HAW HAMBURG

BACHELORTHESIS
Paul Ubbo Nordholt

Prozessautomatisierung im Kontext einer Sharing-Plattform für pflegende Angehörige

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Paul Ubbo Nordholt

Prozessautomatisierung im Kontext einer Sharing-Plattform für pflegende Angehörige

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Stefan Sarstedt
Zweitgutachter: Prof. Dr. Susanne Busch

Eingereicht am: 2. Februar 2022

Paul Ubbo Nordholt

Thema der Arbeit

Prozessautomatisierung im Kontext einer Sharing-Plattform für pflegende Angehörige

Stichworte

Prozessautomatisierung, Matching, Pflege, Python, SQLAlchemy, PyQT5

Kurzzusammenfassung

Mit dem Ziel der Automatisierung der Geschäftsprozesse im Betrieb der Online-Tauschbörse AniTa-Familie wurde im Rahmen dieser Arbeit das AniTa-Tool entwickelt. Hierzu wurden bestehende Prozesse und Rahmenbedingungen sowie Anforderungen an das System analysiert. Basierend auf den Analysen wurde eine Software-Architektur entworfen, diese implementiert und evaluiert. Während zwar keine vollständige Automatisierung erzielt werden konnte, ist davon auszugehen, dass der Einsatz des AniTa-Tools die Effizienz und Ergebnisqualität der Geschäftsprozesse steigern wird.

Title of Thesis

Process Automation in the Context of a Sharing Platform for family caregivers

Keywords

Process Automation, Matching, Nursing, Care, Python, SQLAlchemy, PyQT5

Abstract

To automate the business processes in the operation of the online swap meet AniTa-Familie, the AniTa-Tool was developed within the scope of this work. For this purpose, existing processes and business conditions as well as requirements for the system were analysed. Based on these analyses a software architecture was designed, implemented and evaluated. While full automation could not be achieved, the use of the AniTa tool should increase the efficiency and result quality of the business processes.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

User Stories

Abkürzungsverzeichnis

CA	Clean Architecture
CG	CareGiver
CT	CareTaker
DC	Distance-Caregiving
DIP	Dependency Inversion Principle
GUI	Graphical User Interface
HAW	Hochschule für Angewandte Wissenschaften Hamburg
IKT	Informations- und Kommunikationstechnologie
ITSC	Informationstechnik Service Center
LDC	Long-Distance-Caregiver
LDR	Long-Distance-Relative
MP	Multi-Sided Platform

MS	Microsoft
MW	MatchedWith
ORM	Object Relational Mapper
RO	RelativeOf
SoC	Separation of Concerns
SP	Sharing-Platform

Glossar

AniTa	AniTa steht für Angehörige im Tausch und ist das Forschungs- und Entwicklungsprojekt, in dem eine Tauschbörse für Long Distance Caregiver betrieben und das AniTa-Tool entwickelt wurde.
AniTa-Tool	Desktopanwendung zur Prozessautomatisierung im Projekt AniTa.
Bipartiter Graph	Ein einfacher Graph, dessen Knotenmenge sich in zwei disjunkte Teilmengen aufteilen lässt, sodass zwischen den Knoten innerhalb dieser Teilmengen keine Kanten verlaufen.
CareGiver	Repräsentation der Long Distance Caregiver im Datenmodell des AniTa-Tools.
CareTaker	Repräsentation der Long Distance Relatives im Datenmodell des AniTa-Tools.
Gegenleistung	Ein Long Distance Caregiver erhält eine Gegenleistung, wenn er:sie Unterstützung durch eine:n andere:n Long Distance Caregiver erhält und selbst bereits in Vorleistung gegangen ist.

Long Distance Caregiver	Personen, die räumlich weit entfernt lebende Angehörige pflegen.
Long Distance Relative	Personen, die von räumlich weit entfernt lebenden Angehörigen gepflegt werden.
MatchedWith	Repräsentation eines Matches (s. Matching) zwischen CareGiver und CareTaker im Datenmodell des AniTa-Tools.
Matching	Auffinden von LDR-LDC-Paaren deren Wohnorte in räumlicher Nähe zueinander sind, mit dem Ziel der Anbahnung eines Austauschs von Unterstützungs- und Pflegeleistungen.
Multi-Sided Platform	Multi-Sided Platforms sind ein Geschäftsmodell, bei dem die Wertschöpfung primär dadurch stattfindet, dass eine direkte Interaktion zwischen mindestens zwei Parteien ermöglicht wird.
RelativeOf	Repräsentation einer Verwandschaftsbeziehung zwischen CareGiver und CareTaker im Datenmodell des AniTa-Tools.
Spatial Multi-Sided Market	Besondere Form der Multi-Sided Platforms, bei denen sich die Marktteilnehmer räumlich treffen müssen bzw. wollen.
Vorleistung	Ein Long Distance Caregiver geht in Vorleistung, wenn er:sie eine:n Long Distance Relative unterstützt, ohne selbst Unterstützung zu erhalten.

1 Einleitung

Du hilfst mir, ich helfe dir. Der unentgeltliche Austausch von Dienstleistungen ist vermutlich so alt wie die Menschheit selbst. Während Angebot und Nachfrage in menschheitsgeschichtlich frühen Zeiten durch räumliche Distanzen stark begrenzt waren, erlauben Informations- und Kommunikationstechnologien (IKT) heutzutage einen ortsunabhängigen Dienstleistungsaustausch. Eine zentrale Technologie für die orts- und zeitunabhängige Vernetzung von Menschen stellt hierbei das Internet dar. Auf digitalen Marktplätzen und Plattformen können Dienstleistungen angeboten und nachgefragt werden, wobei sich für die sogenannte Plattformökonomie spezifische Marktdynamiken ergeben. Die genaue Ausgestaltung der jeweiligen Plattform bzw. des Marktplatzes wird hierbei unter anderem durch die Zielgruppen und die Art der angebotenen Dienstleistungen bestimmt.

Die Zielgruppe der hier relevanten Tauschbörse, welche im Forschungs- und Entwicklungsprojekt „AniTa“ des Departments Pflege und Management der Hochschule für Angewandte Wissenschaften Hamburg (HAW) entwickelt wurde, bilden sogenannte Long-Distance-Caregiver (LDC). LDC sind Personen, die über größere räumliche Distanzen ihre meist älteren Angehörigen unterstützen. Die Angehörigen der LDC werden im Folgenden als Long-Distance-Relatives (LDR) bezeichnet. Über eine Webseite können sich LDC bei der Tauschbörse anmelden, sich vernetzen und Unterstützungsleistungen austauschen. Durch die manuelle Verwaltung der teilnehmenden LDC kann sich ein hoher Zeitaufwand für die Projektmitarbeiter:innen ergeben. Ein möglicher Ansatz zur Reduzierung dieses Zeitaufwands stellt eine Automatisierung der Geschäftsprozesse dar.

Die Entwicklung eines Informationssystems, im folgenden AniTa-Tool, zur Geschäftsprozessautomatisierung im Projekt „AniTa“ stellt das übergeordnete Ziel dieser Arbeit dar. Hierfür werden im zweiten Kapitel das Vorgehen und die Ergebnisse der Erhebung und Analyse der Anforderungen an das AniTa-Tool und Rahmenbedingungen seines Einsatzes beschrieben.

