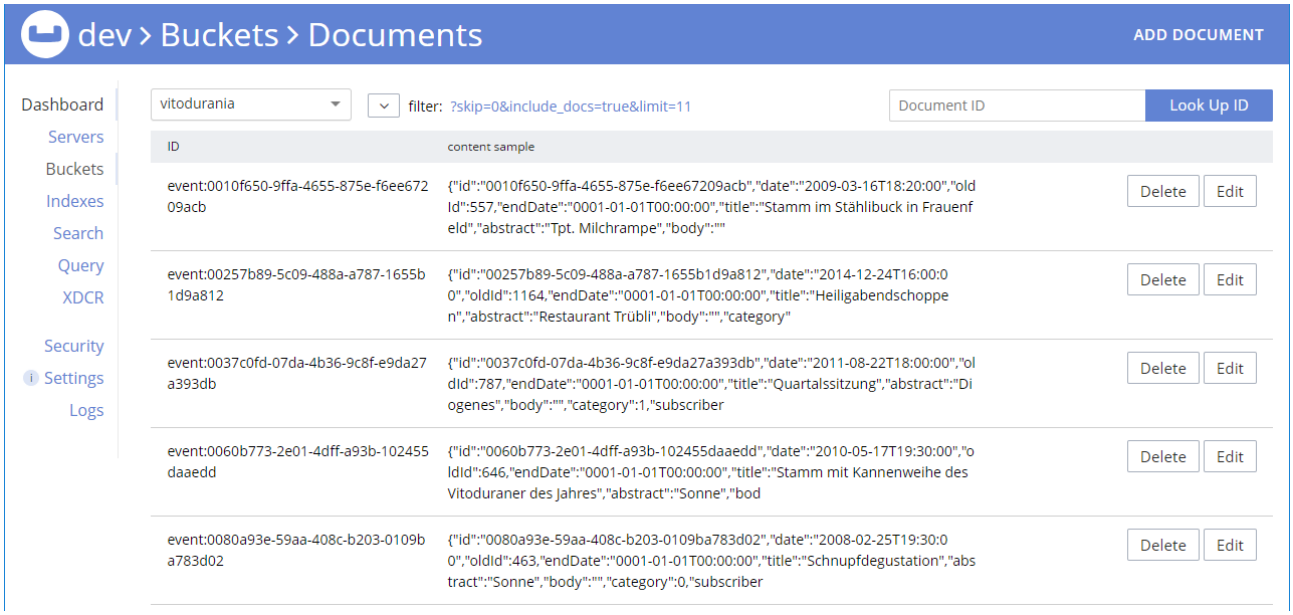


Abbildung 78: couchbase documents

Dokumente können aufgelistet, gesucht und bearbeitet werden. Es muss keine separate Software installiert werden. Die Verwaltung des Servers geschieht im Browser.

Über das XDCR Protokoll können weitere Server angeschlossen werden, welche automatisch synchronisiert werden. In der Evoluations Phase der Masterarbeit wurde so ein ElasticSearch Server angeschlossen, welcher bekannt ist, Volltext Suchen schnell auszuführen. Ein Feature welches später im Verlauf des Projektes verwendet werden kann.

5.3 API

Das API erstellt dank Swagger eine Oberfläche mit welcher es einfach ist, Abfragen auszuprobieren.

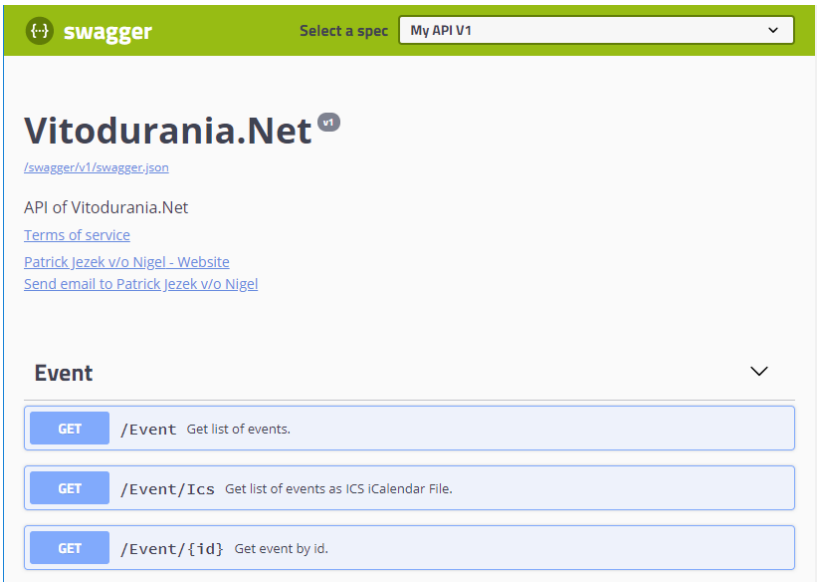


Abbildung 79: Swagger API doc

Die komplette Übersicht ist in Kapitel 4.4 enthalten.

