Hochschule RheinMain
Fachbereich Design Informatik Medien
Studiengang Wirtschaftsinformatik

Bachelor-Arbeit

zur Erlangung des akademischen Grades
Bachelor of Science - B. Sc.

Prozessgestaltung, Architekturkonzeption und Implementierung des Mitgliedschafts-Antragsprozesses einer integrierten webbasierten Anwendung für die Mitgliederverwaltung in Vereinen

vorgelegt von **Maximilian Rosemeier**
Matrikelnummer 1112959
Reiherstraße / 9
65201 Wiesbaden

am 01.01.1900

Referent: Prof. Dr. Marc-Alexander Zschiegner
Korreferent: Korhan Ekinci

PO 2016/2017

Formales

Erklärung gem. ABPO, Ziff. 4.1.5.4

Ich versichere, dass ich die Bachelor-Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Ort, Datum Unterschrift Studierende/Studierender

Hiermit erkläre ich mein Einverständnis mit den im Folgenden aufgeführten Verbreitungsformen dieser Bachelor-Arbeit:

Verbreitungsform	**ja**	**nein**
Einstellung der Arbeit in die Hochschulbibliothek mit Datenträger	X	
Einstellung der Arbeit in die Hochschulbibliothek ohne Datenträger	X	
Veröffentlichung des Titels der Arbeit im Internet	X	
Veröffentlichung der Arbeit im Internet	X	

Ort, Datum Unterschrift Studierender

Genderhinweis

Aus Gründen der besseren Lesbarkeit wird in dieser Arbeit auf die gleichzeitige Verwendung männlicher und weiblicher Sprachformen verzichtet. Bei Personenbezeichnungen und personenbezogenen Hauptwörtern wird zum Ausdruck die männliche Form gewählt. Sämtliche Personenbezeichnungen gelten im Sinne der Gleichbehandlung für alle Geschlechter. Die verkürzte Sprachform beinhaltet keine Wertung.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

SPOAC Sports Business Academy

BGB Bürgerliches Gesetzbuch

MVC Modell View Controller

MVT Modell View Template

NFA Nicht-funktionale Anforderungen

WSGI Web Server Gateway Interface

URI Uniform Resource Identifier

DRY don’t repeat yourself

Abstract

TODO: Titel Ueberarbeiten

Kapitel 1

Einführung

> Die Digitalisierung aller Lebens- und Arbeitsbereiche fordert ihren Tribut und verschafft zugleich ungeahnte Möglichkeiten.
>
> Markus Baumanns - ∗2019 - *Hochschul- und Wissenschaftsberater*

1.1 Motivation

Die Digitalisierung der Gesellschaft verändert zunehmend unser privates und betriebliches Leben. Dabei nimmt sie einen immer höheren Stellenwert für Privatpersonen und Unternehmen ein [Lei15]. Die Mehrzahl aller Deutschen benutzt derzeit mindestens ein Mobilgerät, sodass die Marktdurchdringung von Smartphones, Apps und dem mobilen Internet immer weiter ansteigt. Dabei erwarten die Nutzer, dass sie auf ihre digitalen Produkte und Dienstleistungen immer und überall zugreifen können. Auf der anderen Seite ordnen Unternehmen dem Megatrend Digitalisierung eine hohe Priorität zu und gehen davon aus, dass die Digitalisierung künftig einen großen Einfluss ihre Geschäftsmodelle haben wird.

Im wirtschaftlichen Bereich ist das Thema Digitalisierung und die zugehörigen transformierenden Eigenschaften bereits umfangreich erforscht. Im Fachgebiet der Sportwissenschaften verzeichnet sich hingegen ein großes Forschungsdefizit [Vol19]. Die Sports Business Academy (SPOAC) veröffentlicht jährlich eine Studie zu den aktuellen Trends und Handlungsfeldern im Leistungssportbusiness. In ihrer Studie aus dem Jahr 2016 kommen sie zu dem Ergebnis, dass die Digitalisierung künftig eine besondere Herausforderung an die Führungskräfte der Sportvereine darstellt [Spo16]. Im Jahr 2017 kommen sie zu dem Ergebnis, dass neue Technologien künftig einen entscheidenden Faktor für die Optimierung bestehender als auch Erschließung neuer digitaler Geschäftsfelder darstellen [Spo17].

Sowohl für Leistungssportvereine als auch für Breitensportvereine zeigt sich laut Otto [Ale18], dass die starke Dynamik und Bandbreite der Digitalisierung Sportvereinen eine große Chance bietet, Angebot, Service sowie Kommunikation zu optimieren. Linda Volkmann [Vol19] gibt der Digitalisierung in Sportvereinen diesbezüglich einen höheren Stellenwert: Vereine, die sich der Digitalisierung verschießen, riskieren es ihrer Meinung nach, den gesellschaftlichen Anschluss und damit auch die gesellschaftliche Relevanz zu verlieren.

Golinsky [Gol20] erstellt in seinem Buch Moderne Vereinsorganisation eine Aufgabenübersicht von Vereinen. Im Anschluss prüft er, welche Aufgabengebiete mit aktuellen Vereinssoftware-Anbietern wie z.B. webling, Mein Verein oder Members-App digital bearbeitet werden können. Dabei kommt er zu dem Schluss, dass kein Anbieter die Bearbeitung aller Aufgabengebiete anbietet. Beispielsweise stellt die Bindung und Gewinnung von ehrenamtlich Engagierten nach Volkmann [Vol19] derzeit eine große Herausforderung an Sportvereine. Aktuelle Vereinssoftwarelösungen bieten Funktionen, um einen Verein und seine Mitglieder zu verwalten, sodass nicht mit Interessengruppen außerhalb der Sportvereine interagiert werden kann. Dazu gehören auch Sponsoren, Fans und Personen, die sich für eine Mitgliedschaft im Verein interessieren. Darüber hinaus sind Sportler Personen und können während ihres Lebens in mehreren Vereinen eine Mitgliedschaft besitzen. Mit der Softwarearchitektur derzeitiger Vereinssoftware lassen sich diese Funktionen nicht darstellen, da die entsprechenden Interessengruppen keine Entitäten in der Vereinssoftware besitzen und zwischen Vereinen und Mitgliedern ein 1 (Verein) zu n (Mitgliedern) Verhältnis besteht.

Somit lässt sich sagen, dass die Architektur der aktuellen Vereinssoftware die Erwartungen der Anwender nicht vollständig erfüllen kann. Um die Digitalisierung in Sportvereinen qualitativ zu verbessern, muss eine Softwarearchitektur geschaffen werden, bei der neben den Sportvereinen und Mitgliedern auch die Interessengruppen außerhalb der Sportvereine berücksichtigt werden. Dafür müssen Personen unabhängig von ihrer Vereinszugehörigkeit als eigene Entität in Vereinssoftware erfasst werden.

1.2 Zielsetzung

Das Ziel dieser Bachelorarbeit ist es, eine Softwarearchitektur für Vereinssoftware bereitzustellen, mit der in Zukunft die Prozesse zwischen Sportvereinen und ihren Interessengruppen weiter digitalisiert werden können. Dafür ist es grundsätzlich notwendig, dass Menschen unabhängig Ihrer Vereinszugehörigkeit die Anwendung als User nutzen können und in unterschiedlichen Rollen mit Vereinen kommunizieren können. Beispielsweise sollen Anwender auf der einen Seite als Fans mit Vereinen kommunizieren können und mit anderen Vereinen als Mitglied interagieren können.

Durch die neue Architektur soll zwischen den Interessengruppen und Vereinen ein n zu n Verhältnis ermöglicht werden, sodass Vereine Beispielweise mit mehreren Fans, Mitgliedern, Sponsoren, etc. interagieren können. Die individuellen User stellen dabei Menschen dar und können ebenfalls mit mehreren Vereinen als Fans, Mitglieder, Sponsoren, etc. interagieren. Da mit der neuen Architektur auch Personen ohne Mitgliedschaft in Vereinen die Vereinssoftware nutzen können, ist es für die Sportvereine wichtig zwischen Mitgliedern und externen Interessengruppen unterscheiden zu können.

Der in Kapitel 2.1.3 definierte Anmeldeprozess von Personen bei Sportvereinen spielt dabei eine wichtige Rolle. Bei den Anmeldeprozessen derzeitiger Vereinssoftware meldet sich eine unregistrierte Person bei einer Vereins-spezifischen Website an. Mit der erarbeiteten Softwarearchitektur sollen registrierte Personen die Möglichkeit haben, über eine integrierte webbasierte Anwendung Informationen über mehrere Vereine betrachten zu können. Darüber hinaus sollen sie die Möglichkeit haben, an einen Verein ihrer Wahl einen Mitgliedschaftsantrag zu senden. Für die in dieser Bachelorarbeit erarbeitete Softwarearchitektur können die Anmeldeprozesse der aktuellen Vereinssoftwareanbieter aufgrund der geschilderten Differenzen nicht genutzt werden. Da der Anmeldeprozess eine spezifische und neuartige Herausforderung darstellt, wird er in dieser Arbeit in einem Prototypen implementiert.

1.3 Aufbau der Arbeit

Mit dieser Arbeit werden in einem kaum erforschten Fachgebiet Probleme analysiert und es werden mögliche Lösungen entwickelt. Dazu wird der in Kapitel 2.1.1 geschilderte Ansatz einer prototypischen Entwicklung verwendet: Im Fokus dieser Arbeit steht nicht die Bereitstellung einer finalen Vereinssoftware. Primär sollen Lösungen für die Defizite aktueller Vereinssoftware erarbeitet werden und in einem Prototyp implementiert werden, sodass im Anschluss an diese Arbeit mit der Weiterentwicklung und Verprobung des Prototypen fortgefahren werden kann. Der Prototyp baut auf die Ergebnisse des VEMA Projektes aus dem Wintersemester 2020/2021 auf. Hier wurden bereits erste Features, wie beispielsweise der Registrierungsprozess auf der Website, umgesetzt. Die Softwarearchitektur sowie der Anmeldeprozess müssen allerdings auch hier grundlegend überarbeitet werden.

Um eine Softwarearchitektur und einen Anmeldeprozess zu gestalten, ist diese Bachelorarbeit in drei Schwerpunkte gegliedert. Im ersten Schwerpunkt befasst sich diese Thesis in dem Kapitel 3 mit der Gestaltung der Softwarearchitektur. Dazu wird geprüft,

wie die Lösungsansätze in anderen Branchen auf die Ziele dieser Arbeit angewendet werden können und welche Anpassungen notwendig sind. Die Ergebnisse werden für die Gestaltung einer entsprechenden Softwarearchitektur genutzt, in der Menschen unabhängig von ihrer Vereinszugehörigkeit als individuelle User dargestellt werden können. Dabei liegt der Fokus auf Funktionen, die spezifisch für eine Vereinssoftware sind und nicht eins zu eins von anderen Beispielen kopiert werden können.

Im zweiten Schwerpunkt wird der in Kapitel 2.1.3 definierte Anmeldeprozess eines registrierten Users bei einem bestehenden Verein gestaltet. Dafür werden zu Beginn in dem Kapitel 2 alle notwendigen fachlichen und rechtlichen Grundlagen dargestellt. Im Kapitel 4 werden auf Basis der Grundlagen die Anforderungen an den Anmeldeprozess herausgearbeitet und der Anmeldeprozess wird abschließend gestaltet. Der Anmeldeprozess ist dabei grundsätzlich bei Anbietern wie webling bereits vorhanden. Mit der überarbeiteten Architektur dieser Bachelorarbeit können die bestehenden Anmeldeprozesse die fachlichen und technischen Anforderungen nicht mehr erfüllen und müssen daher ebenfalls überarbeitet werden.

Der dritte Schwerpunkt befasst sich in dem Kapitel 5 mit der Implementierung der Ergebnisse aus dem zweiten Schwerpunkt. Zu Beginn dieses Schwerpunktes wird der Ist-Zustand des Quellcodes erhoben, damit darauf hin die Handlungsfelder hergeleitet werden können. Die entsprechenden Anpassungen werden abschließend im Front- und Backend umgesetzt.

Zum Abschluss werden die Ergebnisse dieser Bachelor Arbeit reflektiert und kritisch betrachtet. Dabei wird geprüft, ob die entstandene Softwarearchitektur und der implementierte Anmeldeprozess alle Anforderungen erfüllt, um künftige Herausforderungen der Digitalisierung an Sportvereine zu bewältigen. Aufgrund des Zeitfensters wird innerhalb dieser Bachelorarbeit keine finale Vereinssoftware erarbeitet. Für die Zukunft wird daher ein Ausblick zu den künftigen Auswirkungen der Digitalisierung auf Sportvereine gegeben. Abschließend werden diesbezüglich die nächsten Schritte und Handlungsfelder dargestellt.

1.4 Umfang der Arbeit

DIESES KAPITEL IST NOCH NICHT VERFASST. DIE STICHPUNKTE SIND NUR NOTIZEN

→ Vordergrund dieser Arbeit: Mehrwert für Mitglieder/ Interessengruppen sowie Kommunikation Verein, mit ihnen herstellen. nicht Prozessoptimierung bestehender Verwaltungsaufgaben

- interne Verwaltungsaufgaben sind in bestehender Software bereits umgesetzt. zwar auf verschiedene Anbieter verteilt, aber allgemein bereits größtenteils erforscht.
→ Vereinsämter sind nicht Zielgruppe der Arbeit, werden aber bei Konzeption berücksichtigt
- Nur Sportvereine (Folgestufe denkbar)
→ Nur Verwaltungssoftware, keine Finanzsoftware
→ Für externe Interessengruppen werden hier digitale Kommunikations/Interaktionskanäle geschaffen/ optimiert
- Mitglieder
- Interessenten
- interessenten am ehrenamt
- fans
- merch Shopper
- Sponsoren
- gibt noch mehr (beispielsweise staat)/ detailierter. wegen Umfang der arbeit nur diese betrachtet und folgende aufgaben: - Fans (gewinnen, interagieren)
- Merchandising(nur anbieten, nicht bestellen... Eshops/ Bezahlprozesse sind nix neues, es werden die üblichen mittel genutzt)
- Interessenten (gewinnen, beraten)
- Anmeldeprozess
- Gewinnung und Betreuung ehrenamtlicher Mitarbeiter und von Unterstützern
- Sponsoren (gewinnen, interagieren)
- Netzwerk und Kontaktpflege

→ Abgrenzung Anmeldung zu einem Verein / Registrierung bei Vereinssoftware
→ Abgrenzung beitritt nur bei bestehendem Verein. Nicht bei Gründungsversammlung
→ Datenschutz
→ keine primär Erhebung
→ aufgrund des Zeitfensters keine finale Implementierung möglich

Kapitel 2

Theoretische Grundlagen

Zu Beginn dieser Arbeit werden in diesem Kapitel die theoretischen Grundlagen von Sportvereinen und der zugehörigen Vereinssoftware dargestellt. Dazu werden zuerst wichtige Begriffe für diese Arbeit definiert. Im Anschluss werden die Kennzeichen und Besonderheiten von Sportvereinen zusammengefasst, die für die Gestaltung eines Anmeldeprozesses und einer Softwarearchitektur mit einem n zu n Verhältnis zwischen Sportverein und Interessengruppen maßgeblich sind.

2.1 Definitionen

2.1.1 Prototypische Entwicklung

Prototypische Entwicklung ist ein Vorgehensmodell für die Entwicklung einer Software [Kle18]. Sie zeichnet sich dadurch aus, dass sie sich auf potenzielle Risiken und Probleme vor dem Beginn der eigentlichen Entwicklung ausrichtet. Der Ansatz eignet sich für die Arbeit in einem neuen oder kaum erforschten Fachgebiet, da schnellstmöglich Feedback von Kunden eingeholt werden kann. Allgemein orientiert sich das Modell an den Phasen des Wasserfallmodells (Anforderungsanalyse, Grobdesign, Feindesign, Implementierung, Test und Integration). Die Prototypische Entwicklung findet sich allerdings auch in allen aktuellen Vorgehensmodellen wieder. Das Produkt, der Prototyp, kann bei Beginn der eigentlichen Entwicklungsphase übernommen oder verworfen werden.

2.1.2 Vereinssatzung

Nach dem Bürgerlichen Gesetzbuch (BGB) bezeichnet Satzung die schriftliche Grundordnung eines Vereines [14]. Der Gesetzgeber lässt Vereinen dabei die Freiheit, die Satzung weitestgehend individuell zu gestalten. Grundsätzlich muss eine Satzung nach § 57 Absatz 1 des BGB den Zweck, den Namen sowie den Sitz des Vereins definieren. Darüber hinaus muss sie nach § 58 des BGB Bestimmungen zu den Ein- und Austritt

von Mitgliedern, den Mitgliedschaftsbeiträgen, der Bildung eines Vorstandes und der Einberufung einer Mitgliederversammlung enthalten.