Aufbauend hierauf wird im dritten Kapitel der Entwurf der Softwarearchitektur des AniTa-Tools dargestellt. Besonderheiten der Implementierung der Software werden im vierten Kapitel beschrieben, gefolgt von einer Besprechung der Maßnahmen zur Qualitätssicherung in Kapitel 5. Die Arbeit schließt mit einer Evaluation des Systems und der zentralen Algorithmen in Kapitel 6 und einem Fazit in Kapitel 7 ab.

Zunächst werden jedoch in den folgenden Abschnitten des ersten Kapitels das Phänomen des Distance-Caregiving (DC) und die Zielgruppe der LDC genauer beleuchtet (Abschnitt 1.1). Nach einer Beschreibung des Projekts „AniTa“ in Abschnitt 1.2, folgen Erläuterungen zu den Besonderheiten der Plattformökonomie (Abschnitt 1.3) und Ansätzen der Geschäftsprozessautomatisierung (Abschnitt 1.4). Das erste Kapitel schließt mit einer detaillierten Darstellung der Problemstellung, Zielsetzung und des Vorgehens dieser Arbeit in Abschnitt 1.5 ab.

1.1 Distance Caregiving – Pflege auf Distanz

Infolge des demographischen Wandels nimmt in Deutschland der relative Anteil der älteren Menschen an der Gesamtbevölkerung zu, während der Anteil der jüngeren Menschen abnimmt. Da sich das Risiko einer Pflegebedürftigkeit mit steigendem Alter erhöht, steigt somit auch der relative Anteil der unterstützungs- und pflegebedürftigen Menschen. In zunehmendem Maße werden diese Unterstützungs- und Pflegeleistungen durch informell Pflegende geleistet. (Geyer and Schulz, 2014) Informelle Pflegepersonen sind in der Regel die erwachsenen Kinder der älteren, unterstützungsbedürftigen Personen oder deren Partner:innen (Nowossadeck et al., 2016). Durch eine steigende Arbeits- und Wohnortsmobilität leben erwachsene Kinder und ihre Eltern in Deutschland jedoch nicht selten räumlich voneinander getrennt. Durch diese Kombination aus Arbeits- und Wohnortsmobilität und demographischem Wandel gewinnt das Phänomen der Pflege und Unterstützung von Angehörigen über räumliche Distanz, das sogenannte „Distance-Caregiving“ (DC), an Bedeutung. (Kricheldorff et al., 2019)

Die Gruppe der Personen, die ihre Angehörigen über größere Distanzen hinweg pflegen bzw. unterstützen – die sogenannten „Long-Distance-Caregiver“ (LDC) – erfährt jedoch bislang noch geringe Beachtung im wissenschaftlichen oder öffentlichen Diskurs. So gibt es für den deutschsprachigen Raum nur wenige Veröffentlichungen, die sich mit der Lebenssituation und den Bedarfen der LDC widmen. Auch an Unterstützungsangeboten, die sich spezifisch an die

LDC richten, mangelt es. Entsprechend werden die Leistungen der LDC oft nicht als Pflege im klassischen Sinne wahrgenommen, da nicht in allen Fällen sogenannte Hands-On-Care geleistet wird, sondern häufig aus der Ferne Organisations- und Beratungsleistungen erbracht werden. Hierbei scheint es für die LDC von zentraler Bedeutung zu sein über das aktuelle Versorgungsgeschehen vor Ort zeitnah informiert zu sein, wozu es wiederum funktionierende soziale Netzwerke bedarf. (Zentgraf et al., 2019) Eben diesem Aufbau sozialer Netzwerke innerhalb der Gruppe der LDC widmet sich das Projekt AniTa.

1.2 Das Projekt AniTa

AniTa steht für Angehörige im Tausch und ist ein vom GKV Spitzenverband gefördertes Projekt am Department für Pflege und Management der Hochschule für Angewandte Wissenschaften Hamburg (HAW). Als Forschungs- und Entwicklungsprojekt verfolgt es zum einen das Ziel weiteres Wissen über die Zielgruppe der LDC zu generieren und zum anderen ein Angebot zu schaffen, welches LDC miteinander vernetzt und einen deutschlandweiten Tausch von Unterstützungsleistungen ermöglicht.

Realisiert wird dies im Sinne einer Tauschbörse oder auch Sharing-Plattform (SP). Die grundlegende Idee dieser SP ist, dass die hier angemeldeten LDC wohnortnah Unterstützung für einen Angehörigen eines anderen LDC anbieten und ihr entfernt lebender Angehöriger im Gegenzug Unterstützung durch einen anderen LDC erhält. Neben den LDC können sich auch Personen bei der SP anmelden, die lediglich ihre Unterstützung anbieten wollen, aber im Gegenzug keine Unterstützung benötigen. Die konkrete Ausgestaltung des Unterstützungsarrangements bleibt hierbei den LDC und ihren Angehörigen überlassen.

Über ein Formular auf der Projektwebseite (anita-familie.de) können LDC sich und ihre Angehörigen bei der SP anmelden. Die Anmeldungen werden via E-Mail an ein Funktionspostfach gesendet. Die weiteren Prozessschritte werden manuell durch die Projektmitarbeiter:innen durchgeführt: Von der Übertragung der Stammdaten, über das Verfassen der E-Mail zur Anmeldungsbestätigung und die Visualisierung der Anmeldungen bis zur Suche möglicher Tauschpaare, dem sogenannten Matching. Eine genauere Beschreibung der einzelnen Prozessschritte findet sich in Kapitel 2 Abschnitt 2.1.

Um eine Weiterführung der Tauschbörse über die Projektlaufzeit hinaus sicherzustellen, war bereits zu Projektbeginn eine Verstetigung der Projektergebnisse durch einen externen Partner geplant. Entsprechend war ein Teil des Projektplans die Suche nach einem passenden Partner, der die notwendigen Voraussetzungen mitbringt, um das Angebot des Projektes im Sinne einer Multi-Sided Platform fortzusetzen.

1.3 Multi-Sided Platforms

Multi-Sided Platforms (MP), auch Matchmaker oder Multi-Sided-Markets, sind ein Geschäftsmodell, bei dem die Wertschöpfung primär dadurch stattfindet, dass eine direkte Interaktion zwischen mindestens zwei Parteien ermöglicht wird. Hierbei bestimmen die jeweiligen Parteien die Rahmenbedingungen (z.B. Preisgestaltung) ihrer Interaktion und nicht die MP (Hagiu and Wright, 2015). Der Anschluss an die MP beruht wiederum auf einem Investment durch eine oder mehrere der Parteien, z.B. die Zahlung einer regelmäßigen Gebühr oder Provisionen. Beispiele für MPs sind etwa Marktplätze, wie eBay, der Amazon Marketplace oder Dating-Plattformen, wie Tinder oder Parship.

Eine besondere Form der MP stellen Spatial Multi-Sided Markets dar, bei denen die Marktteilnehmer davon ausgehen, sich nicht nur virtuell zu treffen. Durch die räumliche Dimension erhöht sich hierbei die Komplexität des Marktgeschehens. Ein Beispiel für einen solchen Spatial Multi-Sided Market ist Uber. (Navidi et al., 2020) Uber vermittelt den Kontakt zwischen Fahrgästen, die von einem Ort zum anderen kommen müssen und Fahrer:innen, welche die Transportdienstleistung anbieten. Die Nachfrage der Fahrgäste muss im Falle von Uber also räumlich mit dem Angebot der Fahrer:innen übereinstimmen.