5.4 VitoSite

Die neue VitoSite hält sich stark an das bisherige Design.

Abbildung 80: VitoSite Startseite

Die ist eine Seite, bei welcher das ursprüngliche Markup aufgeräumt ist. Dies war nötig, damit es vom In Place Editor erkannt wird.

Abbildung 81: VitoSite Startseite IPE

Die Adressseiten richten sich im Design mehr an die VitoApp.

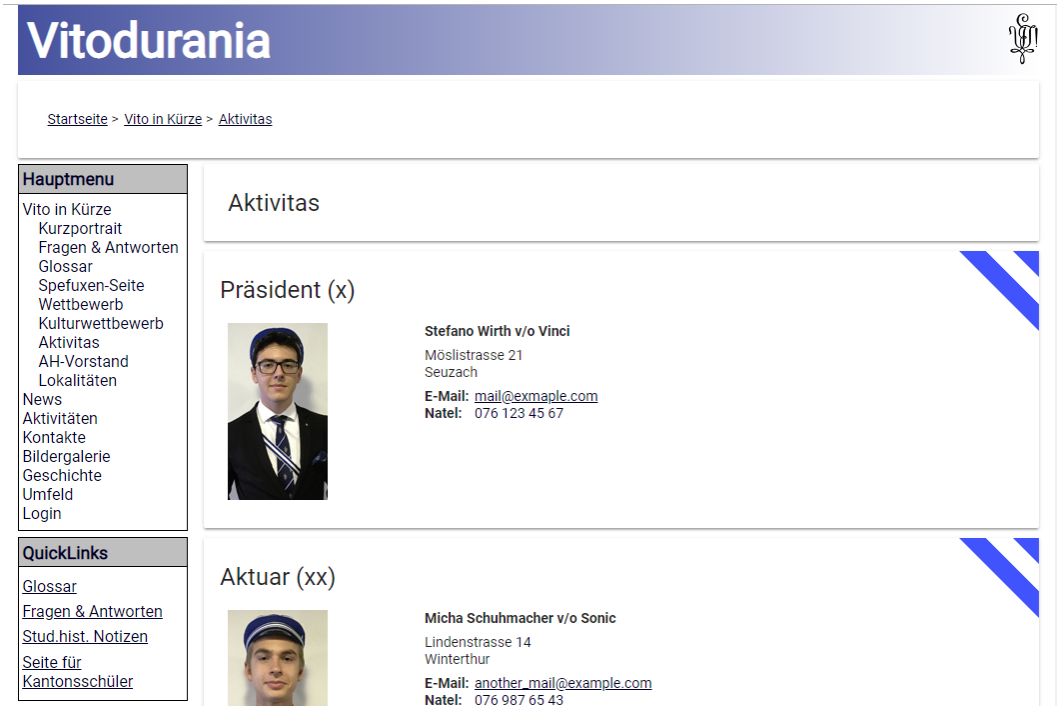


Abbildung 82: VitoSite Aktivitas Seite

Dies hilft auch auf Mobilen Geräten die Seite einfach darstellbar zu haben:

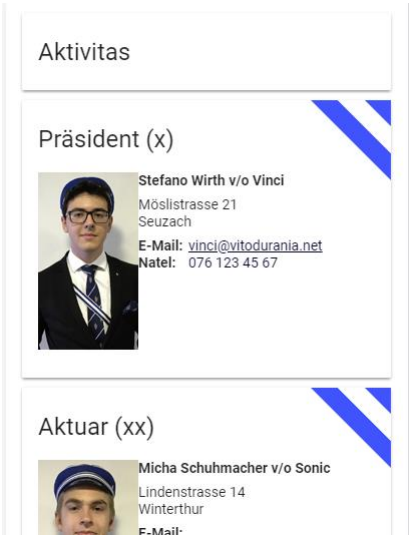


Abbildung 83: VitoSite Aktivitas Seite (Mobile)

Der Kantusprügel enthält dank dem Importer das neue Liederbuch. Die Daten wurden aus der Druckvorlage der neusten Ausgabe entnommen.

Abbildung 84: VitoSite Kantusprügel

Dank HTML5 kann auf der Detailseite des Kantusprügels auf den Flashplayer verzichtet werden und das Abspielen des Liedes funktioniert auch auf Mobilen Geräten.