2.1.3 Anmeldeprozess bei einem Sportverein

Der Anmeldeprozess bei einem Sportverein bezeichnet den Vorgang, bei dem eine Person als Mitglied in einen Verein eintritt. Jörg Wollny et. Al. [14] differenzieren dabei zwischen zwei Möglichkeiten, um eine Vereinsmitgliedschaft zu erlangen. Zum einen kann die Mitgliedschaft in einem Verein durch die Teilnahme an der Gründungsversammlung erlangt werden. Zum anderen kann eine Person in einen bestehenden Verein beitreten. Dazu gibt das BGB in § 58 vor, dass die Satzung des Vereins Bestimmungen über den Eintritt der Mitglieder enthalten muss. Nach Jörg Wollny et. Al. [14] sollte für die Anmeldung einer Person in einem Verein ein schriftlicher Aufnahmeantrag in der Satzung des Vereins vorgeschrieben werden. Darüber hinaus empfehlen Jörg Wollny et. Al. [14], dass die Anerkennung des Aufnahmeantrags von der Entscheidung eines Vereinsorgans abhängig gemacht werden soll. Dementsprechend kann der Aufnahmeantrag angenommen oder abgelehnt werden.

2.1.4 Sportsponsoring

Der Terminus Sportsponsoring beschreibt die Beziehung und vertragliche Bindung zwischen Sponsor (Sponsoringgeber) und Gesponsortem (Sponsoringnehmer) [WS18]. Sowohl Vereine als auch Sportler können von Sponsoren gesponsort werden. Beide Parteien profitieren dabei meist von der Partnerschaft, sodass Nowak [Now19] diesbezüglich von einem Win-win-Verhältnis spricht. Der Gesponsorte stellt dem Sponsor seine Reichweite und sein Image für kommunikative Aktivitäten bereit. Im Gegenzug stellt der Sponsor dem Gesponsorten finanzielle Mittel, Sachmittel, Dienstleistungen oder Know-how zur Verfügung.

In der Vergangenheit wurde Sportsponsoring überwiegend aus Sicht der Sponsoren betrachtet. Laut Nowak [Now19] stellt Sportsponsoring vor allem eine wichtige Unterstützung für Leistungssportvereine dar. Rund 60 - 70% der Gesamteinnahmen werden durchschnittlich durch Sportsponsoring in professionellen Teamsportligen erwirtschaftet. Für Sportvereine und die Sportler ist es daher wichtig, die Sponsoren zu gewinnen und zu binden. Auf der einen Seite konkurrieren dabei Sportler und Vereine bei der Suche nach einem passenden Sponsor. Auf der anderen Seite konkurrieren die Sponsoren auf der Suche nach einem passenden Sponsoringnehmer. Nowak [Now19] sieht diesbezüglich in der Digitalisierung Potenziale für beide Seiten, um sich Wettbewerbsvorteile zu schaffen.

2.1.5 Sportmerchandising

Sportmerchandising fasst Maßnahmen zusammen, um den Verkauf einer Ware oder eines Lizenzrechts von einer Person oder Gruppe, von Vereinen, Verbänden/Institutionen im Sport an aktuelle oder potenzielle Anhänger zu fördern [Now19]. Dabei wird die Ware auch oftmals als Merchandise oder kurz Merch bezeichnet. Mit Fanartikeln können dabei Einnahmen in Sportvereinen generiert werden und die Käufer können sich besser mit dem jeweiligen Sportobjekt identifizieren. Rohlmann geht davon aus, dass der Einsatz von digitalen Medien und E-Commerce neben stationären touchpoints eine hohe Relevanz für den Erfolgt im Sportmerchandising hat.

2.1.6 Vereinssoftware

Der Begriff Vereinssoftware beschreibt Programme, mit denen die Aufgaben eines Sportvereins bearbeitet werden können. Golinsky untersucht in seinem Buch Moderne Vereinsorganisation die Digitalisierung in Vereinen und gliedert Vereinssoftware auf Grundlage der Anwendungsbereiche in zwei Kategorien [Gol20]:

Die erste Kategorie ist die Finanzsoftware. Innerhalb des Gabler Wirtschaftslexikons wird dieser Terminus durch Dr. Markus Siepermann und Prof. Dr. Richard Lackes als Software für die Informationsbereitstellung, Planung und Steuerung der Finanzen eines Unternehmens definiert [Dr 18]. Darüber hinaus unterstützt eine Finanzsoftware die Ergebnis- und Liquiditätsplanung sowie die Risikoanalyse und Simulationen zur Entscheidungsunterstützung. Golinsky nennt dazu die Bereitstellung von Online-Banking Funktionen und die Verwaltung anhand von Berichten, Kostenstellen und Kategorien als Anwendungsgebiete für Finanzsoftware in Sportvereinen [Gol20].

Als zweite Kategorie führt Golinsky die Vereinsverwaltungssoftware auf. Der Begriff Vereinsverwaltungssoftware bezeichnet primär Programme für die Verwaltung von Mitgliedern, Beiträgen, Kursen, Trainings- und Raumbelegungen. Ebenfalls sind nach Golinsky Programme für die Buchhaltung eines Sportvereins der Kategorie Vereinsverwaltungssoftware zuzuordnen.

2.2 Kennzeichen und Besonderheiten von Sportvereinen

2.2.1 Organisation in Sportvereinen

Um den Verein nach außen zu repräsentieren, zu führen und um Aufgaben zu verteilen, gibt es innerhalb von Vereinen Funktionen, die von Personen eingenommen werden können. Der Gesetzgeber kennt in seinen Bestimmungen zur Vereinsorganisation

den Vereinsvorstand zur Vertretung nach Außen und Geschäftsführung sowie die Mitgliederversammlung als mehrheitsbildendes Beschlussgremium [14]. Dabei lässt er in den BGB Vereinen die Freiheit, die genannten Vereinsorgane in ihrer Funktion zu verändern, auszubauen und um weitere Vereinsorgane zu ergänzen. Allgemein sind die Funktionen der Personen spezifisch auf den jeweiligen Verein individuell.

In seinem Buch Moderne Vereinsorganisation fast Golinsky [Gol20] die typischen Vereinsstandards zusammen. Er beschreibt unter anderem, welche Funktionen und ämter in den meisten Vereinen von Personen eingenommen werden. Grundsätzlich besitzen Sportvereine Sportler, die Mitglieder in dem jeweiligen Verein sind. In seiner Arbeit beschreibt er, dass Vereine in den meisten Fällen eine Mitgliederversammlung und einen Vorstand in die Vereinssatzung aufnehmen und darüber hinaus Hauptämter definieren. Personen, die ein Hauptamt einnehmen, stehen laut ihm in einem vertraglich geregelten Arbeitsverhältnis mit dem Verein und verdienen mit ihrer Tätigkeit im Verein Geld. Neben den Hauptämtern gibt er auch an, dass es in Sportvereinen üblich ist, dass Personen Ehrenämter ausüben. Für Ehrenämter steht soziales und gesellschaftliches Engagement im Vordergrund. Sie bekommen Ihre Arbeitsaufträge von dem Vorstand des Vereins oder von der Abteilungs- oder Gruppenleitung, sodass eine Hierarchie besteht. Darüber hinaus nennt Golinsky die ehrenamtlichen Unterstützer, die Ihre Hilfe freiwillig anbieten.

2.2.2 Interaktionen mit Mitgliedern und externen Interessengruppen

Damit in dieser Arbeit die digitale Kommunikation und Interaktionen von Sportvereinen mit ihren Interessengruppen optimiert werden kann, werden in dieser Sektion die ausschlaggebenden Kommunikations- als auch die Interaktionsarten näher betrachtet. Prinzipiell muss dabei zwischen dem Verhältnis von dem Verein zu seinen internen und externen Interessengruppen differenziert werden [14]. Die Kommunikations- / Interaktionsarten mit Personen, die ein Amt in dem Sportverein ausüben oder ein Mitglied sind, gelten im Rahmen dieser Arbeit als interne Interessengruppen. Das Verhältnis von ihnen zu dem Verein wird von Wollny et. Al. [14] als Innenverhältnis deklariert. Das Außenverhältnis beschreibt er hingegen als das Verhältnis zwischen dem Sportverein und Personen, die dem Verein nicht zugehören. Die Kommunikations- / Interaktionsarten mit diesen Personen wird im Rahmen dieser Arbeit als Kommunikations- / Interaktion mit externen Interessengruppen bezeichnet.

Zu den internen Interessengruppen gehören die in Kapitel 2.2.1 geschilderten aktuellen ämter, Ehrenämter, ehrenamtlichen Unterstützen sowie die Mitglieder. Wie in Kapitel 1.4 definiert, sind die digitalen Kommunikations- / Interaktionskanäle zwischen Vereinen und den internen Interessengruppen bereits gut erforscht und stehen daher nicht

im Fokus dieser Arbeit.

Die in Kapitel 1.4 genannten bestehenden und potenziellen Interessenten an einer Mitgliedschaft oder einem Ehrenamt, Fans sowie Sponsoren gehören zu den externen Interessengruppen und stehen somit im primären Fokus dieser Arbeit. Golinsky [Gol20] gibt in seinem Buch Moderne Vereinsorganisation an, dass das Internet einen entscheidenden Faktor einnimmt bei der Repräsentation des Vereins gegenüber externen Interessengruppen. Er geht davon aus, dass eine Internetseite aktuell die beste Möglichkeit darstellt, um den Verein im Internet zu repräsentieren.

Dazu können Vereine entweder eine Fachfirma beauftragen oder in Eigenarbeit erstellen. Beide Möglichkeiten sind für Vereine zeitaufwändig und / oder kostenintensiv. Eine Alternative zu einer eigenen Vereinsseite bieten Social-Media-Kanäle. Die haben allerdings den Nachteil, dass die externen Interessengruppen auf verschiedene Plattformen verteilt sind und daher nicht gezielt mit ihnen kommuniziert werden kann. Eine weitere Möglichkeit, um den Verein im Internet zu präsentieren und mit den externen Interessengruppen zu interagieren, sieht er derzeit nicht. Dementsprechend können die im Kapitel 1.4 erwähnten Interaktions- und Kommunikationsarten mit externen Interessengruppen nur mit einem entsprechenden Aufwand bedient werden.

2.3 Webanwendungen

Webanwendungen basieren auf Webtechnologien (HTTP, HTML, JavaScript, etc.) bestehen aus Client und Server Komponenten [Roh18]. Sie können den Anwendern im World-Wide-Web zur Verfügung gestellt werden. In vielen Fällen werden Webanwendungen allerdings auch anderweitig zur Verfügung gestellt. Beispielsweise über unternehmenseigene Netzwerke. In der Regel werden Webanwendungen über einen Webbrowser (Chrome, Firefox, Safari, etc.) aufgerufen und es wird der vom Server bereitgestellte HTML-, JavaScript- und CSS-Code interpretiert und dargestellt. Alternativ kann der Quellcode auch mit einer Laufzeitumgebung bereitgestellt werden. Die Laufzeitumgebung oder der Webserver wird dabei als Web Tier bezeichnet. Darüber hinaus kann auch über Skripte oder die Kommandozeile auf die Webanwendung zugegriffen werden. Allgemein wird die Komponente, mit der auf den Webserver zugegriffen wird, daher als User Agent oder als Client bezeichnet. Der Webserver kommuniziert mit dem Client und greift auf Hintergrundsysteme, wie beispielsweise eine Datenbank zu. Die Hintergrundsysteme werden auch als Backend / Data Tier bezeichnet. Client Tier, Web Tier und Data Tier bilden somit die in Abbildung 2.1 dargestellte 3-Tier-Architektur einer Webanwendung. In der Abbildung wurden den Schichten bereits die erörterten Technologien zugeordnet. Grundsätzlich können auch andere Technologien in den jeweiligen Schichten verwendet werden.

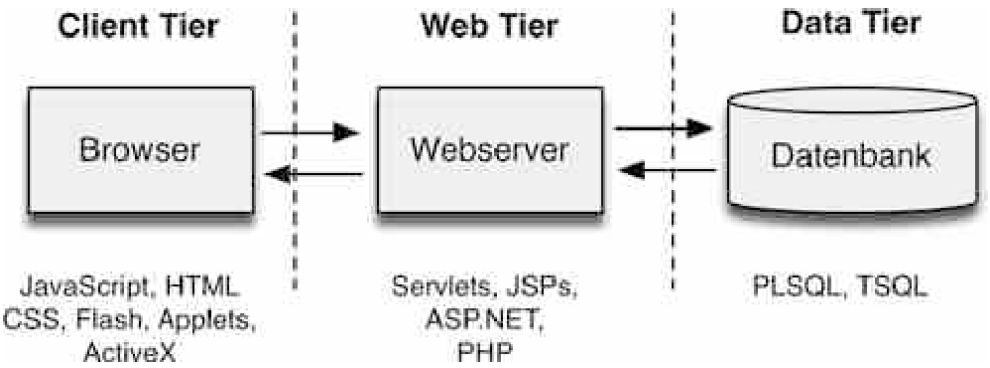

Abbildung 2.1: 3-Tier-Architektur einer Webanwendung
[Roh18]

In modernen Webanwendungen geht der Trend dazu, dass große Teile der Webanwendung dem Client mittels JavaScript übermittelt werden. Mit dem JavaScript-Code ruft der Client im Anschluss eigenständig serverseitige Webdienste auf. Matthias Rohr [Roh18] spricht an dieser Stelle davon, dass die Architektur in modernen Webanwendungen somit differenzierter ausfällt. Daher verschwimmen auch die Grenzen zwischen Front- und Backend.

Der Vorteil von Webanwendungen ist, dass sie für die Nutzung nur einen entsprechenden Client voraussetzen. Somit können die Anwender unabhängig von ihrem Betriebssystem mit der Implementierung einer Webanwendung erreicht werden [ES21]. Für mobile Geräte können Webanwendungen optimiert werden oder sogar in hybride mobile Applikationen eingebunden werden [Sie19].

2.3.1 Integrierte Anwendungssysteme

Integrierte Anwendungssysteme sind Anwendungssysteme, die Teilsysteme auf verschiedenen Ebenen oder Stufen verbinden [HMN19]. Dabei werden Daten, Funktionen, Prozesse sowie Methoden und Programme logisch zusammengeführt. Mertens et. Al. [Mer+17] unterscheiden dabei zwischen den folgenden drei Integrationsarten:

- Unter Datenintegration versteht man die Verwaltung von Daten mehrerer Anwendungssysteme in einem Bestand, um eine redundante Datenhaltung zu vermeiden, sodass eine höhere Datenkonsistenz erreicht werden kann [Pro18a]. Ist eine redundante Datenhaltung unumgänglich, müssen die betroffenen Datenstände über einen periodischen / ereignisgetriebenen Abgleich synchron gehalten werden [Mer+17]. In Bezug auf eine webbasierte Anwendung für Sportvereine gelten diese Kriterien dementsprechend für die verschiedenen Services und Funktionen.

- Die Funktionsintegration wird erfüllt, wenn innerhalb eines Anwendungssystems mehrere Funktionen gebündelt werden [Mer+17]. Nach Lackes et. Al. ist die Datenintegration dafür eine elementare Voraussetzung [Pro18b].
- Prozess- oder Vorgangsintegration setzt voraus, dass aufeinander folgende Funktionalitäten eines Prozesses in einem Anwendungssystem nahtlos miteinander verbunden sind und aufeinander abgestimmt sind. Dafür müssen dem Anwender alle Funktionalitäten und Daten entsprechend zur Verfügung gestellt werden [Mer+17]. Medienbrüche stehen daher mit der Prozess- oder Vorgangsintegration im Widerspruch. Die Prozesse müssen für eine erfolgreiche Prozess- oder Vorgangsintegration umfangreich beschrieben seien. Für den Anmeldeprozess wird dieser Schritt im Kapitel 4 erarbeitet.

2.3.2 Modell View Controller (MVC) Pattern

Modell View Controller (MVC) ist ein Muster (Pattern), welches Software mit grafischer Oberfläche in drei Komponenten unterteilt, sodass die Darstellung von Informationen getrennt von der Veränderung möglich ist [Kle18]. Dadurch können Komponenten verändert werden ohne dass die äußere Darstellung oder die Komponenten zur Veränderung der Informationen angepasst werden müssen. Die Software ist somit leichter erweiterbar und die einzelnen Komponenten können besser wiederverwendet werden.

Die erste Komponente, das Modell, dient der Speicherung von den Informationen. Alle Views sind dem Modell bekannt. Das Modell versorgt sie mit Informationen und es informiert die Views bei Veränderungen. Dazu werden meisten Observer Patterns genutzt [Hoc20]. Darüber hinaus kommuniziert das Modell mit externen Datenquellen, um die Datenintegration zu gewährleisten.

Views sind die zweite Komponente und dienen der Darstellung von den Informationen und von den entsprechenden Veränderungen. Hierbei erfolgt die Darstellung nicht nur auf der grafischen Oberfläche der Software. Die Informationen können zuvor durch andere Klassen aufbereitet werden. Views werden im Modell angegeben, damit sie über die Veränderungen informiert werden. Sie leiten Benutzeraktionen an der grafischen Oberfläche an den Controller weiter.