Eine weitere Besonderheit der MPs stellten sogenannte indirekte Netzwerkeffekte dar. Hierbei wird die Plattform für die teilnehmenden Parteien umso attraktiver, je mehr Teilnehmende der jeweils anderen Parteien aktiv sind. Mit steigenden Teilnehmerzahlen kann es bei MPs auf Grund der indirekten positiven Netzwerkeffekte zu einem exponentiellen Anstieg der Teilnehmenden kommen. Die Schwelle von Teilnehmenden, ab welcher es zu einem solchen exponentiellen Anstieg kommt, wird auch als kritische Masse bezeichnet. Ist diese kritische Masse erreicht, wird angenommen, dass die MP selbst-tragend wachsen kann. Je mehr Verkäufer:innen auf eBay ihre Ware anbieten, desto attraktiver wird es für die Kunden:innen bei eBay nach

Waren zu suchen. Eine höhere Anzahl von Kunden:innen wiederum macht es für Händler:innen attraktiver bei eBay aktiv zu werden. Im Falle von Uber führen mehr Fahrer:innen an einem Ort zu einem besseren Angebot und dadurch auch einer höheren Attraktivität für die Fahrgäste, etwa aufgrund von kürzeren Wartezeiten. Eine höhere Anzahl von Kunden:innen wiederum ermöglicht den Fahrer:innen höhere Einnahmen. (Navidi et al., 2020)

Dating-Plattformen teilen sich zwar einige Aspekte mit anderen MPs: so sind positive Netzwerkeffekte anzunehmen, da bei einem wachsenden Pool an interessierten Teilnehmenden auch die Wahrscheinlichkeit steigt, dass ein:e passende:r Partner:in gefunden wird. Auch der räumliche Aspekt spielt ähnlich wie bei Uber, zumindest partiell eine Rolle. Anders als bei Uber oder eBay finden jedoch soziale Transaktionen statt und die Teilnehmenden scheiden nach einer erfolgreichen Vermittlung in der Regel aus dem Marktgeschehen aus. Es gibt zudem wenig Feedback über den Wettbewerb innerhalb des Marktes. (Burtch and Ramaprasad, 2016)

Auch die AniTa-SP kann als MP bzw. als Spatial Multi-Sided Market gedacht werden. AniTa ermöglicht die Kommunikation zwischen LDCs zum Zwecke des Austausches von Unterstützungs- und Pflegeleistungen. Die LDCs sind frei in der Gestaltung der Rahmenbedingungen ihrer Interaktion und „investieren“ Unterstützungs- und Pflegeleistungen, um bei der Tauschbörse angemeldet zu sein. Da die Teilnehmenden die Versorgung einer Person an ihrem Wohnort anbieten und dafür die Versorgung ihrer LDR an einem anderen Ort nachfragen, spielt zudem der räumliche Aspekt bei AniTa eine entscheidende Rolle. Auch entsprechende Netzwerkeffekte sind anzunehmen. Je mehr Teilnehmende bei der Tauschbörse mitmachen, desto schneller können passende Tauschpartner:innen gefunden und Unterstützungsleistungen ausgetauscht werden. Erfolgreicher Austausch zwischen den Teilnehmenden wird wiederum über Mundpropaganda und positive Pressebeiträge zu steigenden Teilnehmerzahlen führen. Bei längeren Wartezeiten hingegen wird sich die Lebenssituation der Teilnehmenden verändert haben, sodass eventuell kein Bedarf mehr an einer Teilnahme besteht. Unter der Annahme, dass sich bei der AniTa-SP ähnliche Dynamiken wie bei einer Dating-Plattform ergeben, kann eine Automatisierung der Geschäftsprozesse nicht nur der Entlastung der Projektmitarbeiter:innen dienen, sondern zudem die Vermittlung beschleunigen und somit zum Erfolg der SP beitragen.

1.4 Geschäftsprozessautomatisierung

Geschäftsprozessautomatisierung ist die durch IKT gestützte Automatisierung von komplexen Geschäftsprozessen oder Teilprozessen, die oft über die Verwaltung von Stammdaten hinausgehen. Hierbei liegt der Fokus in der Regel auf kritischen Kernprozessen eines Unternehmens in Abgrenzung zu analytischen Prozessen. (Gartner, 2021)

Neben dem übergeordneten Ziel einer Effizienzsteigerung sollen durch die Geschäftsprozessautomatisierung weitere Ziele erreicht werden. So soll u.a. die Ergebnisqualität, etwa durch die Vermeidung menschlicher Fehler, gesteigert werden und Mitarbeitende von repetitiven Aufgaben befreit werden, sodass sie sich auf wertschaffende und kreative Tätigkeiten fokussieren können.

Eine erfolgreiche Geschäftsprozessautomatisierung wiederum benötigt zunächst detailliertes Wissen über die Geschäftsziele, die jeweiligen Geschäftsprozesse und die organisatorischen und rechtlichen Rahmenbedingungen. So beginnen Projekte mit dem Ziel der Geschäftsprozessautomatisierung in der Regel mit der Dokumentation und Analyse der Ist-Prozesse und anschließender Ableitung eines Soll-Prozessmodells, in dem die Teilprozesse alle oder zum Teil automatisiert ablaufen. Dieses Prozessmodell kann als Grundlage für IT-Projekte zur Prozessautomatisierung dienen. (Freund and Rücker, 2019)

Insbesondere wenn auf Basis von Geschäftsdaten automatische bzw. algorithmische Entscheidungen getroffen werden sollen, ist eine hohe Daten- und Informationsqualität notwendig, damit Fehlentscheidungen vermieden werden können. Als Grundlage für die Einschätzung von Informationsqualität können die Informationsqualitäts-Dimensionen (IQ-Dimensionen) herangezogen werden, welche ursprünglich von Wang und Strong entwickelt wurden (siehe hierzu Rohweder et al., 2021). Von den insgesamt 15 IQ-Dimensionen ist für die Vermeidung von automatischen Fehlentscheidungen vor allem die Fehlerfreiheit, also die Übereinstimmung der in den Daten enthaltenen Informationen mit der Realität, sowie deren Aktualität von großer Bedeutung. Damit die Daten automatisch verarbeitet werden können, sind zudem die Dimensionen der Zugänglichkeit und der einheitlichen Darstellung bedeutsam. Denn nur wenn das Softwaresystem unmittelbar auf die Daten zugreifen kann und diese möglichst in strukturierter Form vorliegen, kann eine direkte Verarbeitung erfolgen.

Entsprechend stellten die Prozessanalyse sowie die Herstellung einer möglichst hohen Daten- und Informationsqualität wichtige Schritte in der Entwicklung des AniTa-Tools dar.

1.5 Problemstellung und Zielsetzung

Bereits bei einer regulären Anzahl von Anmeldungen an der SP, aber insbesondere bei Anmeldungsspitzen nach Medienberichten über das Projekt ergibt sich für die Projektmitarbeiter:innen ein hoher Arbeitsaufwand. Dieser ist neben der Verwaltung der Stammdaten und der Kommunikation mit den Teilnehmenden insbesondere durch die manuelle Suche nach potentiellen Tauschpartner:innen bedingt. Um die Bedürfnisse der Teilnehmenden nach einer zeitnahen Vermittlung zu befriedigen und so von positiven Netzwerkeffekten zu profitieren, bedarf es einer hohen Anzahl an Anmeldungen, welche zeitnah bearbeitet werden müssen. Vor dem Hintergrund, dass es durch fixe Budgetpläne in drittmittelgeförderten Forschungs- und Entwicklungsprojekten nicht ohne weiteres möglich ist Personal aufzustocken, ergibt sich die Notwendigkeit die Effizienz und Ergebnisqualität der Prozesse in der Verwaltung der Anmeldungen zu steigern. Nicht zuletzt können die so freiwerdenden Ressourcen für die Forschungstätigkeiten zu den Zielgruppen der LDC und LDR und das Marketing für die SP genutzt werden.