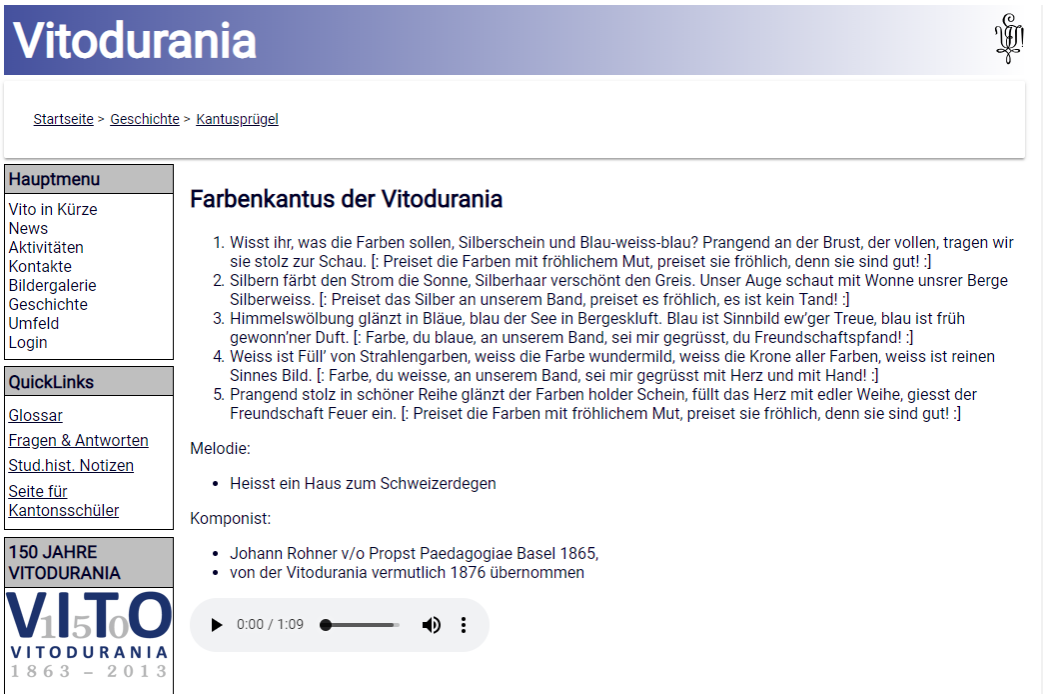


Abbildung 85: VitoSite Farbenkantus

Das Quartalsprogramm sieht ähnlich aus wie in der Legacy VitoSite. Dank einfacherem Markup ist aber auch diese Seite auf Mobilen Geräten noch zu benutzen.

Abbildung 86: VitoSite Quartalsprogramm

Beim An- und Abmelden sind bereits die ersten Verbesserungen in der Benutzerführung sichtbar: Ist das Mitglied eingeloggt, muss es sein Geburtsdatum nicht mehr angeben für eine An- respektive Abmeldung. Neu wird auch der Vulgo hervorgehoben, wenn man bereits angemeldet ist.