Der Controller selbst steuert die Anwendung, indem er Benutzeraktionen und weiteren Quellen, wie beispielsweise Sensoren, bereinigt und an ein Modell weiter gibt oder Werte verändert. Dabei kann es mehrere Controller zu einem Modell geben (1 Modell zu n Controller Beziehung). Zum Zweck der Datenintegration können Controller externen Datenquellen kommunizieren.

Kapitel 3

Architekturgestaltung

3.1 Analyse und Konzeption

3.1.1 Stakeholder der Hauptfunktionalitäten (Use Cases)

Die Stakeholder der Hauptfunktionalitäten sind grundsätzlich die Sportvereine sowie ihre internen und externen Interessengruppen. Für die Gestaltung einer Architektur zu Kommunikation sowie Interaktion zwischen Sportvereinen und ihren externen Interessengruppen ist es notwendig, dass die externen Interessengruppen für die Use Cases genau definiert sind. Auf Grundlage der in Kapitel 2.2.2 geschilderten Kategorien wird in Tabelle 3.1 dazu zwischen verschiedenen Unterkategorien differenziert. Allgemein sind interne Interessengruppen ebenfalls natürliche Personen können als Anwender die Funktionalitäten nutzen. Allerdings sind die sie primär für die externen Interessengruppen optimiert.

Stakeholder	Beschreibung
A1	A1 bezeichnet eine Person, die sich für die Mitgliedschaft in einem Sportverein interessiert
A2	A2 bezeichnet eine Person, die sich für ein Amt im Sportverein interessiert
A3	A3 bezeichnet einen Fan des Sportvereins
A4	A4 bezeichnet einen potenziellen Fan des Vereins. Die Person interessiert sich in dem aktuellen Moment für den Sportverein, hat aber kein dauerhaftes Interesse. Der Sportverein hat das Ziel, dass A4 dauerhaft Interesse hat und zu einem beständigen Fan wird.
A5	A5 bezeichnet einen bestehenden Sponsorgeber des Sportvereins

A6	A6 bezeichnet einen potenziellen Sponsorgeber des Sportvereins
A7	A7 bezeichnet den Sportverein an sich. Dabei wird der Sportverein und seine bestehenden (Ehren-) Ämter, die Verwaltungsaufgaben übernehmen, zusammengefasst.
A8	A8 bezeichnet ein aktives Mitglied in einem Sportverein.

Tabelle 3.1: Stakeholder der Hauptfunktionalitäten (Use Cases)

3.1.2 Hauptfunktionalitäten (Use Cases)

In Kapitel 1.3 wurde bereits erwähnt, dass sich diese Arbeit ausschließlich mit Funktionen beschäftigt, die nicht eins zu eins von anderen Branchen übertragen werden können. In diesem Abschnitt werden daher Use Cases definiert, die mit den aktuellen Vereinssoftware angeboten nicht realisiert werden können und daher in dieser Arbeit behandelt werden.

Um Anwendungsfunktionen standardisiert zu definieren, stellt Kleuker in seinem Buch “Grundkurs Software-Engineering mit UML“ eine Schablone für die Dokumentation von Use Cases zur Verfügung [Kle18]. Die Schablone ist tabellarisch strukturiert und gliedert Use Cases in die folgenden Merkmale:

- Name des Use Case (eine kurze Beschreibung des Use Cases)
- Nummer (eindeutige Nummer, die dem Use Case zugeordnet wird)
- Paket (Teilaufgabenbereich des Use Cases. Wird oftmals bei komplexen Systemen angewendet)
- Autor (Ersteller / Bearbeiter des Use Cases)
- Version (Versionsnummer des Use Cases)
- Kurzbeschreibung (kurze Zusammenfassung des Use Cases)
- beteiligte Aktoren / Stakeholder
- Fachverantwortlicher (fachlicher Ansprechpartner des Use Cases)
- Referenzen (auf Gesetze, Normen, bestehende Systeme)
- Vorbedingungen (Zustand der Anwendung, damit der Use Case möglich ist)
- Nachbedingungen (Ergebnis des Use Cases)
- typischer Verlauf (einzelnen Schritte die während des Use Cases durchlaufen werden)

- alternative Abläufe (Alternative Schritte zum typischen Verlauf)
- Kritikalität (Priorität des Use Cases für das Gesamtsystem)
- Verknüpfungen (Zusammenhang zu anderen Use Cases)
- funktionale Anforderungen (Ableitung funktionaler Anforderungen aus dem Use Case)
- nicht-funktionale Anforderungen (Ableitung nicht-funktionaler Anforderungen aus dem Use Case)

Vor der Nutzung der Schablone müssen die Begriffe funktionale sowie nicht-funktionale Anforderungen näher erklärt werden. Funktionale Anforderungen beschreiben Funktionalitäten, Daten oder ein Verhalten des Systems, das für den Use Case benötigt wird [Kle18]. Nicht-funktionale Anforderungen (NFA) beschreiben hingegen Merkmale der Software, die maßgeblich für die Qualität der Funktionalitäten sind. Sie werden separat in dem Kapitel 3.1.3 erarbeitet. Da die folgenden Use Cases alle von dem Autor dieser Arbeit verfasst wurden und zu der Architektur des Prototypen gehören, sind die Merkmale Autor, Paket, Version und Fachverantwortlicher bei allen Use Cases identisch und werden nicht näher erörtert. Alle Use Cases können im Anhang A in tabellarischer Form betrachtet werden.

Insgesamt widmet sich der zweite Schwerpunkt sieben Use Cases. Der Use Case “Vereinsprofil darstellen“ (U1) bildet dabei ein zentrales Element, über das den Stakeholdern A1 - A6 der Zugang zu den restlichen Hauptfunktionen (Use Cases) ermöglicht wird und hat daher die Prioriät 1 (hoch). Hier werden alle Informationen eines Sportvereins gebündelt dargestellt und über Menüpunkte können weitere Details betrachtet werden. Damit der Anwender das Vereinsprofil suchen und betrachten kann, muss der Sportverein vorab ein Profil angelegt haben. Für den Use Case ist es notwendig, dass die webbasierte Anwendung das Anlegen von n verschiedenen Vereinsprofilen inklusive Profil und Hintergrundbild ermöglicht. Für die Suche nach Sportvereinen muss es eine filterbare Vereinsliste geben.

Der Use Case U2 hat den Titel “Vereinsevents darstellen“ und ist ebenfalls ein Menüpunkt von U1 mit mittlerer Priorität (2). Neben den externen Interessengruppen (A1 - A6) treten hier zusätzlich aktive Mitglieder (A8) als Stakeholder auf. Mit den zugehörigen Funktionalitäten sollen Anwender Events von Vereinen in einem Eventkalender einsehen und filtern können. Dazu befindet sich der Anwender auf einem bestehenden Vereinsprofil und klickt auf den Menüpunkt “Events“. Daraufhin bietet die Anwendung dem Anwender einen Eventkalender mit den Termindaten (Datum, Uhrzeit), dem Titel, der Kategorie und der betroffenen Mannschaft / Trainingsgruppe der Events. Der Anwender kann nach den genannten Information filtern. Für den Use Case müssen Events

ebenfalls Entitäten von einem Sportverein sein (1 Verein zu n Events Verhältnis) und die Anwendung muss über eine zugehörige Frontendkomponente mit Filtern verfügen.

Der vierte Use Case U3 trägt den Namen “Sportlerübersicht darstellen“ und ermöglicht es den Stakeholdern A1 - A6 Informationen über die Sportler zu betrachten. Dazu befinden sich der Anwender auf der Mannschafts- / Trainingsgruppenübersicht und klickt auf den Button “Sportler“ bei einer Mannschaft oder Trainingsgruppe. Der Anwender sieht daraufhin die Sportler der Mannschaft / Trainingsgruppe inklusive ihres Namens, Alters und der Position/ Funktion in der Gruppe. Sofern die Sportler in der Vereinssoftware registriert sind, sind die persönlichen Profile mit der Übersicht verknüpft. In diesem Use Case werden die persönlichen Daten im Prototypen uneingeschränkt dargestellt. Bevor die Funktion produktiv geschaltet wird und echte Nutzer die Anwendung bedienen, sollte die Möglichkeit geschaffen für persönliche Profile und dazugehörige Daten die Privatsphären Einstellungen setzen zu können. Diese Funktionalität muss allerdings im Anschluss an den Prototypen in einem separaten Use Case erarbeitet werden. Im Rahmen dieses Use Cases werden grundsätzlich die Bedingungen geschaffen, dass die Informationen überhaupt in einer Sportlerübersicht angezeigt werden können. Dafür müssen Sportler Entitäten von Mannschaften / Trainingsgruppen seien. (1 Mannschaft / Trainingsgruppe zu n Sportler Verhältnis). Auf der Mannschafts- / Trainingsgruppenübersicht muss es darüber hinaus den Button “Sportler“ geben und persönliche Profile müssen mit den Sportlerdaten verknüpfbar seien. Allgemein muss die grafische Oberfläche für die Sportlerübersicht implementiert werden.

Use Case U4 hat den Titel “Vorstandsübersicht / Ämterübersicht darstellen“. Er bietet vorrangig den Stakeholdern A1 - A6 und A8 Auskunft über die Organisation innerhalb des Sportvereins. Dazu befindet sich der Anwender auf einem Vereinsprofil und klickt auf den Menüpunkt “Vorstand / Organisation“. Der Anwender bekommt daraufhin die Namen der Amtsträger und ihre Funktionen angezeigt. Sofern eine Funktion aktuell nicht besetzt ist, wird sie entsprechend hervorgehoben und der Anwender kann auf den Button “Wunschfunktion“ klicken. Der Sportverein bekommt daraufhin eine Meldung, dass sich der Anwender für die Funktion interessiert und kann Kontakt aufnehmen. Damit dieser Use Case möglich ist, müssen Amtsträger und Funktionsinhaber Entitäten eines Sportvereins seien (1 Sportverein zu n Amtsträger verhältnis). Des weiteren muss die Vorstandsübersicht / Ämterübersicht grafisch dargestellt werden und eine Verknüpfung zwischen persönlichen Profilen und Amtsträgern ermöglichen. Wie bei U4 empfiehlt es sich im Anschluss an den Prototypen Privatsphäreneinstellungen zu entwickeln, sodass die Anwendung im Punkt Datenschutz einen guten Stand bieten kann. Neben dem “Wunschfunktion“ Button muss auch eine Benachichtigungsfunktion implementiert werden.

Der Use Case 5 bieten Fans (Stakeholder A3) die Möglichkeit, die Fanartikel des Sportvereins (Stakeholder A7) zu betrachten und über Links zu dem zugehörigen Onlineshop zu wechseln. Dazu befindet sich der Anwender A3 auf dem Vereinsprofil und klickt auf den Menüpunkt “Fanartikel“. Dort sieht er die Fanartikel des Sportvereins und wird bei einem Klick auf einen Artikel auf die zugehörige Onlineshopseite weitergeleitet. Der Use Case hat eine mittlere Priorität (2) und die Informationen sind Teil des in U1 beschriebenen Vereinsprofils. Für die Realisierung des Use Cases müssen Fanartikel eine Entität von Sportvereinen darstellen (1 Verein n Fanartikel Verhältnis). Für die Darstellung der Artikel wird eine grafische Oberfläche benötigt, auf der Bilder zu den Artikeln dargestellt werden können und Links zu den Onlineshops eingebunden werden können.

Neben den Fanartikeln generieren Sportvereine wie in Kapitel 2.1.5 beschrieben mit Sportsponsoring Einnahmen. Der Use Case U6 mit dem Titel “Darstellung Sponsoren“ orientiert sich an der Präsentation von Sponsoren auf der Website des FV Delkenheim [FV]. Die Sponsorgeber der Sportvereine sind die Stakeholder dieses Use Cases. Sie wollen sich und ihre Marke auch in den Profilen der Vereine und Sportler gegenüber den Profilbesuchern präsentieren. Dazu befindet sich der Anwender auf dem persönlichen Profil eines Sportlers oder dem Profil eines Vereins und scrollt in dem Profil nach unten. Dort ist eine Sponsorübersicht dargestellt. Die einzelnen Sponsoren können ihre Firmenlogos und Links zu ihren eigenen Websites hinterlegen. Für diesen Use Case müssen Sponsoringinhalte (Sponsorgeber Logo, Sponsorname, Sponsor Website Link) Entitäten von Sportvereinen darstellen (1 Verein zu n Sponsoren Verhältnis). Für die Sponsoringinhalte muss allgemein eine Sektion in den Profilen eingerichtet werden.

3.1.3 Nicht-Funktionale Anforderungen

Nicht-funktionale Anforderungen sind nach Kleuker [Kle18] Anforderungen, die den Funktionsumfang der Anwendung nicht erweitern. Sie bezeichnen Anforderungen, die die Nutzbarkeit der Anwendung beeinflussen. Strey et. Al. [Luk88] geben an, dass die ISO-Normareihe 25000 einen hilfreichen Katalog zur Herleitung von NFAs bietet. Die ISO-Norm 25010 enthält Qualitätskriterien für Softwaresysteme und Software-Engineering und eignet sich somit sehr gut als Leitfaden für den ersten Schwerpunkt dieser Arbeit [Int11]. Die ISO Norm unterscheidet zwischen äußeren und inneren Qualitätsmerkmalen. Zu den äußeren Merkmalen gehören folgende Merkmale:

1. Funktionale Eignung (engl.: functunial suitability)
2. Effizienz (engl.: performance efficiency)
3. Kompatibilität (engl. compatibility)
4. Benutzbarkeit (engl. usability)

5. Zuverlässigkeit (engl. reliability)
6. Sicherheit (engl. Security)

Alle Merkmale besitzen Kriterien, auf deren Basis ein System bewertet werden kann oder NFA hergeleitet werden können. Das erste Merkmal (Funktionale Eignung) hat die Kriterien Vollständigkeit, Korrektheit und Angemessenheit. In Bezug auf die beschriebenen Use Cases bedeutet dies, dass sich mit der entstehenden Architektur alle Use Cases wie geplant durchführen lassen. Dieses Merkmal lässt sich gut auf diese Arbeit anwenden und ist daher eine wichtige NFA für die Use Cases.

Die Merkmale des zweiten Kriteriums lassen sich im Rahmen dieser Arbeit nur schwer messen. Mit den Kriterien Antwortzeit, Verhalten, Ressourcenverbrauch und Kapazität soll die Effizienz der Anwendung bewertbar werden. Da die Architektur dieses Kapitels im Rahmen der Arbeit allerdings nicht implementiert wird, können die Ergebnisse in zu diesem Merkmal nur schwer bewertet werden. Dieses Merkmal kann dementsprechend nicht als NFA genutzt werden.

Die Kriterien des dritten Merkmals sind Koexistenz und Interoperabilität. Mit ihnen soll bewertet werden, ob die Anwendung schädliche Auswirkungen auf andere Anwendungen hat und wie gut Informationen zwischen den Systemen ausgetauscht und genutzt werden. Die webbasierte Anwendung dieser Arbeit hat mit den Use Cases U6 und U7 Absprünge zu anderen Websites. Dabei findet allerdings kein Datenaustausch zwischen den Systemen statt. Dementsprechend ist dieses Merkmal für diese Arbeit auch nicht relevant.

Die Benutzbarkeit bewertet die Benutzeroberfläche der Anwendung. Da im Rahmen dieses Schwerpunktes keine Benutzeroberfläche gestaltet oder implementiert, gehört die Benutzbarkeit nicht zu den NFA.

Das fünfte äußere Merkmal ist die Zuverlässigkeit. Dabei soll das Kriterium Ausgereiftheit eine Aussage darüber geben, wie stabil die Anwendung bei einer regelmäßigen Nutzung ist. Da die Architektur nicht implementiert wird, lässt auch dieses Kriterium sich schlecht prüfen. Das Kriterium ist für die entstehende Anwendung sehr wichtig. Gerade wenn die Anwendung international genutzt wird, ist es aufgrund der Zeitzonen wichtig, dass die Anwendung 24 Stunden an jedem Wochentag verfügbar ist. Um die Kriterien Fehlertoleranz sowie Wiederherstellbarkeit zu erfüllen ist es grundsätzlich notwendig ein gutes Error Handling in der Anwendung zu implementieren. Error Handling wird in dieser Arbeit vernachlässigt, damit der Funktionsumfang der Architektur möglichst groß ist. Somit sind diese Kriterien keine NFA.

Das letzte äußere Merkmal mit dem Titel Sicherheit hat die Kriterien Vertraulichkeit, Integrität, Nachweisbarkeit, Authentizität und Verantwortlichkeit. Im Fokus dieses Schwerpunktes steht die Gestaltung einer Architektur, die die genannten Use Cases ermöglicht. Im Rahmen dieser Arbeit wird das Merkmal Sicherheit nicht näher betrachtet. Vor Produktivsetzung einer derartigen Anwendung sollte allerdings geprüft werden, dass die Sicherheitskriterien erfüllt werden. Daher resultieren aus dem Merkmal Sicherheit keine NFA für diese Arbeit.