Mit dem Ziel der Geschäftsprozessautomatisierung werden im Rahmen dieser Arbeit die bestehenden Geschäftsprozesse und Rahmenbedingungen analysiert und Möglichkeiten und Grenzen der Automatisierung im Projekt AniTa abgeleitet. Es werden Anforderungen an ein System zur Geschäftsprozessautomatisierung entwickelt und ein entsprechendes System entworfen. Anhand des Entwurfs wird das System unter dem Namen „AniTa-Tool“ implementiert, evaluiert und die Ergebnisse kritisch diskutiert.

Da durch eine enge Anbindung an das Projektteam die Möglichkeit einer regelmäßigen Abstimmung der erhobenen Anforderungen, Spezifikationen und Entwürfe mit den Vorstellungen des Projektteams gegeben war, wurde eine iteratives Vorgehen gewählt. Hierbei wurden die Schritte der Analyse von Anforderungen und Rahmenbedingungen, des Entwurfs und der Dokumentation der Systemarchitektur, ihrer Implementierung und der Evaluation mehrfach durchlaufen. In den folgenden Kapiteln wird hierbei immer der finale Stand der jeweiligen Schritte beschrieben.

2 Analyse

In einer initialen Erhebung wurde gemeinsam mit den Projektmitarbeiter:innen anhand eines informellen Prozessablaufplans der derzeitige Workflow in der Bearbeitung der Anmeldungen der Tauschbörse und die gewünschten Automatisierungen als Ist-Prozess bzw. Soll-Prozess visualisiert (Abschnitt 2.1). Da die Freiheitsgrade in der technischen Umsetzung durch die organisatorischen und rechtlichen Rahmenbedingungen sowie die bestehenden Altsysteme beschränkt sein können, erfolgte eine Analyse der Altsysteme und des Datenbestandes (Abschnitt 2.2) sowie der Rahmenbedingungen (Abschnitt 2.3). Zudem wurden die zentralen Stakeholder an dem zu entwickelnden System identifiziert (Abschnitt 2.4). Als Grundlage für den Architekturentwurf des Systems wurden die Anforderungen an dieses in Form von User-Stories einschließlich Akzeptanzkriterien spezifiziert (Abschnitt 2.5).

Die in den folgenden Abschnitten dargestellten Ergebnisse stellen den finalen Stand der Erhebungen dar, welcher sich im Laufe mehrerer Iterationen ergeben hat.

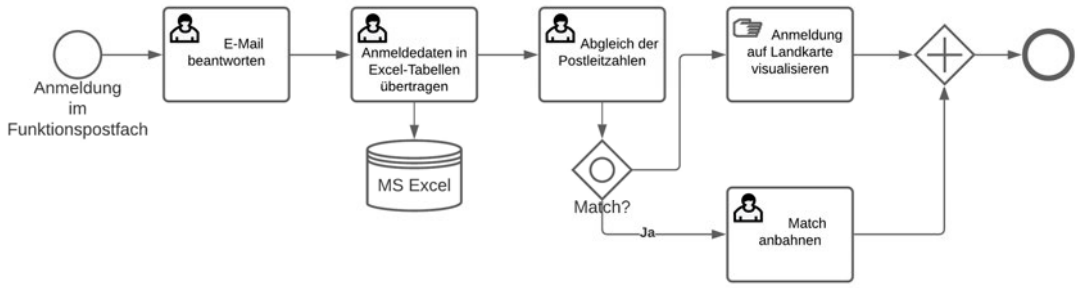
2.1 Prozessanalyse

In einem ersten Schritt wurden die einzelnen Prozessschritte der bestehenden Ist-Prozesse sowie der gewünschten Soll-Prozesse durch die Projektmitarbeiter:innen in informellen Prozessablaufplänen dargestellt. Diese Prozessablaufpläne wurden anschließend in die Business Process Model and Notation (BPMN) (Object Management Group, 2014) übersetzt und so formalisiert.

2.1.1 Ist-Prozess

Im bestehenden Ist-Prozess (siehe Abbildung 1) beginnen die mit der SP verbundenen Tätigkeiten der Projektmitarbeiter:innen, wenn sich ein interessierter LDC über das Anmeldeformular auf der Webseite der SP anita-familie.de anmeldet und die Anmeldedaten im Funktionspostfach eintreffen. Die darauffolgenden Prozessschritte werden allesamt durch die Projektmitarbeiter:innen, entweder softwaregestützt oder manuell durchgeführt.

Abbildung 1: Darstellung des Ist-Prozesses im Betrieb der Sharing-Plattform nach BPMN



Im ersten Schritt wird die Anmeldeemail durch die Projektmitarbeiter:innen beantwortet. Hierzu werden in der Regel vorliegende Standardformulierungen verwendet. In Einzelfällen erfolgt jedoch eine Individualisierung der Bestätigungsmail. Im zweiten Prozessschritt werden die Stammdaten aus der Anmeldemail händisch in eine Exceltabelle übertragen und so persistiert. Daraufhin erfolgt ein Abgleich der Postleitzahlen der neu angemeldeten LDC und LDR mit den bestehenden Anmeldungen mit dem Ziel potenzielle Matches zu identifizieren. Hierbei werden nicht nur exakte Übereinstimmungen der Postleitzahlengebiete gesucht, sondern auch angrenzende Postleitzahlengebiete berücksichtigt. Wird ein potenzielles Match identifiziert, werden die Tauschparteien durch die Projektmitarbeiter:innen kontaktiert und so ein Match angebahnt. Zudem werden alle Anmeldungen auf einer Landkarte im Büro der Projektmitarbeiter:innen visualisiert. Abweichend von dem hier dargestellten Prozess kann eine Anmeldung zur Tauschbörse auch telefonisch erfolgen.

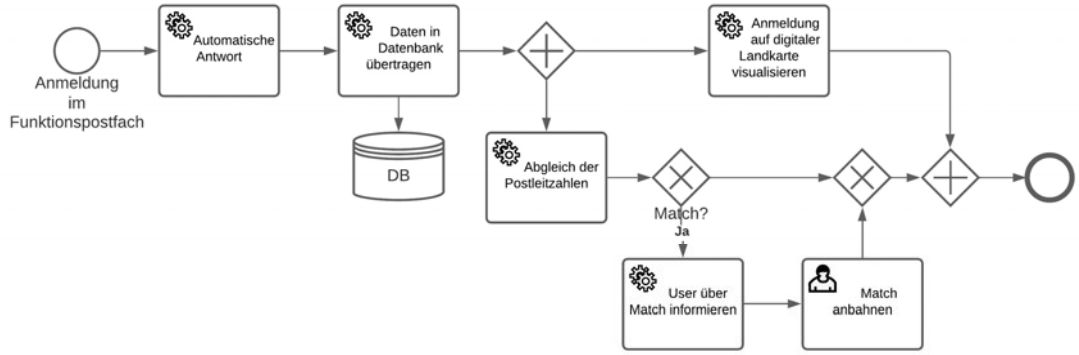
Insbesondere die händische Übertragung der Anmeldedaten und der manuelle Abgleich der Postleitzahlen stellen sich hierbei als besonders zeitaufwändig dar. Da die Systematik der Postleitzahlengebiete eine direkte Identifikation von aneinander angrenzenden

Postleitzahlengebieten anhand der Postleitzahl nicht zulässt, ist eine manuelle Prüfung als fehleranfällig anzunehmen. Insbesondere das Übersehen von potenziellen Matches, also falsch negative Prüfungen, können sich hierbei negativ auf den Erfolg der SP auswirken. Falsch positive Prüfungen würden sich im Gegensatz hierzu jedoch während des Anbahnungsprozesses durch die Rückmeldung der Tauschparteien korrigieren lassen.