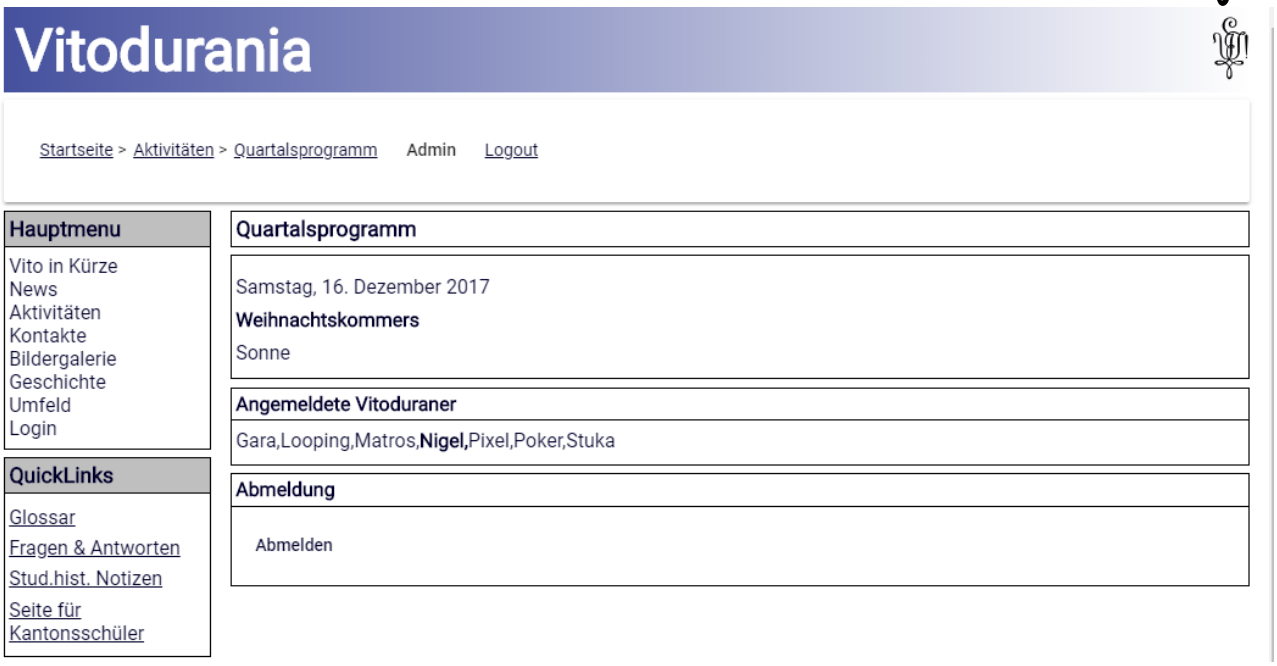


Abbildung 87: VitoSite Quartalsprogramm Eintrag

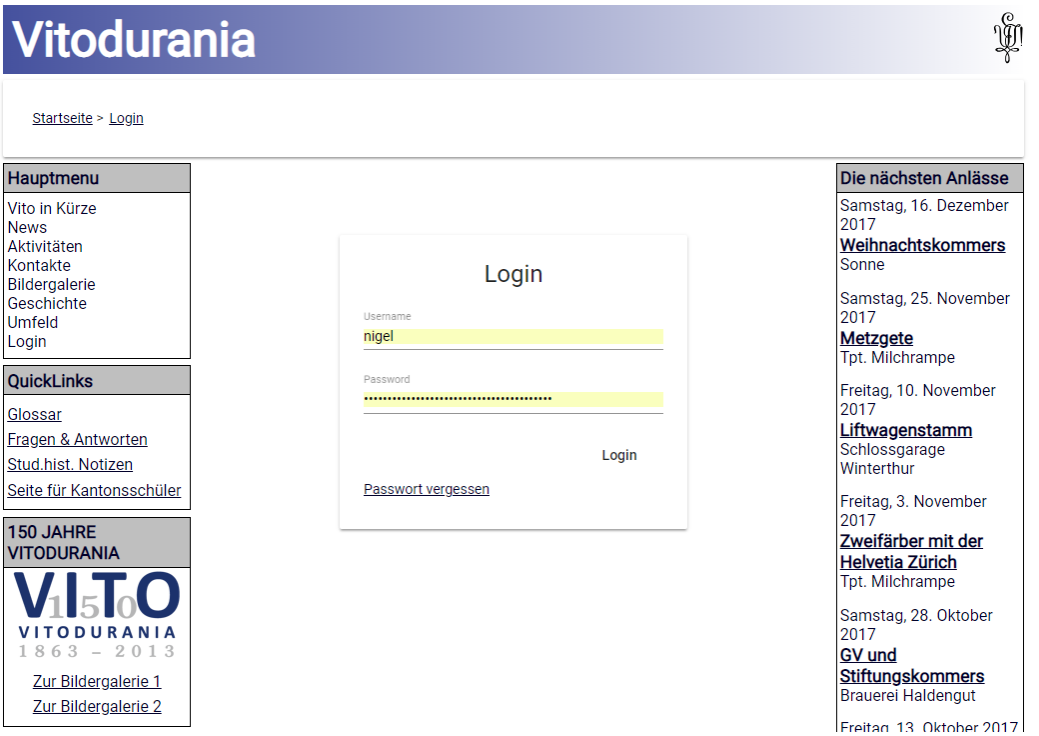


Abbildung 88: VitoSite Login Dialog

Das Login funktioniert auch bereits. Beim Zurücksetzten des Passwortes wurden ebenfalls Verbesserungen an der Benutzerführung vorgenommen. Es steht ein Kalender Widget bereits zur Eingabe des Geburtsdatums. Eingaben werden auch schon auf dem Browser validiert, bevor das Formular auf dem Server geprüft wird.