Zu den inneren Merkmalen gehören darüber hinaus folgende Merkmale:

1. Wartbarkeit (engl. Maintainability)
2. Übertragbarkeit (engl. Portability)

Mit den Kriterien Modularität, Wiederverwendbarkeit, Analysierbarkeit, Modifizierbarkeit und Prüfbarkeit soll dabei die Wartbarkeit einer Anwendung bewertet werden. Die Modularität gibt an, wie gut einzelne Komponenten der Software ausgetauscht werden können, ohne dabei Auswirkungen auf andere Komponenten zu haben. Für die Architektur dieses Kapitels ist dieses Kriterium maßgeblich und zählt somit zu den NFA. Das Kriterium Wiederverwendbarkeit gibt darüber hinaus an, wie gut Bestandteile der Anwendung auf andere Systeme übertragen werden können. In Bezug lässt sich daraus die NFA ableiten, dass die Methoden und Objekte von möglichst vielen Komponenten der Anwendung verwendet werden können. Es muss vermieden werden, dass für die Lösung ähnlicher Probleme immer eine neue Lösung umgesetzt werden muss. Um die Architektur auch in Anschluss an diese Arbeit weiterentwickeln zu können, zählt die Modifizierbarkeit auch zu den NFA. Das Kriterium Analysierbarkeit gibt an, wie gut Fehler und der Systemstatus (nach Anpassungen) überwacht werden kann. Das Kriterium Prüfbarkeit gibt des Weiteren an, wie gut die Anwendung getestet werden kann. Für beide Kriterien müssen im Anschluss an diese Arbeit weitere Use Cases erarbeitet werden. Im Rahmen dieser Arbeit zählen diese Merkmale nicht zu den NFA.

Das letzte Merkmal Übertragbarkeit hat die Kriterien Anpassungsfähigkeit, Installierbarkeit, Austauschbarkeit. Die Anpassungsfähigkeit gibt an, wie gut die Anwendung auf unterschiedlicher Hardware, Betriebssystemen oder Nutzungsumgebungen funktioniert. Daraus lässt sich die Anforderung ableiten, dass die Architektur Websiteabrufe mit den gängigen Webbrowsern (Microsoft Edge, Chrome, Firefox, Opera, Apple Safari) ermöglichen muss. Darüber hinaus sollte sie über einen Hostinganbieter wie beispielsweise Ionos betreibbar seien. Das Kriterium Installierbarkeit gibt an, wie gut die Anwendung installierbar / deinstallierbar ist und Austauschbarkeit gibt an, wie gut die Komponenten der Anwendungen ersetzt werden können. Für die entstehende Architektur ist es an dieser Stelle wichtig, dass die Oberfläche ohne Auswirkung auf die Funktionalitäten ausgetauscht werden kann. Aus dem Kriterium Installierbarkeit lässt

sich an dieser Stelle keine NFA ableiten.

3.2 Entwurf

3.2.1 Gesamtarchitektur

Für die Architekturgestaltung wird in diesem Projekt die Softwarearchitektur des Projektes VEMA aus dem Modul “Wahlprojekt“ weiterentwickelt. Die VEMA Webanwendung wurde mit dem Django Framework implementiert und basiert somit auf dem unter Kapitel 2.3.2 definierten Modell-View-Controller Pattern. Django arbeitet allerdings mit der Modell View Template (MVT) Variation [Rav18]. Dabei entspricht das MVT Model dem Model des MVC Patterns und die View Komponenten dem Controller des MVC Pattern. Die Templates nehmen dementsprechend die Rolle der MVC Views ein. Die Abbildung 3.1 stell grafisch dar, wie eine typische Django Webanwendung auf einen Request reagiert.

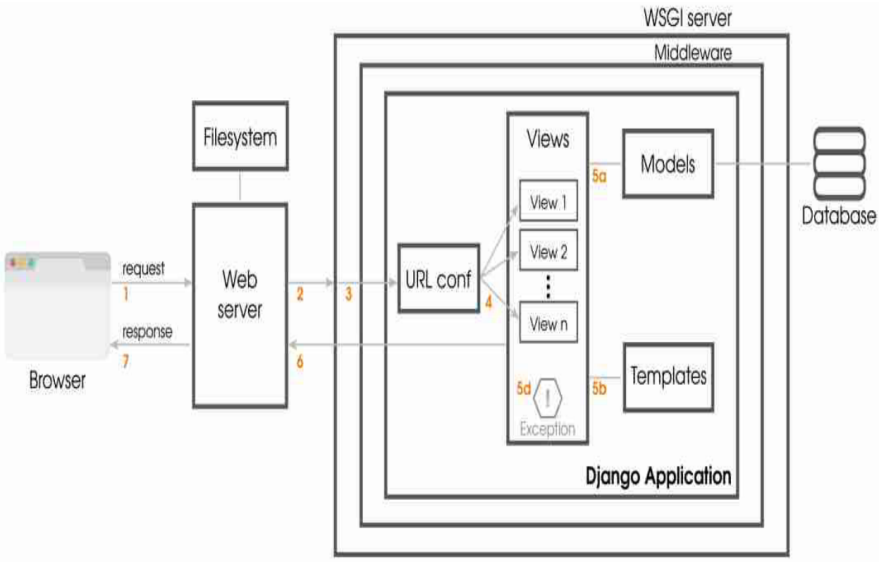

Abbildung 3.1: Komponenten einer Django Application [Rav18]

Zu Beginn löst der Anwender mit einem Webbrowser einen Request an den Webserver aus (In Grafik: 1). Der Webserver übergibt den Request an ein Web Server Gateway Interface (WSGI) (In Grafik: 2). WSGI Bezeichnet eine Schnittstellen-Spezifikation zwischen Webservern und Web Anwendungen / Web Frameworks [Pyt]. Im Anschluss kann der Request an die Django Applikation übergeben werden [Rav18]. In einigen Fällen gibt es allerdings weitere Schnittstellen. Diese sind in der Grafik als Middleware dargestellt und sind für Django Webanwendungen optional (In Grafik: 3). Innerhalb der Django Webanwendung fängt das URL-Konfigurationsmodul den Request ab. Die URL Konfiguration erfolgt bei Django allgemein in der *urls.py* Datei. In den URL-Konfigurationen werden die URL-Pfade mit den jeweiligen Views verknüpft. Dement-

sprechend leiten URL-Konfiguration den Request als *HttpRequest* Objekt an den entsprechenden View weiter (In Grafik: 4). Views enthalten die Geschäftslogik der Anwendung, interagieren mit Datenbanken über Models (In Grafik: 5a) und werfen Exceptions (In Grafik: 5d). Models bezeichnet in dieser Grafik die *models.py* Dateien. Hier werden die Objekte in Form von Klassen definiert. Darüber hinaus rendern Views basierend auf den *Templates* die Antwort auf das *HttpRequest* Objekt (In Grafik: 5b). Alternativ können die Antworten auch auf Basis von einfachen Texten gerendert werden. Die Antwort der Django Applikation wird in Form eines *HttpResponse* Objekts von den Views an den Webserver zurückgegeben (In Grafik: 6). Der Webserver gib die Antwort an den Webbrowser zurück, wo die Inhalte für den Anwender grafisch dargestellt werden (In Grafik: 7).

Die dargestellte Anwendung ist allerdings ein Minimalbeispiel. In der Realität können Django Anwendungen sehr Umfangreich werden. Für eine bessere Strukturierung ermöglichst das Django Framework die Unterteilung des Projekts in mehrere Apps [Gag21]. In der Regel besitzt jede App ein eigene Model / View / Template Komponenten. Je nachdem welche Funktion der Anwendung beansprucht wird, interagiert der Anwender dementsprechend mit verschiedenen Apps der Django web Anwendung.

3.2.2 Django Forms

Neben den MVT Komponenten ermöglicht das Django Framework mit Forms Klassen eine vereinfachte generierung von Web Formularen. *Forms* in ihnen werden die Datenfelder der Webformulare sowie die Datenvalidierung implementiert [Rav18]. Darüber hinaus besitzt die Formklasse von Django die beiden Attribute *is_bound* und *is_valid*. Das Attribut *is_bound* gibt dabei an, ob das jeweilige Formular gefüllt wurde und *is_valid* gibt an, ob die Eingaben den Vorgaben entsprechen. Im Anhang ist mit der Abbildung 3.2 ein Sequenzdiagramm aufbauend auf Ravindran [Rav18] erarbeitet worden.

Darin ist zu erkennen, welche Funktion die einzelnen Komponenten für die Darstellung eines Formulars haben. Die Formklasse enthält die Attribute, die in dem Formular von dem Anwender abgefragt werden sowie eine *is_valid()* Methode, die prüft, ob die eingegebenen Daten valide sind. Bei dem Aufruf des Formulars durch den Anwender geht ein Get-Request bei der View Komponente ein. Die View Komponente rendert das Template mit den in dem Form definierten Datenfeldern, sodass der Anwender sie im Webbrowser betrachten kann. Der Anwender füllt das Formular und versendet einen Post-Request. An dieser Stelle werden sensible Daten über das Formular übermittelt. Daher ist die Get-Methode, die die Parameter in die Uniform Resource Identifier (URI) schreibt, an dieser Stelle ungeeignet. Der View prüft typischerweise mit der is_valide Methode der Form-Klasse, ob die Daten korrekt sind und übermittelt das Ergebnis zurück an den View.

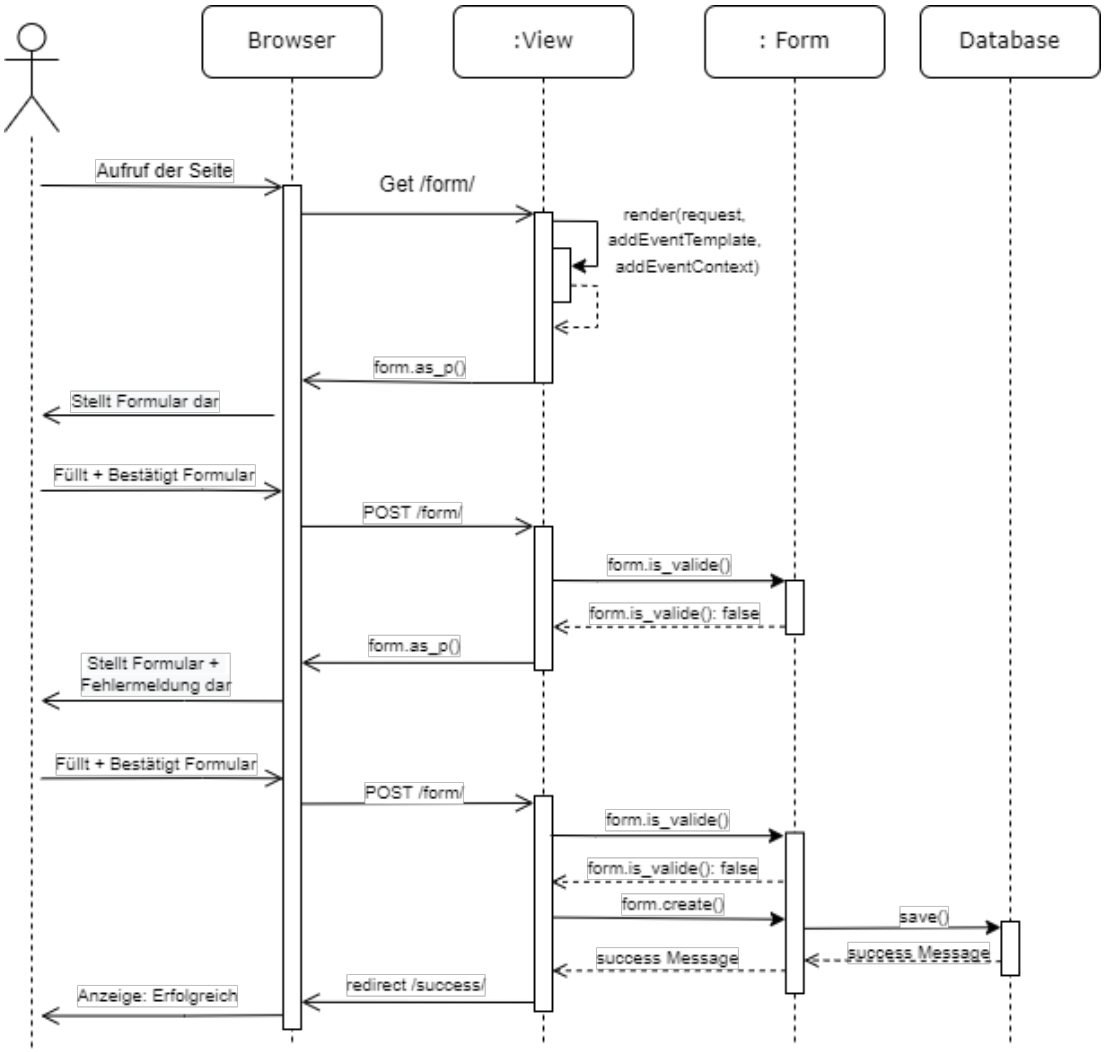

Abbildung 3.2: Typischer Prozess eines Formulars in Django (Sequenzdiagramm)

Sind die Daten nicht valide, übergibt der View typischer Weise wieder das Formular mit einer Fehlermeldung an den Webbrowser und der Anwender kann seine Eingaben korrigieren und erneut versenden. Sind die Daten hingegen valide, gibt die *is_valid()* Methode ein *true* zurück und der View ruft die *create* Methode des Forms auf. Hier werden die Datenfelder des Formulars auf das Model gemapt und im Anschluss in der Datenbank gespeichert. Dafür ruft das Form weitere Create-Methoden des Models auf. Diese sind zur Vereinfachung in diesem Diagramm nicht dargestellt. Ein Beispiel dazu kann allerdings in der Abbildung 3.8 sowie 3.9 betrachtet werden. Die Daten wurden erfolgreich erhoben und dem Anwender wird eine dementsprechende Meldung übermittelt.

In dem Diagramm werden größtenteils asynchrone und synchrone Nachrichten benutzt. Bei synchronen Nachrichten wartet der Sender auf die Antwort des Empfängers

bis er weiter Arbeitet [Kle18]. Bei Asynchronen Nachrichten wartet der Sender nicht auf die Antwort und arbeitet währenddessen weiter. Sofern nicht explizit anders implementiert, kommunizieren Django Views in der Regel synchron [Pat21].

3.2.3 Ableitung der Klassen und Funktionen

In dieser Sektion werden die notwendigen Klassen und Funktionen für die definierten Use Cases hergeleitet. Diese Arbeit baut dabei auf die in Abbildung A.1 dargestellte Softwarearchitektur des Projektes VEMA auf. Die Abbildung wurde bei zur Analyse des Ist-Zustands als ein UML Klassendiagramm erstellt. Das Klassendiagramm wird dabei in Anlehnung an [Dav03] zur Modellierung der Entitäten und ihrer Beziehungen in der Anwendung genutzt. Um die Softwarearchitektur weiterzuentwickeln, wird für jeden Use Case geprüft, welche Anpassungen in der Architektur notwendig sind. Dazu werden die beschriebenen funktionalen sowie die nicht funktionalen Anforderungen als Basis genutzt.

Use Case 1

Der Use Case U1 hat fünf funktionale Anforderungen. Dabei werden die ersten beiden Anforderungen bereits von dem Ergebnis des Projektes VEMA gedeckt. Die Abbildung 3.3 stellt den derzeitigen Ist-Zustand aus dem Projekt VEMA dar:

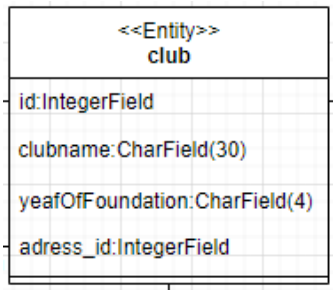

Abbildung 3.3: Ist-Zustand der Vereins-Entität

Dabei besteht wie in Abbildung A.1 dargestellt bereits ein n zu n Verhältnis zwischen den registrierten Anwendern und den Vereinen. Die Entität für Vereine ist dabei als die Klasse *club* im Model der Django App *clubs* implementiert. Für die Anforderungen fünf und sechs müssen weitere Attribute in die Entität *club* aufgenommen werden. Für die Kontaktdaten werden dazu wie in Abbildung 3.4 dargestellt die Attribute *phone* und *email* als *CharField* Objekte mitaufgenommen. Für die Bilder müssen die Attribute *profile_pic* und *background_pic* müssen *ImageField* genutzt werden [Dja].