2.1.2 Soll-Prozess

Im Soll-Prozess (s. Abbildung 2) werden die einzelnen Prozessschritte größtenteils als vollautomatisiert spezifiziert. So soll nach Eingang einer Anmeldung im Funktionspostfach eine automatische Antwort via E-Mail erfolgen und die Stammdaten automatisch in eine Datenbank übertragen werden. Wird beim automatischen Abgleich der Postleitzahlen ein Match identifiziert sollen die Nutzer:innen über dieses informiert werden, sodass die Anbahnung wie im Ist-Prozess durch die Projektmitarbeiter:innen erfolgen kann. Die Anmeldedaten sollen zudem auf einer digitalen Landkarte visualisiert werden.

Abbildung 2: Darstellung des Soll-Prozesses im Betrieb der Sharing-Plattform nach BPMN

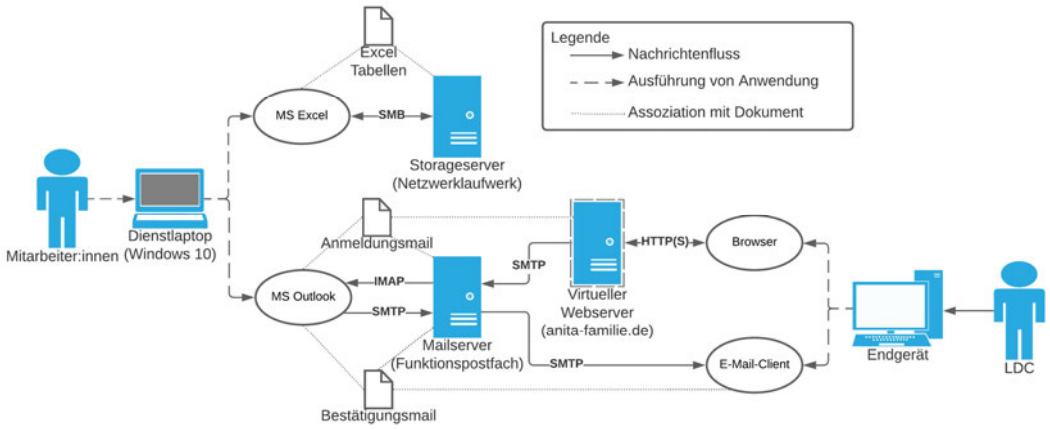


Die primären Ziele einer solchen Geschäftsprozessautomatisierung liegen hierbei in einer Reduzierung des Zeitaufwandes und der Fehleranfälligkeit durch die automatische Übertragung der Daten in die Datenbank und den automatischen Abgleich der Postleitzahlen. Um den Forschungsfragen des Projektes nachgehen zu können, ist es hingegen naheliegend, dass die Anbahnung und Begleitung von Matches weiterhin durch die Projektmitarbeiter:innen koordiniert wird und ein Überblick über bestehende Anmeldungen notwendig bleibt.

2.2 Altsysteme und Datenbestand

Um sich bei der SP anzumelden, nutzen LDC einen Webbrowser, über den sie das Anmeldeformular auf der Webseite anita-familie.de aufrufen. Über das Hypertext Transfer Protocol (HTTP) (R. Fielding and J. Reschke, 2014) werden die Anmeldedaten auf den virtuellen Webserver übertragen, welcher diese in Form einer E-Mail mittels Simple Mail Transfer Protocol (SMTP) (Klensin, 2008) an das Funktionspostfach des Projektteams schickt. Zum Abruf der E-Mails via Internet Message Access Protocol (IMAP) (Crispin, 2003) nutzen die Projektmitarbeiter:innen die Applikation Microsoft Outlook. Zum Versand der Bestätigungsmail werden wiederum Microsoft Outlook und SMTP genutzt. Um die Stammdaten aus der Anmeldungsmail in die Excel-Tabellen zu übertragen, nutzen die Projektmitarbeiter:innen Microsoft Excel. Damit alle Mitarbeiter:innen Zugriff auf die Excel-Tabellen haben, werden diese auf einem Netzwerklaufwerk gespeichert. Für die Übertragung der Daten von den Dienstlaptops der Mitarbeiter:innen mit dem Betriebssystem Microsoft Windows 10 auf der Netzwerklaufwerk auf dem Storageserver wird das Server Message Block Protocol (SMB) (Microsoft Corporation, 2021) verwendet.

Abbildung 3: Darstellung der verwendeten Systeme und Kommunikationsprotokolle im Projekt AniTa



Die gesamte technische Infrastruktur, sprich der Storageserver, der Mailserver und der virtuelle Webserver werden vom Informationstechnik Service Center (ITSC) der HAW betrieben.

Ebenso werden die Systemupdates der Dienstlaptops der Mitarbeiter:innen durch die HAW verwaltet.

2.2.1 Webserver, Webseite und Anmeldeformular

Die Projektwebseite (anita-familie.de), über die sich interessierte Personen bei der SP anmelden und aktuelle Informationen zum Projektstand beziehen können, basiert auf dem Open Source Content Management System „Contao“ (contao.org) und wurde von einem externen Dienstleister entwickelt. Wie bereits in Abschnitt 2.2 erläutert, wird das Hosting der Website durch das ITSC unter Ausschluss jeglicher Wartungsleistungen gewährleistet. Für die Projektmitarbeiter:innen ist der Zugriff auf den verwendeten virtuellen Server (siehe Abbildung 3) nicht möglich. Daher können weder genauere Angaben zu den verwendeten Technologien gemacht noch Änderungen an der Webseite vorgenommen werden, die über Konfigurationen im Contao-Backend hinausgehen.

Die Schnittstelle zu den interessierten Menschen stellt das in die Webseite integrierte Anmeldeformular dar. Wie in Tabelle 1 ersichtlich ist, erfolgt eine Prüfung der Eingabewerte lediglich für die Formularfelder E-Mail-Adresse, Postleitzahl und Postleitzahl Angehöriger, welche anhand der in HTML5 definierten Constraints validiert werden (vgl. W3C (2017)). Für die Postleitzahlfelder bedeutet dies, dass clientseitig nicht geprüft wird, ob eine korrekte deutsche Postleitzahl angegeben wurde, sondern lediglich, ob es sich um numerische Zeichen handelt. Da die Anmeldung als ein Single-Opt-In-Verfahren angelegt ist, erfolgt zudem keine automatische Überprüfung, ob die angegebene E-Mail-Adresse tatsächlich vergeben ist oder ob die angegebene E-Mail-Adresse zu der Person gehört, welche das Formular ausgefüllt hat.