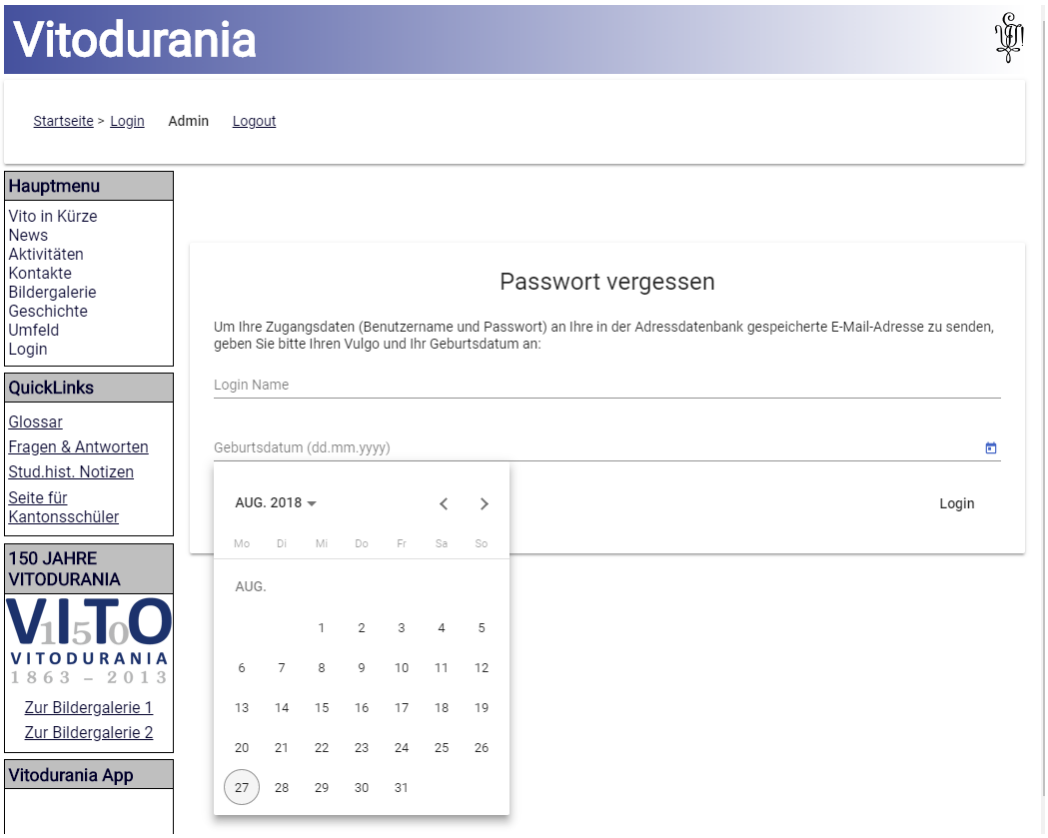


Abbildung 89: VitoSite Passwort vergessen

Das Zurücksetzten des Passwortes wurde vereinfacht. Das Mitglied erhält eine Aufforderung zum Wechseln des Passwortes.

Abbildung 90: Passwort Zurücksetzten E-Mail

Ein Mitglied kann seine Adresse ändern lassen.

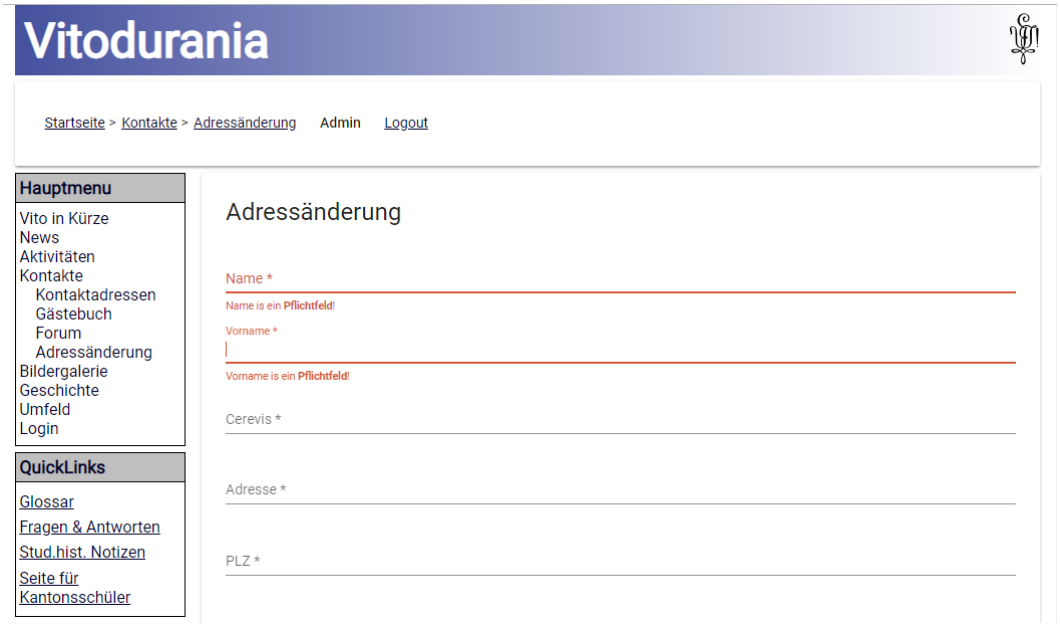


Abbildung 91: VitoSite Adressänderung melden

Auch hier ist die Benutzerführung verbessert. Felder werden bereits vor dem Abschicken validiert. Der Server validiert auch noch.