Die dritte Anforderung kann bereits teilweise erfüllt werden. In dem Projekt VEMA wurde bereits eine Vereinsliste implementiert. Allerdings ist diese Liste nicht filterbar und sobald viele Vereine die Anwendung nutzen, kann der gewünschte Verein nicht mehr in kurzer Zeit gefunden werden. Für die Filterung der müssen zwei Komponenten der

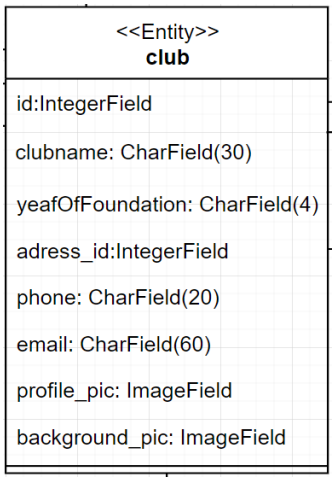

Abbildung 3.4: Soll-Zustand der Vereins-Entität

clubs App angepasst werden: in der Template-Komponente muss ein Eingabefeld für die Vereinssuche ergänzt werden und die View in der View Komponente muss die Logik für die Suche implementiert werden. Die Kommunikation zwischen den Komponenten ist als Aktivitätsdiagramm im Anhang unter A.2 dargestellt. Sofern eine Suche gestartet wird, gibt die Template-Komponente die Suchparameter an die View Komponente. Diese startet eine Suchanfrage über das *club* Model und rendert im Anschluss das Template mit den Suchergebnissen.

Mit den Ergebnissen des Projektes VEMA können Sportvereine bereits angelegt und Daten bearbeitet werden. Die Informationen können aber noch nicht den Interessengruppen mit einem Profil präsentiert werden. Um dies zu ermöglichen, muss ein neues Template erstellt werden und die Viewkomponente angepasst werden, damit das Template gerendert werden kann. Ruft der Anwender ein Vereinsprofil auf, verhält sich die Anwendung wie in Abbildung 3.1 dargestellt. Die Parameter des Requests zur Spezifikation des gewählten Vereins sollen in der URI an die Django Anwendung übergeben werden, sodass jedes Profil eine eigene URI hat.

Use Case 2

Die Entität Events ist in dem Ergebnis des Projekts VEMA noch nicht vorhanden. Daher wird eine neue Entität im Datenmodell der Anwendung benötigt. In der Abbildung 3.5 wurde daher die Entität *club_events* ergänzt. Sie enthält das Feld *id* als Primary Key und steht im 1 (Verein) zu n (Events) Verhältnis mit der Entität *club*.

Neben den Attributen für die Details eines Events besitzt *club_events* ein 1 (Eventkategorie) zu n (Events) Verhältnis zu der Entität *event_categories*. Durch die Erfassung der Kategorien in einer eigenen Entität wird eine redundante Datenhaltung bei mehrmals genutzten Kategorien vermieden. Darüber hinaus können mit weiteren Ausbaustufen Vorschläge in Dropdown-Menüs etc. ermöglicht werden. Für die Events sollte in dem

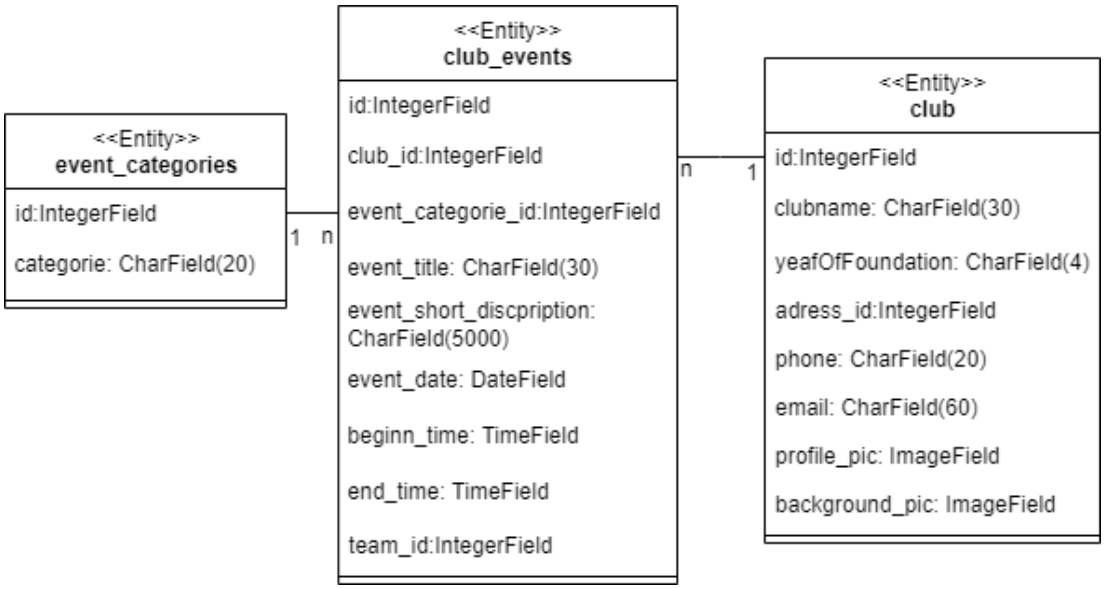

Abbildung 3.5: Entitäten Vereinsevents (ERD)

Django Projekt allgemein eine neue App mit dem Titel “events“ ergänzt werden. Die beiden neuen Entitäten werden dementsprechend als Klassen in der *models.py* Datei der neuen App implementiert. Damit bestehende Kategorien mehrmals genutzt werden können und neue erfasst werden können, muss in der *event_categorie* Klasse eine *get_or_create* Methode programmiert werden. Falls ein Event mit einer bestehenden Kategorie angelegt wird, liefert die Methode das bestehende *event_categories* Objekt. Falls es sich um eine neue Kategorie handelt, erstellt sie ein neues Objekt.

Für die Erfassung neuer Event-Daten eignet sich die Nutzung eines wie in Kapitel 3.2.2 vorgestellten Forms. Neben einem *eventForm* wird wie in der Abbildung 3.2 ein (*addEventTemplate*) und ein View (*addEventView*) benötigt. Neben den geschilderten Funktionen sollte die View Komponente eine Logik zur Prüfung der Berechtigung implementieren, dass nur Anwender mit Adminrechten des Vereins Events erstellen können.

Die Darstellung eine Eventkalenders sowie einer Termindetailansicht erfolgen über neue Templates und neue Views. Ähnlich zu dem Vereinsprofil sollen auch diese beiden Seiten für alle zugänglich seien. Daher eignet sich die Get Methode zur Übermittlung des Requests. Da die Inhalte des *HttpResponse* Objekts sehr umfangreich seien können, eignet sich hier die Post Methode besser zur Übermittlung der Veranstaltungsdaten.

Use Case 3

Für die Realisierung des dritten Use Cases muss die Architektur der web Anwendung vier funktionale Anforderung ermöglichen. Die erste Anforderung ist mit dem aktuellen Stand des Projekts VEMA bereits erfüllt. Die Abbildung 3.6 ist ein Ausschnitt des im Anhang unter A.1 dargestellten Entity Relation Diagramms. Die Entität *team* entspricht

dabei einer Mannschaft oder Trainingsgruppe und die Entität *membership* entspricht der Mitgliedschaft (inkl. Mitgliedsdaten) einer Person in einem Sportverein. Die Entität *team_members* entspricht die Mitgliedschaft in einer Mannschaft oder Trainingsgruppe und repräsentiert dabei Personen als Sportler. Ein Mitglied in einer Trainingsgruppe / Mannschaft besitzt in dem Verein eine Mitgliedschaft und kann in mehreren Trainingsgruppen / Mannschaften des Vereins aktiv seien. Daher besteht zwischen *membership* und *team_members* ein 1 (*membership*) zu n (*team_members*) Verhältnis. Ein Team besteht hingegen aus n Mitgliedern und ein Teammitglied kann auch nur eine Mitgliedschaft in einem Team haben. Zwischen *team* und *team_members* besteht daher ein 1 (*team*) zu n (*team_members*) Verhältnis. Im Frontend der Anwendung ist bereits eine Funktion implementiert, mit der Vereinsmitglieder zu Trainingsgruppen hinzugefügt werden können.

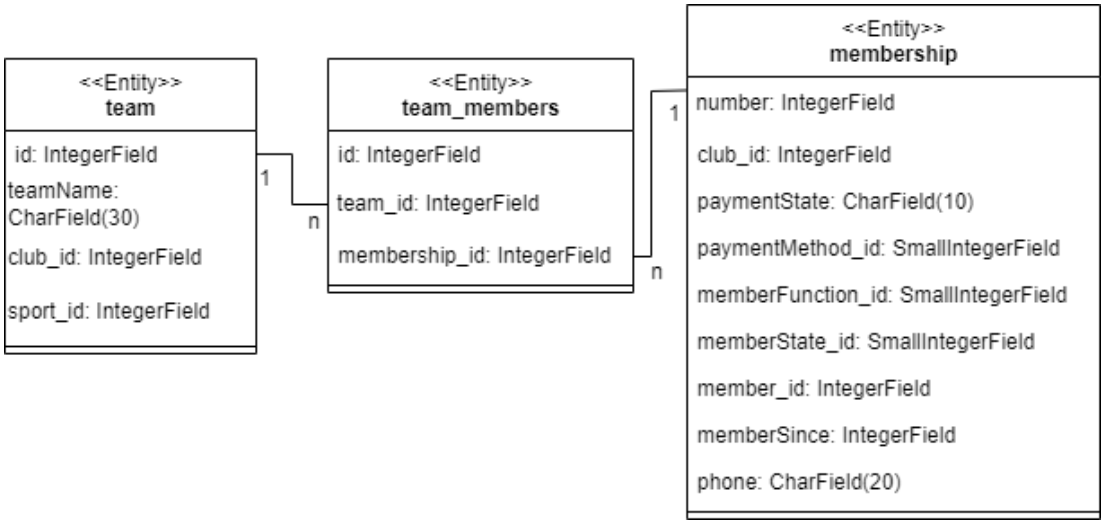

Abbildung 3.6: Ist-Zustand Sportler-Entität in VEMA

Allerdings können mit dieser Architektur keine Rollen / Spielerpositionen dargestellt werden. Die Entitäten müssen daher wie in Abbildung 3.7 dargestellt angepasst werden.

Die Spielerpositionen und sind in Abhängigkeit von der Sportart in verschiedenen Team oftmals ähnlich. Um eine redundante Datenhaltung zu vermeiden, wird in dem Datenmodell die neue Entität *team_position* ergänzt. Eine Rolle / Spielposition kann von mehreren Sportlern eingenommen werden. Innerhalb einer Mannschaft / Trainingsgruppe kann ein Sportler jedoch nur eine Rolle / Spielposition einnehmen. Daher besteht ein eins (*team_positions*) zu n (*team_members*) Verhältnis zwischen den beiden Entitäten. Um eine redundante Datenhaltung zu vermeiden, muss die Klasse *team_positions* bei der Implementierung im Modell, wie bei der Entität *club_events*, eine *get_or_create* Methode besitzen. Die Entität *sport* ist wie im Anhang unter A.1 dargestellt in der Anwendung VEMA bereits implementiert und wird in diesem Zusammenhang mit der *team_positions* verbunden. Es besteht dabei ein eins (*sport*) zu n (*team_positions*) Verhältnis. Eine Rolle / Spielerposition kann nur einer Sportart zugeordnet werden. Bei

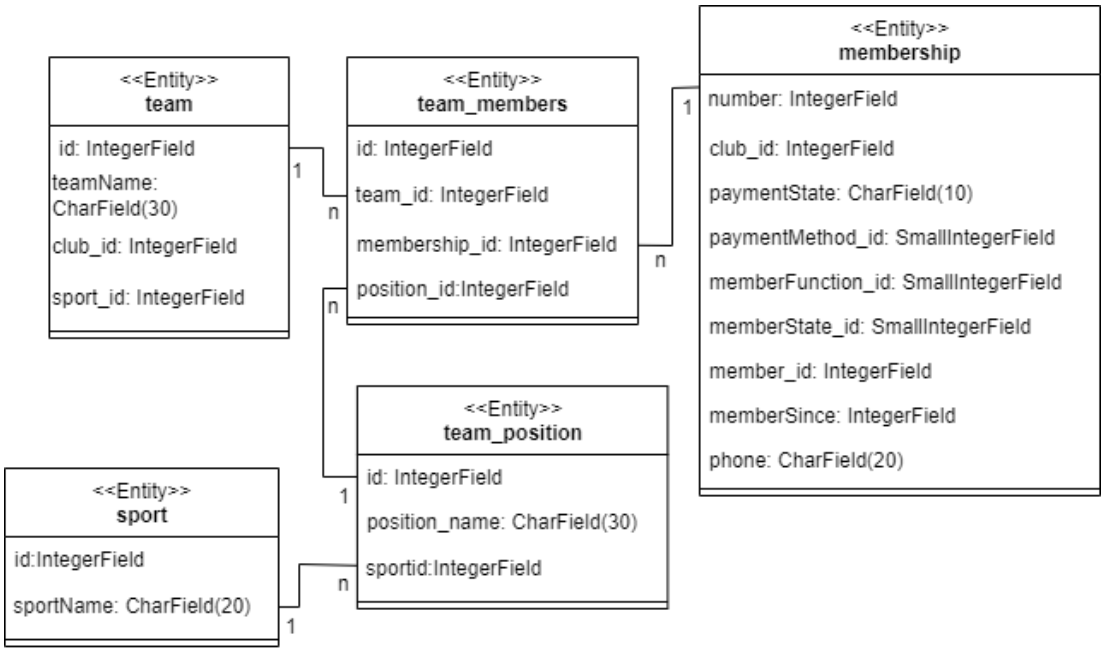

Abbildung 3.7: Soll-Zustand Sportler-Entität in VEMA

einer Sportart kann es jedoch mehrere Rollen / Spielerpositionen geben.

Für die zweite funktionale Anforderung des Use Cases eignet sich ebenfalls das in Kapitel 3.2.2 vorgestellte Muster. Allerdings muss die Architektur an dieser Stelle besonders angepasst werden: Vereine können die Sportlerdaten entweder von einem persönlichen Profil nutzen können oder sollen die Daten manuell erfassen können. Somit können auch Sportler präsentiert werden, die noch nicht in der Anwendung registriert sind. Die persönlichen Daten eines Sportlers bleiben unabhängig von der Mannschaft / Trainingsgruppe gleich. Die persönlichen Daten eines Vereinsmitgliedes werden somit wie in Abbildung 3.7 dargestellt in der *mebership* Entität gespeichert. Die Verknüpfung des persönlichen Profils oder die Erhebung der persönlichen Daten ist daher im Bereich der Mitgliederverwaltung eines Vereins zu implementieren. Das Datenmodell muss dabei folgende Kriterien erfüllen: Ist der Sportler bereits registriert, können die Entitäten verknüpft werden und es erfolgt keine redundante Datenhaltung. Aktualisiert eine Person in ihrem persönlichen Profil die Daten, werden die Änderungen auf den Sportleransichten übernommen. Falls der Sportler jedoch nicht registriert ist, können die Daten manuell eingegeben werden.

In der Abbildungen 3.8 wird dargestellt, wie sich die Anwendung verhält, wenn der Anwender die *membership* Entität mit einem persönlichen Profil verknüpfen will. Nachdem er angegeben hat, dass er ein Profil verknüpfen möchte, rendert der View ihm ein Formular auf dem der Anwender die E-Mail-Adresse des Profilinhabers eingibt und absendet. Der View nimmt den Request an und ruft nach der Validierungsprüfung die Methode *syncProfile()* des Models *membership* auf und übergibt die eingegebene E-Mail-Adresse. Die *syncProfile()* Methode ermittelt die UserID des Sportlers und speichert sie

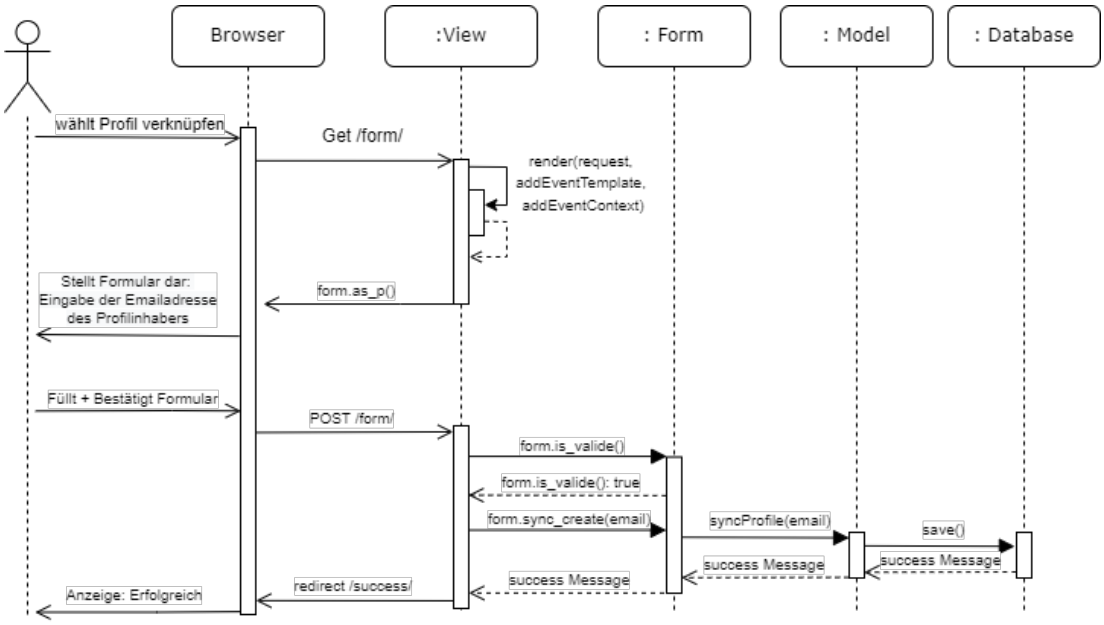

Abbildung 3.8: Verhalten der Anwendung bei Verknüpfung des persönlichen Profils mit der *membership* Entität (Sequenz Diagramm)

in dem Attribut *member_id*. Darüber hinaus werden die in Abbildung 3.7 dargestellten persönlichen Datenfelder auf null gesetzt.