Da die Anmeldedaten via SMTP übermittelt werden, gibt es für die Teilnehmenden keine direkte Möglichkeit ihre Daten zu korrigieren. Eine Überprüfung der Anmeldedaten kann nach dem Absenden daher nur durch die Projektmitarbeiter:innen vorgenommen werden, wenn diese die E-Mail mit den Formulardaten erhalten haben. Augenscheinliche Fehler können an dieser Stelle direkt oder in Rücksprache mit dem jeweiligen LDR korrigiert werden. Bei fehlerhaften Kontaktdaten, z.B. wenn die Bestätigungsmail nicht zugestellt werden konnte, müsste hingegen der gesamte Datensatz gelöscht werden.

Tabelle 1: Datenfelder im Anmeldeformular für die SP auf anita-familie.de

No.	*Formularfeld*	*Typ*	*Beschreibung*	*Erlaubte Werte*
AF1	Teilnahme*	Radio	Grund der Anmeldung bei der Tauschbörse.	1. Unterstützung geben und erhalten., 2. ausschließlich Unterstützung geben.
AF2	Umkreis*	Text	Aktivitätsradius in KM	Keine Einschränkungen
AF3	Vorname*	Text	Vorname des LDC	Keine Einschränkungen
AF4	Nachname*	Text	Nachname des LDC	Keine Einschränkungen
AF5	Telefonnummer	Tel	Telefonnummer des LDC	Keine Einschränkungen
AF6	E-Mail-Adresse*	E-Mail	E-Mail-Adresse des LDC	Gültige E-Mail-Adresse
AF7	Geburtsjahr	Text	Geburtsjahr des LDC	Keine Einschränkungen
AF8	Geschlecht	Radio	Geschlecht des LDC	1. weiblich, 2. männlich, 3. keine Angabe
AF9	Postleitzahl*	Number	Postleitzahl des LDC	Numerische Zeichen
AF10	Angehöriger* **	Checkbox	Verhältnis des LDC zu LDR	1. Mutter, 2. Vater, 3. Schwiegermutter 4. Schwiegervater, 5. Schwester, 6. Bruder, 7. Eine/n andere/n Angehörigen und zwar:
AF11	Angehöriger Extra**	Text	Verhältnis des LDC zu LDR, bei AF10 Auswahl 10.	Keine Einschränkungen
AF12	Postleitzahl Angehöriger* **	Number	Postleitzahl des LDR	Numerische Zeichen
AF13	Unterstützungsinfos	Textarea	Weitere Informationen zum Unterstützungsrahmen, Besonderheiten etc.	Keine Einschränkungen
AF14	Einwilligungserklärung*	Checkbox	Bestätigung, dass Speicherung und Verarbeitung der Daten akzeptiert wird	1. Einwilligungsbestätigung
AF15	Angehörige Informiert*	Checkbox	Bestätigung, das LDR über Teilnahme informiert wurde	1. Bestätigung, dass LDR informiert ist

* Pflichtfeld, ** Nur sichtbar, wenn im Formularfeld Teilnahme Option 1 gewählt wurde

2.2.2 Exceltabellen

Die Verwaltung und Persistierung der Anmeldedaten erfolgt mittels einer Exceldatei mit den zwei Tabellen Caregiver (CG) und Caretaker (CT). Neben den über das Anmeldeformular

übermittelten Daten werden auch telefonische Anmeldungen und Ergebnisse der weiteren Prozessschritte, wie der Anbahnung eines Matches oder der Visualisierung der Anmeldungen auf der Landkarte, über die Tabellen verwaltet. Während in der Tabelle CG primär die Informationen zu den LDC vermerkt werden (siehe Tabelle 2), finden sich in der Tabelle CT die Informationen zu den LDR wieder (siehe Tabelle 3). Eine Verknüpfung der LDC und LDR erfolgt über manuell erzeugte Unique Identifier (UID). Diese UID setzen sich aus dem Präfix A (LDC) bzw. B (LDR) und einer Ziffernfolge zusammen. Hierbei teilen sich zusammengehörige LDC und LDR die gleiche Ziffer, der LDC mit dem UID A1 lässt sich somit dem LDR mit dem UID B1 zuordnen. Im Falle von 1:N-Beziehungen wurden entsprechende Buchstaben an die UID angefügt. Ist der LDC mit dem UID A2 mit zwei LDR verbunden erhalten diese die UID B2a bzw. B2b. Der Fall, dass ein LDR mit mehreren LDC verbunden ist, wird analog behandelt.

Bei der Sichtung der Tabellen wurde deutlich, dass in einigen Spalten nicht mit einer einheitlichen Syntax und teils mit farblichen Markierungen gearbeitet wurde. Diese Abweichungen in der Syntax und Farbmarkierungen sind für Menschen leicht zu verstehen. Um sie einer maschinellen Verarbeitung zugänglich zu machen, bedarf es jedoch einer manuellen Bereinigung oder einer Berücksichtigung aller Fälle, die sich durch die uneinheitliche Syntax ergeben. Zudem zeigt sich, dass in den Tabellen 1:N-Beziehungen zwischen den LDC und LDR berücksichtigt werden, welche jedoch nicht über das Formular auf der Webseite abgedeckt werden (siehe Abschnitt 2.2.1). Diese Informationen ergeben sich entweder im Verlauf durch die Kommunikation mit den LDR, bei telefonischer Anmeldung oder durch zusätzliche Informationen in den Freifeldern des Anmeldeformulars.

Tabelle 2: Spalten der Excel-Tabelle Caregiver einschließlich Erläuterungen

No.	***Spaltenname***	***Beschreibung***	***Quelle***	***Besonderheiten***
CG1	ID CG	UID des LDC	m	/
CG2	Name	Nachname und Vorname des LDC	AF3, AF4	/
CG3	Anmeldedatum	Datum der Anmeldung	m	/
CG4	Geburtsdatum	Geburtsdatum des LDC	AF7	Keine einheitliche Syntax: Entweder Geburtsdatum, -jahr oder Alter
CG5	E-Mail-Adresse / Telefonnummer	E-Mail-Adresse u./o. Telefonnummer des LDC	AF5, AF6	Keine einheitliche Syntax: Unterschiedliche Trennzeichen, teilweise mit Zusatz „Tel:“
CG6	m/w	Geschlecht des LDC	AF8	Mögliche Werte: m o. w
CG7	PLZ	Postleitzahl des LDC	AF9	Teilweise Verwendung des Buchstaben O bei Postleitzahlen mit führender Null
CG8	Wohnort	Wohnort des LDC	m	Keine einheitliche Syntax
CG9	Einseitig tätig werden	Angabe ob LDC nur Hilfe gibt	AF1	Keine einheitliche Syntax: Farbmarkierung und teilweise Freitext
CG10	Kann tätig werden im Umkreis von	Aktivitätsradius des LDC	AF2	Keine einheitliche Syntax: Teilweise Kilometerangaben mit Zusatz km oder Kilometer, teilweise Postleitzahlen, teilweise Ortsangaben
CG11	DS akzeptiert	Angabe, ob Datenschutzvereinbarung akzeptiert wurde	AF14	Mögliche Werte: ja, nein
CG12	CT informiert	Angabe ob LDA über Teilnahme informiert wurde	AF15	Mögliche Werte: ja, nein
CG13	Angefragt als TP	Angabe LDC als Tauschpartner angefragt wurde	m	Keine Einheitliche Syntax: Vorwiegend Schema UID CG → UID CT, teilweise mit Datumszusatz
CG14	Ja/Nein	Angabe LDC bereit ist dem Tausch als TP zuzustimmen	m	Mögliche Werte: ja, nein
CG15	Angefragt als Angehörige	Angabe, ob der LDC als Angehöriger angefragt wurde	m	Keine Einheitliche Syntax: Vorwiegend Schema IC CG → ID CT, teilweise mit Datumszusatz
CG16	Ja/nein	Angabe ob LDC als Angehöriger bzw. der LDR bereit ist dem Tausch zuzustimmen	m	Mögliche Werte: ja, nein
CG17	Zusatzinformation	Zusatzinformationen zum LDC	AF13	/
CG18	Auf Karte eingetragen	Angabe, ob der LDC auf der Landkarte visualisiert wurde	m	Mögliche Werte: ja, nein