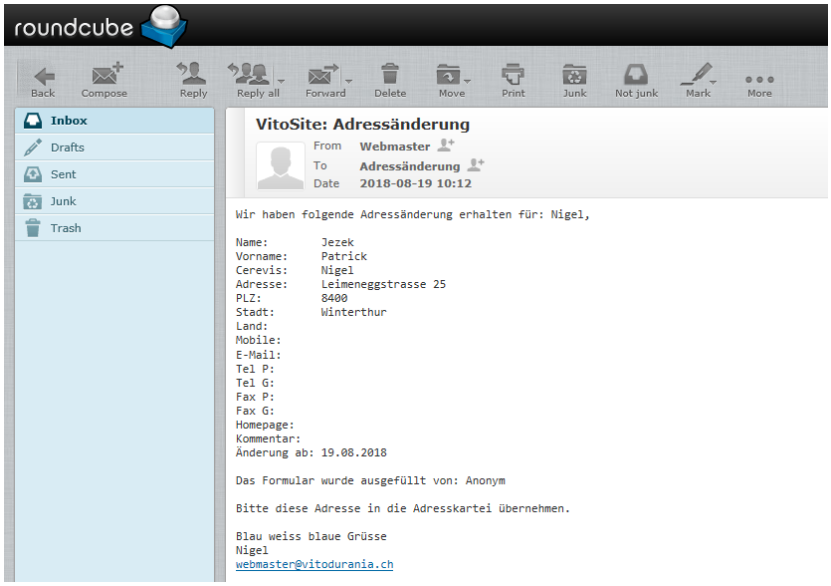


Abbildung 92: VitoSite Adressänderung Mail

Der Prozess einer Adressänderung wurde teilweise vereinfacht: Das Mail erkennt neu ob die Adressänderung von einem angemeldeten Mitglied gesendet wurde. Das hilft bei der manuellen Validierung der Adresse. Hier könnte der Prozess auch automatisiert werden, setzt aber voraus, dass nur noch eingeloggte Mitglieder Adressänderungen melden können.

5.5 Weitere Lieferobjekte

In dieser Masterarbeit Entstand auch die Betriebsdokumentation. Ein Dokument welches Beschreibt, wie die Server und Dienste konfiguriert wurden. Da es aber sehr Technisch ist und die einzelnen Schritte enthält wird verzichtet dieses hier wiederzugeben.

Einzig der Umstand, dass dieses Dokument half, sehr schnell ein Qualifikation System aufzusetzen, als plötzlich die Freiminuten aufgebraucht waren im VSTS. Der Aufwand dieses zu erstellen hat sich somit bereits ausbezahlt.

Das Qualifikation System wurde auch dokumentiert (zu finden im Quelltext unter QA.md), da es Software Komponenten für die Qualitätssicherung enthält, welche auf dem Produktion System nicht verwendet werden.

5.6 Umfang

5.6.1 Zeilen pro Komponente

Komponente	Zeilen	Test Zeilen	Klassen	Komponenten
Shared Components	1264	156	20	-
API	1548	17	23	-
Importer	3110	-	16	-
VitoSite	5449	409	78	24
Total	**11371**	**582**	**137**	**24**

5.6.2 Zeit

Was	Zeit [h]
Vorprojekt	**100**
Projekt	**378**
Dokumentation	125
Infrastruktur	27
Tests	42
Code	184

6. Abbildungsverzeichnis

7. Literaturverzeichnis

[1] php.net, «PHP: Hypertext Preprocessor,» 2018. [Online]. Available: http://php.net/. [Zugriff am 15 04 2018].

[2] E. D.-. u. Ö. (EDÖB), «Umgang mit Mitgliederdaten in einem Verein,» 2018. [Online]. Available: https://www.edoeb.admin.ch/edoeb/de/home/datenschutz/dokumentation/merkblaetter/umgang-mit-mitgliederdaten-in-einem-verein.html. [Zugriff am 15 04 2018].

[3] E. Doug, «What is VSTS,» 2018. [Online]. Available: https://docs.microsoft.com/en-us/vsts/user-guide/what-is-vsts?view=vsts. [Zugriff am 15 04 2018].

[4] sonarqube, «Continuous Code Quality | SonarQube,» 2018. [Online]. Available: https://www.sonarqube.org/. [Zugriff am 15 04 2018].

[5] Canonical, «Ubuntu 14.04.5 LTS (Trusty Tahr),» Canonical, 2018. [Online]. Available: http://releases.ubuntu.com/14.04/. [Zugriff am 15 04 2018].

[6] couchbase, «NoSQL Engagement Database | Couchbase,» 2018. [Online]. Available: https://www.couchbase.com/. [Zugriff am 15 04 2018].