Die Abbildung 3.9 stellt hingegen dar, wie sich die Anwendung verhält, wenn der Anwender die *membership* Entität nicht mit einem persönlichen Profil verknüpfen will. Hier enthält das Formular, das von dem View gerendert wird, Felder für den Vor- sowie Nachnamen und der Geburtstag. Nachdem das Formular korrekt gefüllt und versendet wurde, prüft der View, ob die Daten valide sind. Im Anschluss übergibt der View die Parameter an das Form. Dieses gibt die Parameter in die *noSyncProfile()* Methode des Models weiter. Die *noSyncProfile()* Methode setzt das Attribut *member_id* in der *membership* Entität auf null. Die Felder *first_name*, *last_name* und *birthday* werden mit den eingegebenen Daten gefüllt und gespeichert. Abschließend wird dem Anwender eine Bestätigung für die Datenspeicherung gegeben.

Für die letzte funktionale Anforderung dieses Use Cases muss ein *teamMembersOverviewTemplate* und ein *teamMembersOverviewView* implementiert werden. Der View filtert dazu die Sportlerdaten auf die gewählte Mannschaft / Trainingsgruppe und nutzt das Template, um die Inhalte zu rendern. Je nachdem, ob das Feld *member_id* in der *membership* Entität null ist oder nicht, werden die Daten dazu entweder aus dem persönlichen Profil oder der *membership* Entität genutzt.

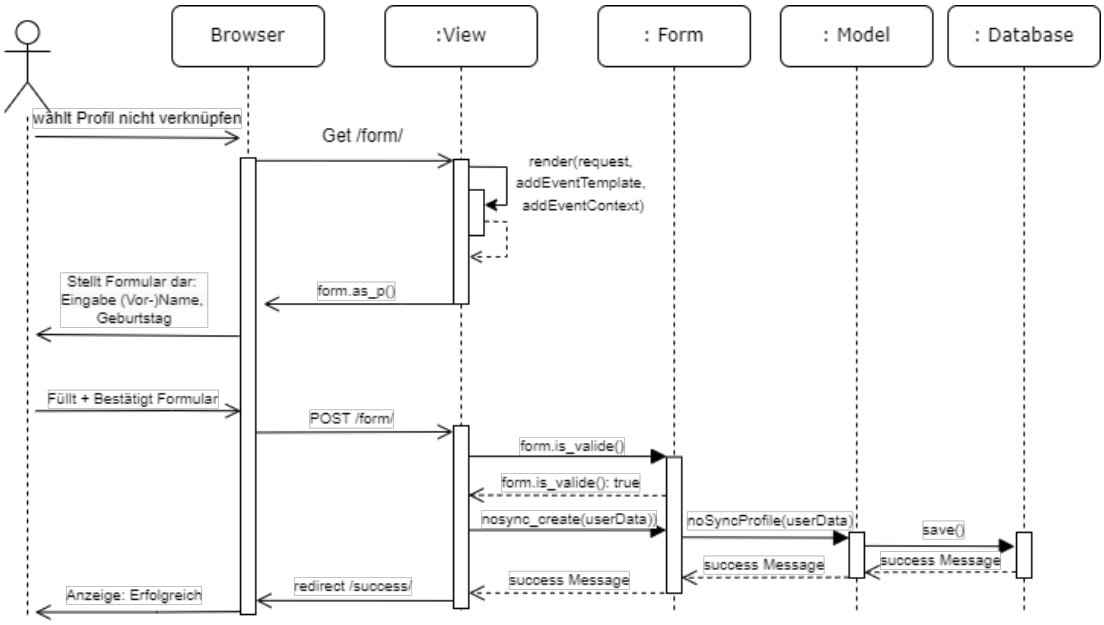

Abbildung 3.9: Verhalten der Anwendung bei manueller Erfassung der persönlichen Daten (Sequenz Diagramm)

Use Case 4

Die zweite und dritte funktionale Anforderung des vierten Use Cases wurden bereits mit der Arbeit an dem dritten Use Case erfüllt. Da die Profilverknüpfung oder Speicherung der persönlichen Mitgliedsdaten über die *membership* Entität erfolgt, sind die Informationen über Vereinsmitglieder, die ein Amt ausüben bereits verfügbar. In den VEMA Projektergebnissen ist bereits die Entität *memberfunction* wie in Abbildung 3.10 implementiert.

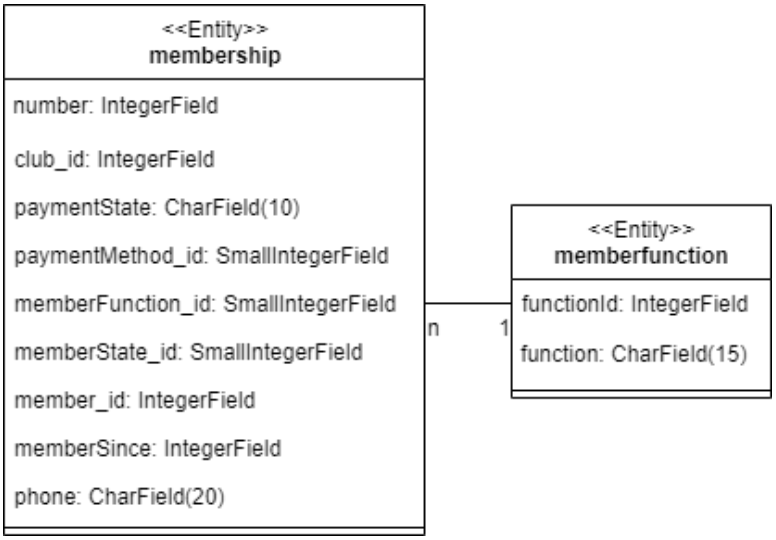

Abbildung 3.10: Ist-Zustand Entitäten für Funktionen oder Ämter in Vereinen (Entity Relation Diagramm)

Mit dieser Implementierung kann jedes Mitglied allerdings nur eine Funktion / Amt ausüben und unbesetzte Funktionen in dem Sportverein können nicht im Datenmodell

repräsentiert werden. Darüber hinaus können Sportvereine in der Satzung festlegen, welche Ämter es in dem Verein gibt [14]. Damit in der Anwendung die Funktionen innerhalb des Vereins repräsentiert werden können und unbesetzte Funktionen und Ämter dargestellt werden können, muss es eine Entität für die verfügbaren Funktionen und Ämter geben. In der Abbildung 3.11 ist diese Entität unter mit der Bezeichnung *functions* abgebildet. In der Abbildung ist die User Entität vereinfacht dargestellt.

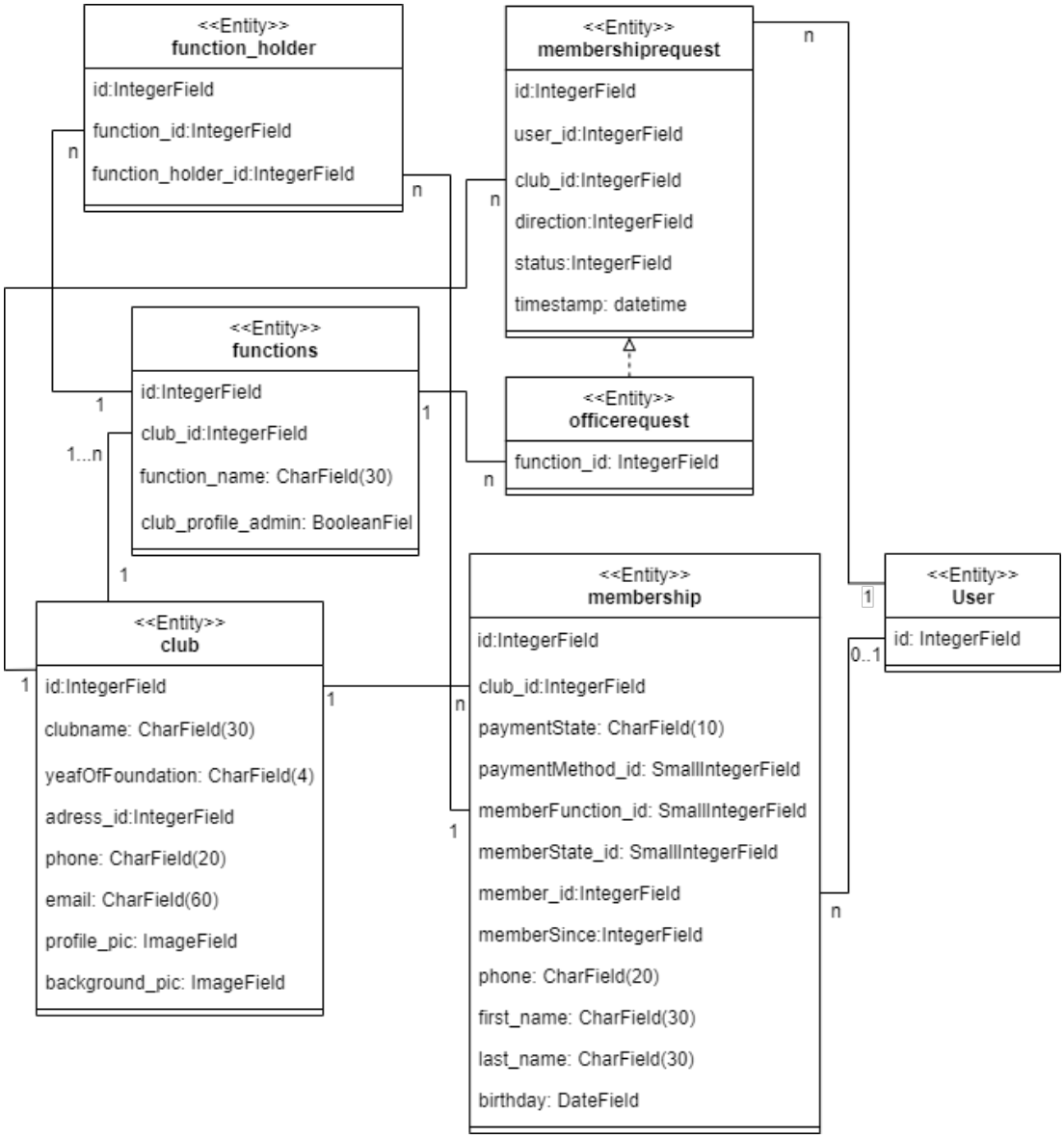

Abbildung 3.11: Soll-Zustand Entitäten für Funktionen oder Ämter in Vereinen (Entity Relation Diagramm)

Da ein Club mehrere Rollen (aber mindestens eine Administrations-Rolle) in seiner Satzung definieren kann und diese Funktion für einen Verein definiert wird, besteht zwischen den Entitäten ein eins (*club*) zu 1 bis n (*functions*) Verhältnis. In der Realität können Vereinsmitglieder mehrere Funktionen in einem Verein einnehmen. Übertragen auf das Datenmodell der Anwendung muss somit ein 1 (*membership*) zu n (*function_holder*) Verhältnis bestehen. Die Entität *function_holder* beschreibt dabei, wer die

Funktion ausübt (Verhältnis zu *membership*) und welche Funktion er ausübt (Verhältnis zu *functions*). Dabei gibt das Attribut *club_profile_admin* an, ob der Funktionsträger Administrationsrechte für das Vereinsprofil hat. Allgemein können mehrere Amtsträger sich ein Amt teilen und die Eigenschaft *function_holder* beschreibt jeweils die Ausübung eines Amts. Daraus resultiert ein eins (*functions*) zu n *function_holder* Verhältnis.

Mit den geschilderten Anpassungen der Softwarearchitektur sind die Informationen für die Darstellung einer Amtsträgerübersicht verfügbar. Für die Darstellung der Übersicht wird ein dazugehöriges Template inklusive Wunschfunktionsbutton benötigt, dass die Informationen in einer Liste darstellen kann und ein View der das Template rendern kann. Da die Übersicht für jeden zugänglich seien soll, muss keine Berechtigungsprüfung im View erfolgen.

Für die letzte funktionale Anforderung ist es notwendig, dass eine Benachichtigungsentität in das Datenmodell mit aufgenommen wird. Die Entität *membershiprequest* ist bereits in VEMA implementiert und ermöglicht es Anwendern Benachichtigungen und Mitgliedschaftsanfragen an einen Verein zu übermitteln. In der Abbildung 3.11 ist diese Entität bereits dargestellt. In dem zugehörigen Modell sind bereits die Funktionen *setStatusAccepted()* und *setStatusDeclined()* implementiert, um den den Status der Mitgliedschaftsanfrage *membershiprequest* bei Annahme oder Ablehnung anzupassen. Das Attribut *direction* gibt an, ob der Verein oder der User die Anfrage erstellt hat. Dabei kann eine Anfrage von einem Anwender erstellt oder empfangen werden und ein Anwender kann n Anfragen erstellen und empfangen (1 Anwender zu n Benachrichtigungen Verhältnis). Dieses Verhältnis besteht auch zwischen Vereinen und den Benachrichtigungen (1 Verein zu n Benachrichtigungen). Damit Anwender eine Wunschfunktion in dem Verein angeben können und eine dazugehörige Benachrichtigung an den Verein senden können, muss die Benachrichtigung allerdings auch die Information zu der gewünschten Funktion enthalten.

An dieser Stelle könnte man eine weitere Entität in das Datenmodell ergänzen. Allerdings müssten somit die Datenfelder und Funktionen redundant implementiert werden. Somit würde das don't repeat yourself (DRY) Prinzip nicht eingehalten werden. Dieses setzt voraus, dass redundanter Quellcode vermieden wird. Um die bestehenden Attribute und Methoden des *membershiprequest* Models wiederzuverwenden und für eine Wunschfunktionsbenachitigung anpassen zu können, eignet sich an dieser Stelle eine Vererbung. Dabei erbt die *officerequest* Entität, wie in Abbildung 3.11 dargestellt, von der *membershiprequest* Entität. Im *officerequest* wird dabei ein Fremdschlüssel zu der *functions* Entität mit einem 1 (*functions*) zu n (*officerequest*) aufgenommen. Dementsprechend erbt das *officerequest* alle Felder und Funktionen von dem *membershiprequest* Modell und ergänzt ein Verhältnis zu, *functions* Modell. In der Abbildung 3.12 ist dazu

dargestellt, wie sich die Django Komponenten verhalten, wenn der Anwender auf den Button “Wunschfunktion“ klickt.