Anmerkung. Die Quelle bezeichnet die primäre Quelle der Daten, dass die Daten einen anderen Ursprung haben oder verschiedene Quellen zusammengeführt werden ist hierbei nicht auszuschließen.
m: Manuelle Eingabe, Daten im Prozess entstanden

Tabelle 3: Spalten der Excel-Tabelle Caretaker einschließlich Beschreibungen

No.	***Spaltenname***	***Beschreibung***	***Quelle***	***Besonderheiten***
CT1	ID CT	UID des LDR	m	/
CT2	Verhältnis	(Verwandtschafts-)Verhältnis zwischen LDC und LDR	AF10	Keine einheitliche Syntax, teilweise Mehrfachnennungen
CT3	PLZ	Postleitzahl des LDR	AF12	Teilweise Verwendung des Buchstaben O bei Postleitzahlen mit führender Null
CT4	Ort	Wohnort des LDR	m	/
CT5	Informationen	Zusatzinformationen zum LDC	AF13	/
CT6	Matching	Informationen zu möglichen Matches	m	Keine einheitliche Syntax: Vorwiegend Schema IC CT → ID CG, teilweise mit Datumszusatz
CT7	Status	Informationen zum aktuellen Status eines Matches bzw. einer Anmeldung	m	Keine einheitliche Syntax: Freitext
CT8	Auf Karte eingetragen	Angabe, ob der LDR auf der Landkarte visualisiert wurde	m	/

Note: Die Quelle bezeichnet die primäre Quelle der Daten, dass die Daten einen anderen Ursprung haben oder verschiedene Quellen zusammengeführt werden ist hierbei nicht auszuschließen.
m: Manuelle Eingabe, Daten im Prozess entstanden

2.2.3 Netzwerklaufwerk

Das Projektteam koordiniert seine Arbeit über ein projektbezogenes Netzwerklaufwerk, welches vom ITSC zur Verfügung gestellt wird. Auf diesem Laufwerk werden neben der in Abschnitt 2.2.2 beschriebenen Excel-Datei auch weitere Dokumente gespeichert, die durch die Projektmitarbeiter:innen gemeinsam bearbeitet werden müssen. Eine Sicherung der Daten auf dem Netzwerklaufwerk erfolgt automatisch mittels des Dateiversionsverlaufs von Microsoft Windows 10, welcher standardmäßig auf allen Dienstlaptops der Projektmitarbeiter:innen eingerichtet ist.

Um auf das Netzwerklaufwerk zugreifen zu können, müssen sich die Projektmitarbeiter:innen im Firmennetzwerk der HAW befinden, entweder durch Nutzung des Ethernet-Zugangs in ihrem Büro oder mittels des Virtual Private Networks der HAW. In allen Fällen muss zuvor eine Authentifizierung gegen die Infrastruktur des ITSC erfolgen, bevor der Zugriff autorisiert wird.

2.3 Organisatorische und rechtliche Rahmenbedingungen

Da sich das dreiköpfige Projektteam ein Büro teilt und zum Projektbeginn kaum Arbeit im Home-Office stattfand, war die Arbeit durch kurze Kommunikationswege geprägt. Durch die parallel zum Betrieb der SP stattfindenden Forschungstätigkeiten zu Bedarfen und Lebensumständen der LDR und LDC entstehen sukzessive neue Erkenntnisse zur Zielgruppe und folglich veränderte Anforderungen an das zu entwickelnde System. Jede:r Mitarbeiter:in verfügt über einen Dienstlaptop. Darüber hinaus steht keine technische Infrastruktur mit der Möglichkeit zur Programmausführung, etwa Server oder Cloudlösungen, zur Verfügung.

Wie bereits in Abschnitt 1.2 erwähnt, war bereits zu Projektbeginn geplant, die SP und die Daten der angemeldeten LDC und LDR an einen externen Verstetigungspartner zu übergeben. Da zum Zeitpunkt der Entwicklung des Anita-Tools ein entsprechender Partner noch nicht bestimmt war, können keine Annahmen zu den verwendeten Systemen und Arbeitsprozessen des Partners in die Entwicklung einfließen.

Da das Projekt am Department Pflege und Management der HAW angegliedert ist, wurden bereits im Vorfelde des Launches der SP eine Datenschutzerklärung, ein Verzeichnis der Verarbeitungstätigkeiten sowie zugehörige technisch organisatorische Maßnahmen mit den Datenschutzbeauftragten der HAW abgestimmt. Diese Dokumente geben den rechtlichen Rahmen für die Speicherung und Verarbeitung der Anmeldedaten der Teilnehmenden an der SP vor. In ihnen wurde unter anderem festgehalten, dass die Daten der Teilnehmenden über das Anmeldeformular in das Funktionspostfach übertragen und auf dem Netzwerklaufwerk gespeichert werden.

2.4 Stakeholder

Insgesamt lassen sich vier relevante Gruppen von Stakeholdern am AniTa-Tool identifizieren:

1. Die Projektmitarbeiter:innen als Nutzer:innen des Tools, deren zentrales Interesse in einer Steigerung der Effizienz und Reduktion der Fehler im Rahmen der Verwaltung der Teilnehmenden an der SP besteht.

2. Die teilnehmenden LDC und LDR, die wie die Projektmitarbeiter:innen ein Interesse an einer schnellen Vermittlung haben, zudem aber auch an einem hohen Grad an Datensicherheit und Schutz vor Datenverlust interessiert sind.
3. Die HAW als übergeordnete Organisation des Projekts, die im Zusammenhang mit der Entwicklung des AniTa-Tools insbesondere an der Einhaltung der Datenschutzerklärung und dem Verzeichnis der Verarbeitungstätigkeiten interessiert ist.
4. Ein noch unbekannter externer Verstetigungspartner, der ein Interesse daran hat, das AniTa-Tool ohne größeren Aufwand in seine technische Infrastruktur integrieren zu können.

2.5 Anforderungen

Die Anforderungen an ein Softwaresystem legen fest, welche Eigenschaften die verschiedenen Stakeholder von diesem Softwaresystem erwarten (Balzert, 2009). Die in den folgenden zwei Abschnitten dargestellten Anforderungen an das AniTa-Tool stellen das Ergebnis mehrerer Iterationen aus Analyse, Entwurf, Implementierung und Evaluation dar. Bei Betrachtung der finalen Anforderungen wird deutlich, dass diese von dem in Abschnitt 2.1.2 spezifizierten Soll-Prozess abweichen. Dies ist dadurch bedingt, dass sich im Entwicklungsverlauf herausstellte, dass eine Vollautomatisierung der Prozesse an verschiedenen Stellen nicht oder nur mit großem Aufwand realisierbar ist. Die Gründe hierfür liegen zum einen in den bestehenden Altsystemen und der verfügbaren technischen Infrastruktur und zum anderen in den rechtlichen und organisatorischen Rahmenbedingungen, aber auch in den begrenzten zeitlichen und personellen Ressourcen für die Entwicklungstätigkeit. Auf die Einschränkungen wurde entsprechend mit Anpassungen der Anforderungen reagiert. An welchen Stellen im Entwicklungsprozess, welche Gründe zu welchen Anpassungen der Anforderungen und zu welchen Designentscheidungen geführt haben, wird genauer in Kapitel beschrieben.