[7] A. Terry, «NGINX | High Performance Load Balancer, Web Server, & Reverse Proxy,» 2018. [Online]. Available: https://www.nginx.com/. [Zugriff am 15 04 2018].

[8] S. Systems, «Enterprise Architect - UML Design Tools and UML CASE tools for software development,» 2018. [Online]. Available: http://sparxsystems.com/products/ea/. [Zugriff am 15 04 2018].

[9] mermaidjs, «mermaid · GitBook,» 2018. [Online]. Available: https://mermaidjs.github.io/. [Zugriff am 15 04 2018].

[10] mermaidjs, «mermaidjs/mermaid.cli: Command-line interface for mermaid,» 2018. [Online]. Available: https://github.com/mermaidjs/mermaid.cli. [Zugriff am 15 04 2018].

[11] JetBrains, «Rider: Cross-platform .NET IDE by JetBrains,» 2018. [Online]. Available: https://www.jetbrains.com/rider/. [Zugriff am 15 04 2018].

[12] Wikipedia, «Kanban (development),» 2018. [Online]. Available: https://en.wikipedia.org/wiki/Kanban_(development). [Zugriff am 15 04 2018].

[13] JetBrains, «Code Analysis - Help | JetBrains Rider,» 2018. [Online]. Available: https://www.jetbrains.com/help/rider/Code_Analysis__Index.html. [Zugriff am 15 04 2018].

[14] Wikipedia, «ISO/IEC 9126 – Wikipedia,» 2018. [Online]. Available: https://de.wikipedia.org/wiki/ISO/IEC_9126. [Zugriff am 15 04 2018].

[15] ngrx, «ngrx/platform: Monorepo for ngrx codebase,» 2018. [Online]. Available: https://github.com/ngrx/platform. [Zugriff am 01 07 2018].

[16] D. Abramov, «Read Me - Redux,» redux.js.org, 2018. [Online]. Available: https://redux.js.org/. [Zugriff am 15 04 2018].

[17] btroncone, «A Comprehensive Introduction to @ngrx/store - Companion to Egghead.io Series,» 2018. [Online]. Available: https://gist.github.com/btroncone/a6e4347326749f938510. [Zugriff am 01 07 2018].

[18] T. e. a. Dykstra, «Kestrel web server implementation in ASP.NET Core,» 2018. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel?tabs=aspnetcore2x. [Zugriff am 15 04 2018].

[19] mozilla.org, «Cross-Origin Resource Sharing (CORS) - HTTP | MDN,» 2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS. [Zugriff am 01 07 2018].

[20] swagger.io, «Swagger UI | API Development Tools | Swagger,» 2018. [Online]. Available: https://swagger.io/tools/swagger-ui/. [Zugriff am 01 07 2018].

[21] Wikipedia, «Request–response - Wikipedia,» 2018. [Online]. Available: https://en.wikipedia.org/wiki/Request%E2%80%93response. [Zugriff am 15 04 2018].

[22] Wikipedia, «Observer pattern - Wikipedia,» 2018. [Online]. Available: https://en.wikipedia.org/wiki/Observer_pattern. [Zugriff am 15 04 2018].

[23] A. Blackshaw, «Try before downloading · ContentTools,» 2018. [Online]. Available: http://getcontenttools.com/demo. [Zugriff am 15 04 2018].

[24] github.com, «Implement ContentTools in to Angular (2+) · Issue #380 · GetmeUK/ContentTools,» 2018. [Online]. Available: https://github.com/GetmeUK/ContentTools/issues/380. [Zugriff am 15 04 2018].

[25] auth0.com, «JSON Web Tokens - jwt.io,» 2018. [Online]. Available: https://jwt.io/. [Zugriff am 15 04 2018].

[26] M. B. Jones, J. Bradley und N. Sakimura, «JSON Web Token (JWT),» 2018. [Online]. Available: https://tools.ietf.org/html/rfc7519. [Zugriff am 15 04 2018].

[27] Wikipedia, «Reactor pattern - Wikipedia,» 2018. [Online]. Available: https://en.wikipedia.org/wiki/Reactor_pattern. [Zugriff am 15 04 2018].

[28] S. Overflow, «asp.net core - Is Kestrel using a single thread for processing requests like Node.js? - Stack Overflow,» 2018. [Online]. Available: https://stackoverflow.com/questions/40948857/is-kestrel-using-a-single-thread-for-processing-requests-like-node-js. [Zugriff am 15 04 2018].

[29] angular, «angular/protractor: E2E test framework for Angular apps,» 2018. [Online]. Available: https://github.com/angular/protractor. [Zugriff am 15 07 2018].

[30] protractortest.org, «Protractor - end-to-end testing for AngularJS,» 2018. [Online]. Available: http://www.protractortest.org/. [Zugriff am 15 07 2018].