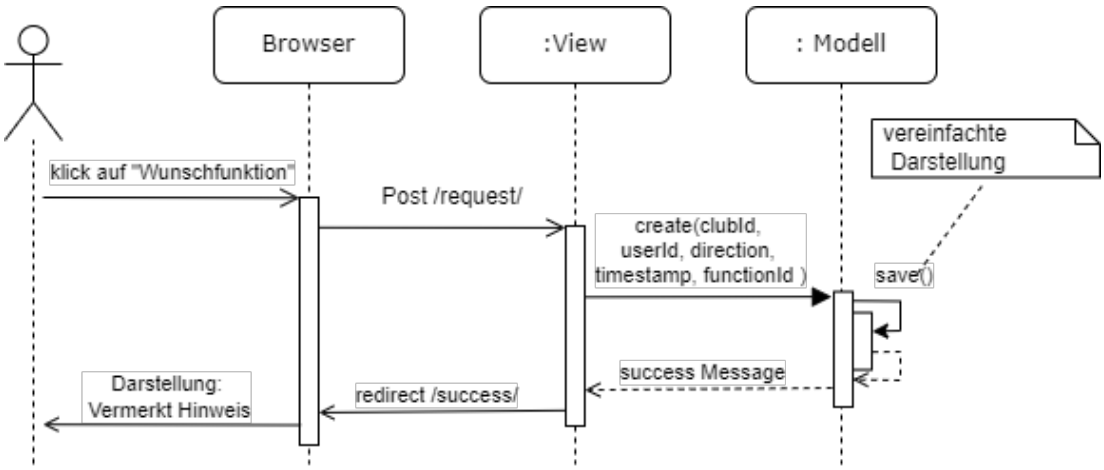

Abbildung 3.12: Speicherung einer Wunschfunktions-Benachrichtigung (Sequenzdiagramm)

Der Webbrowser versendet dabei einen Request. An dieser Stelle werden viele Daten an die Django Anwendung übertragen und die URI soll sich bei dem Anwender nicht verändern. Daher wird an dieser Stelle die Post Methode für den Request genutzt. In Django wird der Request an eine View Methode übergeben. Dieses erstellt ein neues *officerequest* Objekt, indem es die *create()* Methode des *officerequest* Modells aufruft. Nach der erfolgreichen Speicherung geht eine Success-Message vom Modell zum View und wird vom View an den Webbrowser weitergeben. Grafisch könnte dies beispielsweise durch eine inaktivsetzung des Buttons erfolgen. In der Abbildung 3.13 ist dargestellt, wie Vereine die Benachrichtigung einsehen sowie behalten oder verwerfen können:

Im ersten Abschnitt des Sequenzdiagramms ruft ein Vereinsadministrator die bereits im VEMA Projekt implementierte Benachrichtigungsübersicht auf. In der Übersicht werden Mitgliedschaftsanfragen und Wunschfunktionsbenachitigungen dargestellt. Zum einen kann er die Benachrichtigung lesen, indem er sie anklickt. Der Webbrowser sendet im Anschluss einen Request, der der View Komponente der Django Anwendung übergeben wird. Die Django Anwendung ruft die *setStatusAccepted()* Methode des Models auf, sodass der Status der Benachrichtigung auf gelesen geändert wird. Das Modell übergibt eine Success-Message an den View und der View rendert die Benachrichtigung. In der Darstellung der im Webbrowser ist die Benachrichtigung als gelesen markiert. Der Vereinsadministrator sieht in der Benachrichtigung, wer Interesse an welchem Amt hat und kann über das persönliche Profil des Interessenten Kontakt aufnehmen.

Falls der Vereinsadministrator die Benachrichtigung als nicht relevant einstuft, kann er die Benachrichtigung durch Klick auf einen Button löschen. In diesem Fall ruft der View die *setStatusDeclined()* Methode des Models auf. Diese Methode setzt den Status der Benachrichtigung auf verworfen und gibt im Anschluss eine Success-Message zu-

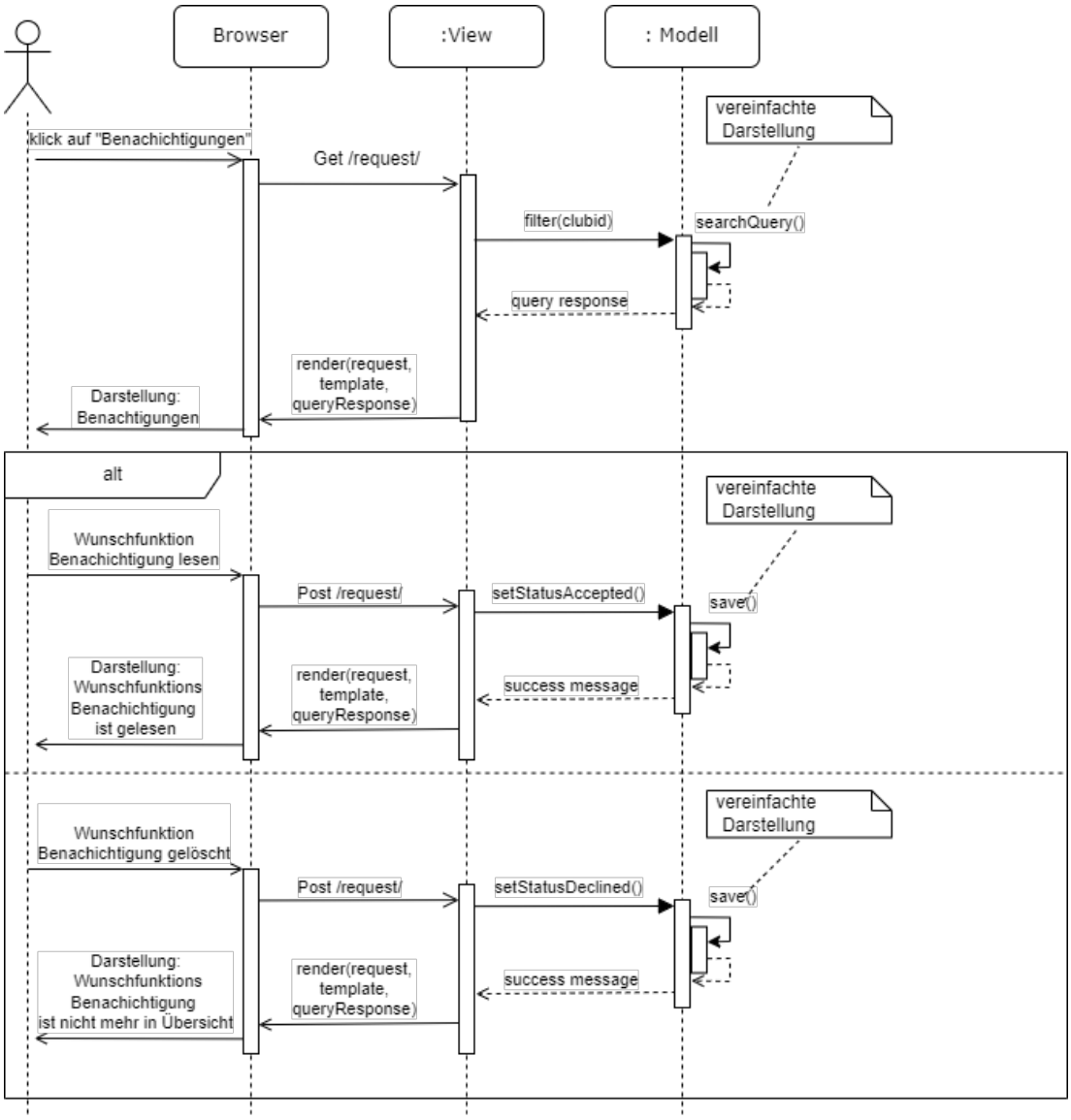

Abbildung 3.13: Interaktion mit einer Wunschfunktions-Benachrichtigung (Sequenzdiagramm)

rück an den View. Dieser rendert die Übersicht neu. Da die Benachrichtigung den Status verworfen hat, ist sie dem in der Antwort an den Webbrowser nicht enthalten und ist daher nicht mehr in der Benachrichtigungsübersicht für den Anwender dargestellt.

Use Case 5

Für den fünften Use Case sind weitere Anpassungen im Datenmodell der VEMA Anwendung notwendig. Dafür wird wie in Abbildung

Kapitel 4

Analyse und Prozessgestaltung

4.1 Analyse und Konzeption

- Anmeldeformulare sind wegen indiv satzung ebenfalls individuell

4.2 Prozessgestaltung

Kapitel 5

Implementierung

5.1 MVP Pattern

5.1.1 Modell / Datenhaltung

5.1.2 View

5.1.3 Controller

5.1.4 Software/ Quellcode

Synchron Asyn-chron kurz erklären

Kapitel 6

Bewertung und Ausblick

6.1 Fazit

6.2 Abgleich funktionale Anforderungen

sh. Schneider.2019

6.3 Nicht-funktionale Anforderungen

sh. Schneider.2019

6.4 Ausblick

- Stellenanzeigen Funktion (Sportler/ Ämter) wäre gut
- Suchfunktion (profile mit filtern)
- Neben Merch Links afiliate Links ermöglichen
- Spendenfunktion
- kann der Prototyp weiter verwendet werden [Kle18] seite 30?
- Vorgehensweise künftig iterativ / inkrementell Prototypen erweitern
- Hauptfunktionalitäten für interne Interessengruppen
- Design Pattern im Code glatt ziehen (Namen der variablen, etc.)
- Sponsoren bekommen eigene Entität

Literatur

[14] *Grundlagen der Vereinspraxis: Jörg Wollny ; Olivia Pauthner. Red.: Johannes Eichelsdörfer ...]* 7., überarb. und aktualisierte Aufl. München: Hanns-Seidel-Stiftung, 2014. ISBN: 9783887954413.

[Ale18] Alexander Otto. „Wie wird mein Sportverein digital: Ein Praxishandbuch zur Digitalisierung des Vereinssports“. In: (2018). URL: https://www.tsg-bergedorf.de/wp-content/uploads/2019/06/Wie_wird_mein_Sportverein_digital.pdf.

[Dav03] Davor Gornik. *Entity Relationship Modeling with UML*. 2003.

[Dja] Django Software Foundation. *Managing files*. Letzter Zugriff: 31.01.2022. URL: https://docs.djangoproject.com/en/4.0/topics/files/.

[Dr 18] Dr. Markus Siepermann, Prof. Dr. Richard Lackes. *Finanzinformationssystem*. Letzter Zugriff: 11.01.2022. 2018. URL: https://wirtschaftslexikon.gabler.de/definition/finanzinformationssystem-34825/version-258318.

[ES21] Timm Eichstädt und Stefan Spieker. *52 Stunden Informatik*. Wiesbaden: Springer Fachmedien Wiesbaden, 2021. ISBN: 978-3-658-33428-4. DOI: 10.1007/978-3-658-33429-1.

[FV] FV Delkenheim e.V. *Internetseite FV Delkenheim e.V.* Letzter Zugriff: 13.01.2022. URL: https://fvdelkenheim.de/.

[Gag21] Valentino Gagliardi. *Decoupled Django*. Berkeley, CA: Apress, 2021. ISBN: 978-1-4842-7143-8. DOI: 10.1007/978-1-4842-7144-5.

[Gol20] Falk Golinsky. *Moderne Vereinsorganisation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020. ISBN: 978-3-662-60526-4. DOI: 10.1007/978-3-662-60527-1.

[HMN19] Hans Robert Hansen, Jan Mendling und Gustaf Neumann. *Wirtschaftsinformatik: Grundlagen und Anwendungen*. 12. völlig neu bearbeitete Auflage. De Gruyter Studium. Berlin: De Gruyter, 2019. ISBN: 9783110608731. DOI: 10.1515/9783110608731.

[Hoc20] Hochschule Augsburg. *Model-View-Controller-Paradigma*. Letzter Zugriff: 11.01.2022. 2020. URL: https://glossar.hs-augsburg.de/Model-View-Controller-Paradigma.

[Int11] International Organization for Standardization. *ISO/IEC 25010: System and software quality models*. 2011.

[Kle18] Stephan Kleuker. *Grundkurs Software-Engineering mit UML*. Wiesbaden: Springer Fachmedien Wiesbaden, 2018. ISBN: 978-3-658-19968-5. DOI: 10.1007/978-3-658-19969-2.

[Lei15] Jan Marco Leimeister. *Einführung in die Wirtschaftsinformatik*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. ISBN: 978-3-540-77846-2. DOI: 10.1007/978-3-540-77847-9.

[Luk88] Lukas Strey, Christian Hein, Tom Ritter, Christian Knauer, Angelika Ganz. „Modellbasierte Methode zur Ableitung nicht-funktionaler Anforderungen im Kontext der Softwaremodernisierung“. In: *Einleitung*. Hrsg. von Klaus Grubmüller und Bernhard Schnell. De Gruyter, 1988, S. 4–40. ISBN: 9783110924992. DOI: 10.1515/9783110924992-003.

[Mer+17] Peter Mertens u. a. *Grundzüge der Wirtschaftsinformatik*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017. ISBN: 978-3-662-53361-1. DOI: 10.1007/978-3-662-53362-8.

[Now19] Gerhard Nowak, Hrsg. *Angewandte Sportökonomie des 21. Jahrhunderts*. Wiesbaden: Springer Fachmedien Wiesbaden, 2019. ISBN: 978-3-658-26967-8. DOI: 10.1007/978-3-658-26968-5.

[Pat21] Patrick Loeber. *Async Views in Django 3.1*. Letzter Zugriff: 03.02.2022. 2021. URL: https://www.python-engineer.com/posts/django-async-views/.

[Pro18a] Prof. Dr. Richard Lackes, Dr. Markus Siepermann. *Datenintegration*. Letzter Zugriff: 11.01.2022. 2018. URL: https://wirtschaftslexikon.gabler.de/definition/datenintegration-31223/version-254785.

[Pro18b] Prof. Dr. Richard Lackes, Dr. Markus Siepermann. *Funktionsintegration*. Letzter Zugriff: 11.01.2022. 2018. URL: https://wirtschaftslexikon.gabler.de/definition/funktionsintegration-34919/version-258410.

[Pyt] Python Software Foundation. *WSGI Utilities and Reference Implementation*. Letzter Zugriff: 30.01.2022. URL: https://docs.python.org/3/library/wsgiref.html.

[Rav18] Arun Ravindran.
Django Design Patterns and Best Practices: Industry-Standard Web Development Techniques and Solutions Using Python, 2nd Edition.
Second edition. Birmingham: Packt Publishing Ltd, 2018.
ISBN: 9781788834971. URL: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=5405693.

[Roh18] Matthias Rohr. *Sicherheit von Webanwendungen in der Praxis*.
Wiesbaden: Springer Fachmedien Wiesbaden, 2018.
ISBN: 978-3-658-20144-9. DOI: 10.1007/978-3-658-20145-6.

[Sie19] Florian Siebler-Guth.
Der Prozess mobiler Entwicklungsprojekte: Muster agiler Methoden.
ISBN: 978-3-658-26730-8.
Wiesbaden: Springer Fachmedien Wiesbaden, 2019.
ISBN: 978-3-658-26730-8. DOI: 10.1007/978-3-658-26731-5.

[Spo16] Sports Business Academy. „SPOAC Sportbusiness Studie 2016“.
In: (2016).

[Spo17] Sports Business Academy. „SPOAC Sportbusiness Studie 2017“.
In: (2017).

[Twi] Twitch. *Twitch Profil: NICKMERCS*. Letzter Zugriff: 13.01.2022.
URL: https://www.twitch.tv/nickmercs/about.

[Vol19] Linda Volkmann. *Sportverein 4.0 – Eine Potenzialanalyse der digitalen Transformation für den Breitensport*. Hrsg. von Gerhard Nowak.
Wiesbaden: Springer Fachmedien Wiesbaden, 2019, S. 241–262.
ISBN: 978-3-658-26967-8. DOI: 10.1007/978-3-658-26968-5.

[WS18] Stefan Walzel und Manfred Schubert. *Sportsponsoring*.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2018.
ISBN: 978-3-662-55245-2. DOI: 10.1007/978-3-662-55246-9.