[31] techempower.com, «TechEmpower Web Framework Performance Comparison,» 2018. [Online]. Available: https://www.techempower.com/benchmarks/. [Zugriff am 15 04 2018].

[32] apache.org, «ab - Apache HTTP server benchmarking tool - Apache HTTP Server Version 2.4,» 2018. [Online]. Available: https://httpd.apache.org/docs/2.4/programs/ab.html. [Zugriff am 15 04 2018].

[33] Wikipedia, «iCalendar - Wikipedia,» 2018. [Online]. Available: https://en.wikipedia.org/wiki/ICalendar. [Zugriff am 15 04 2018].

[34] Duden, «Duden | Kantus | Rechtschreibung, Bedeutung, Definition, Herkunft,» 2018. [Online]. Available: https://www.duden.de/rechtschreibung/Kantus. [Zugriff am 15 04 2018].

[35] A. Inc, «App Store,» 2018. [Online]. Available: https://itunes.apple.com/de/app/id657432488?mt=8. [Zugriff am 15 04 2018].

[36] Google, «Play Store,» 2018. [Online]. Available: https://play.google.com/store/apps/details?id=ch.vitodurania.vitoapp. [Zugriff am 15 04 2018].

[37] Vitodurania, «VitoSite,» 2018. [Online]. Available: https://vitodurania.ch. [Zugriff am 15 04 2018].

[38] nstudio.io, «nstudio - xplat,» 2018, [Online]. Available: https://nstudio.io/xplat/. [Zugriff am 25 08 2018].

[39] Vitodurania, «VitoSite,» 2018. [Online]. Available: https://vitodurnaia.ch. [Zugriff am 15 04 2018].

[40] JetBrains, «Rider: Cross-platform .NET IDE by JetBrains,» 2018. [Online]. Available: https://www.jetbrains.com/rider/. [Zugriff am 25 08 2018].

8. Glossar

Begriff	Beschreibung
Aktivitas / Aktiver	Aktive Mitglieder (Schüler) der Vitodruania.
Alte Herren / AH	Die alten Herren haben ihre aktive Zeit (Schulzeit) hinter sich und sind Mitglieder des Altherrenvereins.
Altherrenverband	Der Altherrenverein (AHV; auch Altherrenverband), ist der Zusammenschluss der Alten Herren einer Studentenverbindung, also derjenigen Mitglieder, die ihre Mittelschule abgeschlossen haben.
Bierfamilie / Biervater/ Biersohn	Die Mitglieder der Vitodurania hat 5 Bierfamilien. Jedes Mitglied hat einen Biervater welcher sich um den Biersohn (jungen Studenten) kümmert.
ICS	iCalendar ist ein Format welche Termine enthält. Beschreibung und Spezifikation ist auf Wikipedia [33] beschrieben.
Kantus	Wort aus dem Duden [34] welches seinen Ursprung im Lateinischen hat. Es heisst Gesang. Es ist auch ein Lied gemeint, welches im Verein gesungen wird.
Kantusprügel	Liederbuch des Vereins. Es liegen Texte und Lieder als MP3 vor.
Quartalsprogramm	Eventkalender welche die Veranstaltungen des Vereins auflistet. Die Veranstaltungen werden jeweils auf ein Quartal im Jahr geplant. Somit ist ein Jahr in 2 Semestern mit jeweils zwei Quartalen unterteilt.
Quästor	Amt im Verein, welches für die Buchhaltung zuständig ist.
RESTful API	Ist eine Anwendungsprogrammschnittstelle, welche auch als RESTful Web Service bezeichnet wird. Sie basiert auf der Representational State Transfer (REST) Technologie, einem Architekturstil und Ansatz für die Kommunikation, die häufig bei der Entwicklung von Web-Services verwendet wird.
SCL	Shared Component Library: Bibliothek mit wiederverwendbaren Quelltext, welcher zur Laufzeit eingebunden wird.
VitoApp	Kurzname für die Mobile App der Vitodurania, welche für iOS [35] und Android [36] erhältlich ist.
Vitodurania	Die Vitodurania ist eine 1863 gegründete Mittelschulverbindung der drei Kantonsschulen Im Lee, Rychenberg und Büelrain in Winterthur in der Schweiz.
VitoSite	Kurzname der Website der Vitodurania [37]
VSTS	Visual Studio Team Services (VSTS) [3] ist ein Cloud-Service für die Zusammenarbeit bei der Code-Entwicklung. Es bietet neben weiteren folgende Funktionen: GIT Repositories, Build und Release Management, Agile Tools für die Planung und die Fortschrittskontrolle.
Vulgo / Cerevis	Vulgo und dessen Synonym Cerevis ist ein Übername welches jedes Vereinsmitglied bekommen hat. Im Verein spricht man sich mit dem Vulgo an.