Anhang A

Use Cases

U1 - Vereinsprofil darstellen

Name des Use Case	Vereinsprofil darstellen
Nummer	U1
Paket	Prototyp
Autor	Maximilian Rosemeier
Version	V1.0
Kurzbeschreibung	Als Anwender möchte ich die Informationen zu einem Sportverein in einem Profil zusammengefasst betrachten können
Stakeholder	A1 / A2 / A3 / A4 / A5 / A6
Fachverantwortlicher	Maximilian Rosemeier
Referenzen	soziale Netzwerke (insbesondere zur Personalgewinnung für Unternehmen wie z.B. XING)
Vorbedingungen	1. Der Sportverein hat ein Profil angelegt 2. Aktor hat die Website der Vereinssoftware aufgerufen
Nachbedingungen	Der Anwender ist auf dem Profil des gewählten Vereins. Dort hat er Zugriff auf weitere Informationen und Funktionen
typischer Ablauf	1. Der Anwender geht auf die Vereinsliste 2. Der Anwender filtert die Vereinsliste 3. Der Anwender hat den Verein, den er betrachten möchte, gefunden und öffnet das Vereinsprofil 4. Auf dem Vereinsprofil sind die in U2-U7 definierten Inhalte zusammengefasst und der Absprung zu den Detailansichten ist möglich
alternative Abläufe	Der Anwender nutzt die Suchleiste
Kritikalität	Priorität: 1 (hoch)

Verknüpfungen	U2-7 können werden über das Vereinsprofil erreicht.
funktionale Anforderungen	1. Vereine sind in der Anwendung eine Entität 2. N Vereine können auf der Website ein Profil einrichten 3. Es gibt eine (nach dem Vereinsnamen) filterbare Liste aller Sportvereine 4. Die Inhalte eines Vereinsprofils können für Besucher der Website dargestellt werden 5. Vereine können ein Profilbild und Hintergrundbild hinterlegen 6. Vereine können Kontaktdaten hinterlegen

U2 - Vereinsevents darstellen

Name des Use Case	Vereinsevents darstellen
Nummer	U2
Paket	Prototyp
Autor	Maximilian Rosemeier
Version	V1.0
Kurzbeschreibung	Als Anwender habe ich Interesse an aktuellen Vereinsevents (Wettkämpfe, Spendenveranstaltungen, etc.). Ich möchte in einer Übersicht sehen, welche Events in der Zukunft geplant sind.
Stakeholder	A1 / A2 / A3 / A4 / A5 / A6 / A8
Fachverantwortlicher	Maximilian Rosemeier
Referenzen	soziale Netzwerke (insbesondere zur Personalgewinnung für Unternehmen wie z.B. XING)
Vorbedingungen	1. Der Sportverein hat ein Profil angelegt 2. Der Anwender hat die Website der Vereinssoftware aufgerufen 3. Der Anwender hat das Profil des Vereins aufgerufen
Nachbedingungen	Der Anwender ist über die künftigen Events des Vereins informiert

typischer Ablauf	1. Der Anwender wechselt in dem Profil zu dem Menüpunkt “Events“ 2. Der Anwender sieht die Vereinsevents in einem Kalender. 3. Der Anwender klickt auf einen Termin im Kalender. Er erhält folgende Details: - Termindaten des Events (Datum, Uhrzeit) - Titel des Events - Kategorie des Events - betroffene Mannschaft / Trainingsgruppe
alternative Abläufe	
Kritikalität	Priorität: 2 (Mittel)
Verknüpfungen	Gehört zu U1 - Vereinsprofil Ansicht
funktionale Anforderungen	1. Events stellen Entitäten von Sportvereinen dar. (1 Verein zu N Events) 2. Vereine können neue Termine erfassen. 3. Die Anwendung bietet die Darstellung eines Eventkalenders 4. Die Anwendung bietet die Detailansicht eines Termins

U3 - Sportlerübersicht darstellen

Name des Use Case	Sportlerübersicht darstellen
Nummer	U3
Paket	Prototyp
Autor	Maximilian Rosemeier
Version	V1.0
Kurzbeschreibung	Als Anwender möchte ich mir einen Überblick über die Sportler des Vereins verschaffen. Bei Sportlern die mich besonders interessieren möchte ich weitere Details einsehen können.
Stakeholder	A1 / A2 / A3 / A4 / A5 / A6
Fachverantwortlicher	Maximilian Rosemeier
Referenzen	
Vorbedingungen	1. Der Sportverein hat ein Profil angelegt 2. Der Anwender hat die Website der Vereinssoftware aufgerufen 3. Der Anwender hat das Profil des Vereins aufgerufen 4. Der Anwender hat die Mannschafts- / Trainingsgruppen Übersicht aufgerufen

Nachbedingungen	Der Anwender weiß in welchen Mannschaften / Trainingsgruppen welche Personen sind und welche Eigenschaften sie haben
typischer Ablauf	1. Der Anwender öffnet die Detailansicht einer Trainingsgruppe / Mannschaft 2. Der Anwender sieht die entsprechenden Mitglieder in einer Liste. Ein Datensatz enthält folgende Merkmale: - Name - Alter - Position / Funktion in der Mannschaft / Trainingsgruppe
alternative Abläufe	Der Anwender klickt auf den Namen des Sportlers. Sofern der Sportler in der Vereinssoftware registriert ist, gelangt er auf sein Profil
Kritikalität	Priorität: 2 (Mittel)
Verknüpfungen	Gehört zu U1 - Vereinsprofil Ansicht und U2 - Mannschaften / Trainingsgruppen
funktionale Anforderungen	1. Sportler stellen Entitäten von Mannschaften/ Trainingsgruppen dar. (1 Verein zu N Sportler) 2. Sportlerdaten können erfasst werden 3. Sportler können mit Profilen von Personen verknüpft werden. Dabei werden die Daten aus dem persönlichen Profil in die Sportler Entität übernommen 4. Die Anwendung bietet die Darstellung einer Sportlerübersicht

U4 - Vorstandsübersicht / Ämterübersicht darstellen

Name des Use Case	Vorstandsübersicht / Ämterübersicht darstellen
Nummer	U4
Paket	Prototyp
Autor	Maximilian Rosemeier
Version	V1.0
Kurzbeschreibung	Als Anwender möchte ich mir einen Überblick über die Organisation des Vereins verschaffen. Dabei interessiert mich vor allem, welche Ämter der Verein hat und wer sie ausübt. Falls Ämter aktuell nicht besetzt sind, möchte ich wissen für welche Ämter der Verein noch Personen sucht.
Stakeholder	A1-A8
Fachverantwortlicher	Maximilian Rosemeier

Referenzen	
Vorbedingungen	1. Der Sportverein hat ein Profil angelegt 2. Der Anwender hat die Website der Vereinssoftware aufgerufen 3. Der Anwender hat das Profil des Vereins aufgerufen
Nachbedingungen	Der Anwender ist auf der Vorstandsübersicht /Ämterübersicht
typischer Ablauf	1. Der Anwender wechselt in dem Profil zu dem Menüpunkt “Vorstand / Organisation“
alternative Abläufe	1. Der Anwender klickt auf den Namen des Vorstandsmitgliedes / Amtsträgers. Sofern die Person in der Vereinssoftware registriert ist, gelangt er auf das zugehörige Profil 2.1 Ein Amt ist derzeit unbesetzt. Es wird dementsprechend dargestellt und der Anwender kann auf den Button “Wunschfunktion“ klicken, um sein Interesse an dem Amt auszudrücken. 2.2 Der Verein bekommt eine Benachrichtigung und kann mit dem Anwender interagieren
Kritikalität	Priorität: 2 (Mittel)
Verknüpfungen	Gehört zu U1 - Vereinsprofil Ansicht
funktionale Anforderungen	1. Ämter stellen Entitäten von Sportvereinen dar. (1 Verein zu N Amtsträger) 2. Ämter können mit Profilen von Personen verknüpft werden 3. Persönliche Daten der Amtsträger können von unregistrierten Personen erfasst werden 4. Die Anwendung bietet die Darstellung einer Amtsträgerübersicht 5. Der Button “Wunschfunktion“ löst eine Benachrichtigung bei dem Verein aus

U5 - Darstellung Fanartikel Links

Name des Use Case	Darstellung Fanartikel Links
Nummer	U5
Paket	Prototyp
Autor	Maximilian Rosemeier
Version	V1.0

Kurzbeschreibung	Als Fan eines Sportvereins möchte ich mich mit dem Verein identifizieren und meine Fanbeziehung zu dem Verein gegenüber dritten präsentieren. Ich möchte daher wissen, welche Fanartikel ein Verein anbietet und wo ich sie erwerben kann.
Stakeholder	A3 und A7
Fachverantwortlicher	Maximilian Rosemeier
Referenzen	Twitch Profil NICKMERCS [Twi]
Vorbedingungen	1. Der Sportverein hat ein Profil angelegt 2. Der Anwender hat die Website der Vereinssoftware aufgerufen 3. Der Anwender hat das Profil des Vereins aufgerufen
Nachbedingungen	Der Anwender ist auf der Merch-Übersicht.
typischer Ablauf	1. Der Anwender wechselt in dem Profil zu dem Menüpunkt “Fanartikel“
alternative Abläufe	Der Anwender klick auf einen Artikel und wird zu einem Merch Shop weitergeleitet.
Kritikalität	Priorität: 2 (Mittel)
Verknüpfungen	Gehört zu U1 - Vereinsprofil Ansicht
funktionale Anforderungen	1. Merch Links stellen Entitäten von Sportvereinen dar. (1 Verein zu N Links) 2. Merch Links ermöglichen den Absprung zu externen Websites 3. Merch Links können mit Bildern und Informationen dargestellt werden 4. Die Anwendung bietet die Darstellung einer Fanartikel-Übersicht

U6 - Darstellung Sponsoren

Name des Use Case	Darstellung Sponsoren
Nummer	U6
Paket	Prototyp
Autor	Maximilian Rosemeier
Version	V1.0

Kurzbeschreibung	Als Sponsorgeber möchte ich, dass ich und meine Marke über die Kanäle des Sponsoringnehmers dargestellt werden. Dabei erhoffe ich mir, dass meine Marke und ich bekannter werden. In der Vereinssoftware stellen sich Vereine und Sportler in ihren Profilen gegenüber dritten vor und ich erwarte, dass ich auf diesen Seiten mitpräsentiert werde.
Stakeholder	A5
Fachverantwortlicher	Maximilian Rosemeier
Referenzen	Darstellung der Sponsoren auf der Website des FV Delkenheim
Vorbedingungen	1. Der Sponsornehmer hat ein Profil angelegt 2. Der Anwender hat die Website der Vereinssoftware aufgerufen 3. Der Anwender hat das Profil des Vereins aufgerufen
Nachbedingungen	Der Anwender kennt die Sponsoren des Profilinhabers
typischer Ablauf	1. Der Anwender scrollt im Profil nach unten. Dort ist eine Sektion mit der Überschrift “Sponsoren“. Die Sponsoren werden in hier einer Übersicht dargestellt.
alternative Abläufe	Der Anwender klick auf einen Sponsor und wird zu der Website des Sponsors weitergeleitet.
Kritikalität	Priorität: 2 (Mittel)
Verknüpfungen	Gehört zu U1 - Vereinsprofil Ansicht
funktionale Anforderungen	1. Sponsoringinhalte stellen Entitäten von Sportvereinen und Personen dar. (n Sponsornehmer zu n Sponsorgebern) 2. Sponsoringinhalte ermöglichen den Absprung zu externen Websites 3. Sponsoringinfhalte können mit Bildern und Informationen dargestellt werden 4. Die Anwendung bietet die Darstellung einer Sponsoren-Übersicht

Entity Relation Diagramm: Ist-Zustand der Anwendung

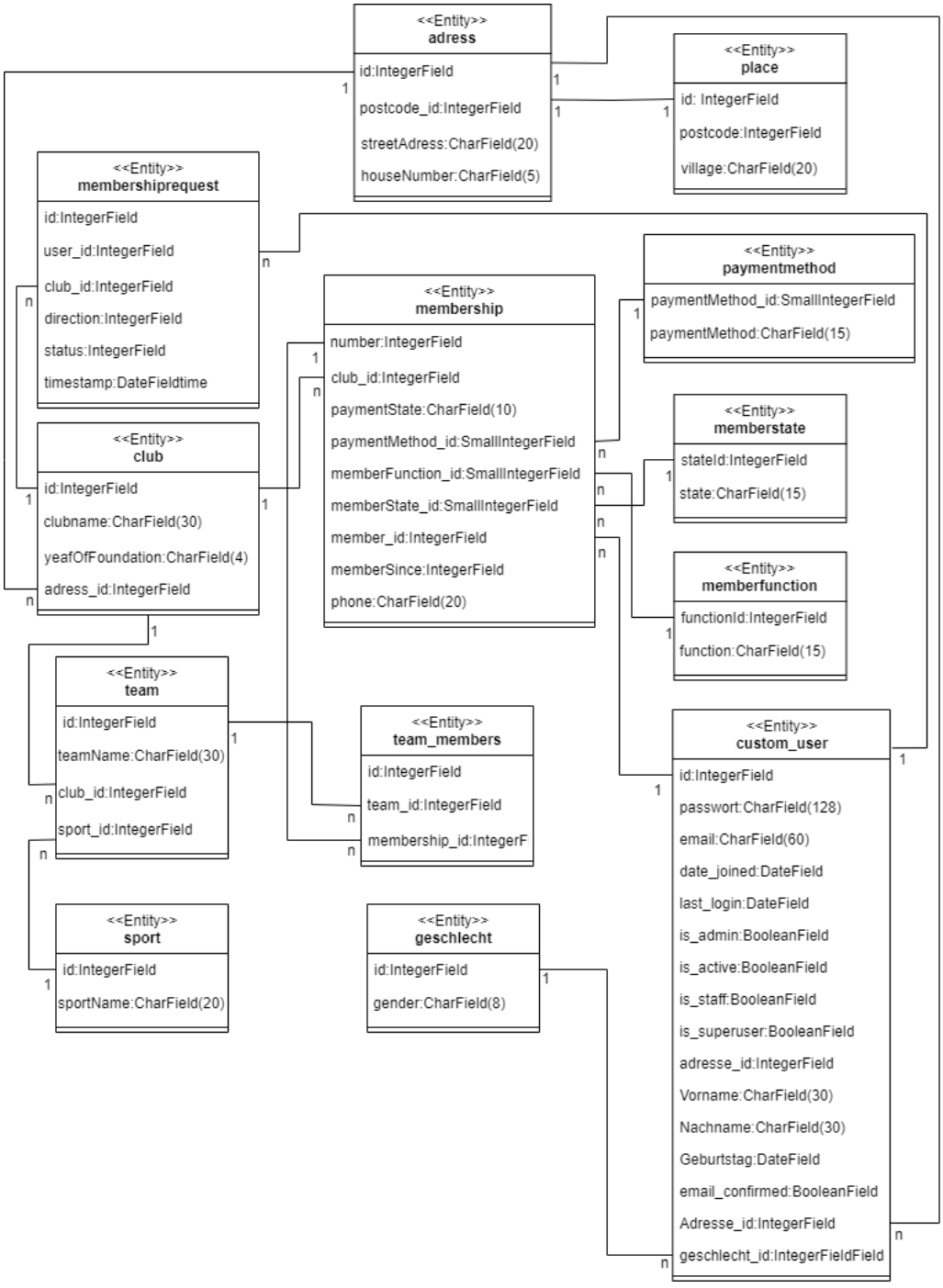

Abbildung A.1: Ist-Zustand der Anwendung (Entity Relation Diagramm)

Aktivitätsdiagramm: Suche für Sportvereine

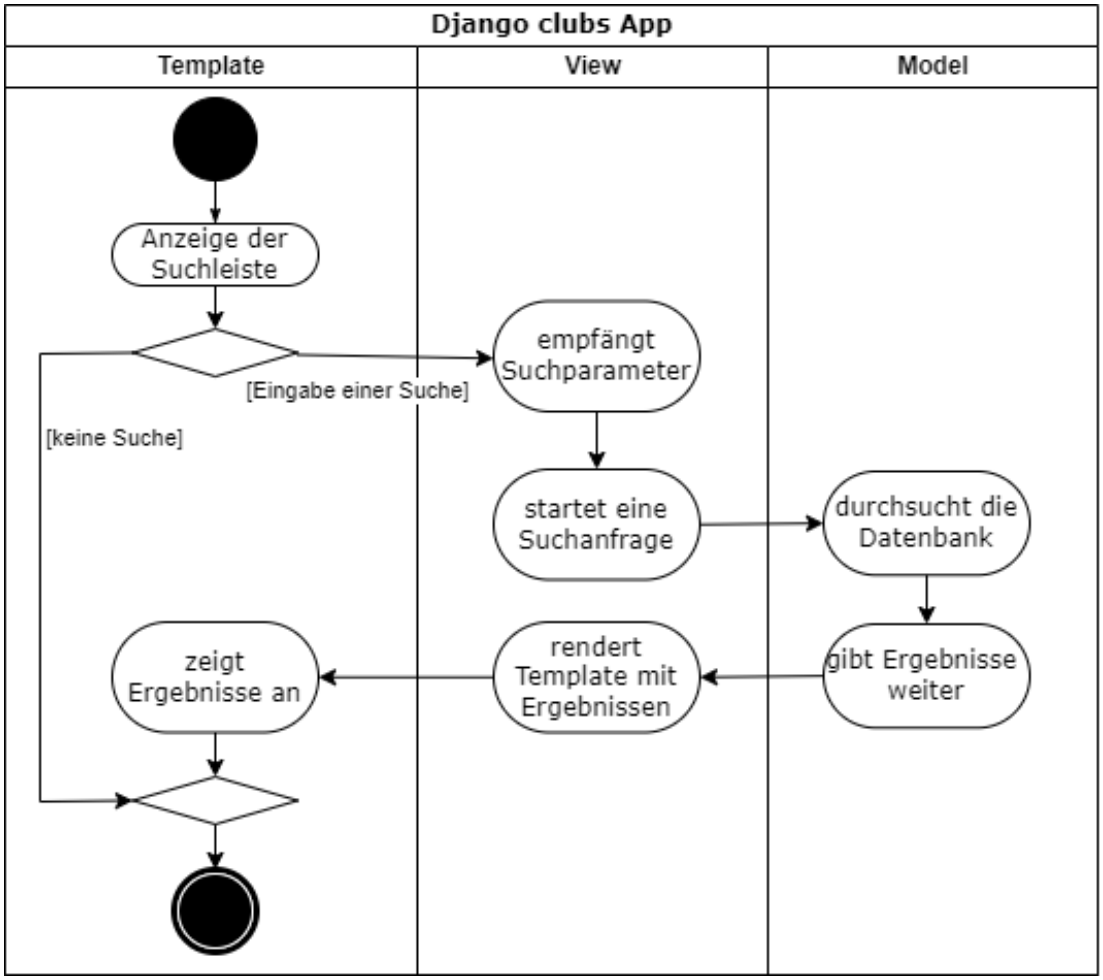

Abbildung A.2: Suche für Sportvereine (Aktivitätsdiagramm)