Die Anforderungen an das AniTa-Tool sind in Form von User Stories und zugehörigen Akzeptanzkriterien dokumentiert. Vergleicht man User Stories mit klassischen Formen der katalogisierten Dokumentation, stellen diese eine relativ offene Beschreibung einer Anforderung an ein Softwaresystem dar. User Stories beschreiben, in welcher Benutzerrolle Nutzer:innen welches Ziel aus welchem Grund mit dem Softwaresystem erreichen möchten. Der Fokus einer

User Story liegt daher stets auf der Fachlichkeit der Kund:innen und nicht auf technischen Aspekten der Implementierung. Im Entwicklungsprozess dienen die User Stories als Grundlage für und Dokumentation von Kommunikation innerhalb des Entwicklungsteams und mit den Kund:innen. Den User Stories werden Akzeptanzkriterien zugeordnet, die diese zum einen weiter spezifizieren und zum anderen dazu dienen zu überprüfen, ob eine User Story umgesetzt ist. (Wirdemann and Mainusch, 2017)

2.5.1 Funktionale Anforderungen

Funktionale Anforderungen beschreiben, welche Funktionen bzw. Services ein Softwaresystem zur Verfügung stellen soll (Balzert, 2009). Da die Projektmitarbeiter:innen die Nutzerinnen des AniTa-Tools darstellen, beschreiben die User Stories in diesem Abschnitt ausschließlich deren Ziele und Gründe für diese Ziele. Die User Stories sind in weitere Unterabschnitte gegliedert, um eine Zuordnung zu den einzelnen Komponenten des AniTa-Tools, welche in Kapitel 3 beschrieben werden zu vereinfachen.

Bearbeitung von Anmeldungen

In der Bearbeitung der Anmeldungen bei der SP muss durch die Projektmitarbeiter:innen zum einen eine Bestätigungsmail erstellt und abgeschickt werden und zum anderen die Anmeldedaten übertragen und persistiert werden (siehe Abbildung 1). Die User Story 1 und User Story 2 adressieren diese Prozessschritte und die Ziele der Effizienzsteigerung und Fehlervermeidung.

User Story 1: Erzeugung von Bestätigungsmails

US 1	Als Projektmitarbeiter:in möchte ich eine automatische Bestätigungsmail basierend auf Anmeldedaten erzeugen können, um Zeit zu sparen.
AK 1	1. Die Anrede, der Name und das Verwandtschaftsverhältnis werden automatisch im E-Mail-Text an die Anmeldedaten angepasst. 2. Die E-Mail-Adressen der Empfänger:innen werden automatisch aus den Anmeldedaten übernommen. 3. Die erzeugte E-Mail enthält die offizielle Signatur des Projektes. 4. Nutzer:innen haben die Möglichkeit vor dem Absenden individuelle Anpassungen am Text der jeweiligen E-Mail vorzunehmen.

User Story 2: Einpflegen von Anmeldedaten

US 2	Als Projektmitarbeiter:in sollte ich die Anmeldedaten der Teilnehmenden in das System einpflegen können, ohne die Formularfelder einzeln übertragen zu müssen, um Zeit zu sparen und Übertragungsfehler zu vermeiden.
AK 2	1. Bei geöffneter Anmeldemail und geöffnetem AniTa-Tool müssen die Daten mit maximal 5 Aktionen in das AniTa-Tool überführt werden können. 2. Die Postleitzahlen des LDC und LDR werden automatisch auf Korrektheit geprüft. 3. Nutzer:innen bekommen eine Rückmeldung, bei der Eingabe inkorrekter Postleitzahlen. 4. Nutzer:innen haben die Möglichkeit die Anmeldedaten manuell zu ändern, bevor diese gespeichert werden.

Verwalten von Stammdaten

Die User Story 3, User Story 4 und User Story 5 beziehen sich auf die Verwaltung der Stammdaten der Teilnehmenden, konkret auf das Anzeigen, Verändern und Löschen dieser. Diese Operationen verfolgen Ziele, die nicht direkt in den Prozessdarstellungen abgebildet sind (siehe Abbildung 1 und Abbildung 2), jedoch bei Abweichungen von diesen Musterprozessen notwendig werden. Etwa wenn eine Anmeldung via Telefon erfolgt oder zusätzliche Daten in den Freitextfeldern des Anmeldeformulars angegeben wurden.

User Story 3: Anzeigen von Stammdaten

US 3	Als Projektmitarbeiter:in muss ich die Stammdaten der Teilnehmenden im AniTa-Tool einsehen können, um Informationen über die Teilnehmenden (z.B. zur Kontaktaufnahme) zu entnehmen.
AK 3	1. Nutzer:innen werden die Anmeldedaten der LDC und LDA in einem GUI angezeigt. 2. Nutzer:innen können die Anmeldedaten anhand der zugehörigen Attribute in aufsteigender und absteigender Reihenfolge sortieren.

User Story 4: Änderung von Stammdaten

US 4	Als Projektmitarbeiter:in muss ich die Stammdaten der Teilnehmenden manuell verändern können, um neue Informationen zu ergänzen oder Fehler zu korrigieren.
AK 4	1. Nutzer:innen können die Stammdaten der LDC und LDA über ein GUI verändern. 2. Nutzer:innen bekommen eine Rückmeldung bei der Eingabe inkorrekter Postleitzahlen. 3. Nutzer:innen können fehlerhafte Eingaben rückgängig machen.

User Story 5: Löschen von Stammdaten

US 5	Als Projektmitarbeiter:in muss ich Teilnehmende manuell löschen können, um auf Abmeldungen oder Inaktivität von Teilnehmenden zu reagieren.
AK 5	1. Nutzer:innen können über eine GUI Teilnehmende löschen. 2. Bei Löschung eines LDC werden auch die zugehörigen LDR entfernt, sollten diese nicht mit einem anderen LDR verbunden sein. 3. Nutzer:innen können fehlerhaftes Löschen von Teilnehmenden rückgängig machen.

Matching von Teilnehmenden

Das Kernziel der SP besteht im Matching der Teilnehmenden. Die manuelle Suche nach potenziellen Matches ist zugleich eine der zeitaufwändigsten und fehleranfälligsten Tätigkeiten im Betrieb der SP. Während diese Tätigkeit durch die User Story 6 adressiert wird, beziehen sich die User Story 7 und die User Story 8 auf die Anbahnung potenzieller Matches.

User Story 6: Automatisches Matching von Teilnehmenden

US 6	Als Projektmitarbeiter:in muss ich ein automatisches Matching der Teilnehmenden durchführen können, um alle potenziellen Tauschbeziehungen zu entdecken und gegenüber dem manuellen Matching Zeit zu sparen.
AK 6	1. Nutzer:innen können das automatische Matching über die GUI starten. 2. Als Nutzende kann ich auswählen, ob nur neue oder alle Anmeldungen beim Matching berücksichtigt werden. 3. Das Matching berücksichtig angrenzende Postleitzahlengebiete und den Aktivitätsradius des LDC. 4. Im Falle neuer potenzieller Matches erhalten Nutzer:innen eine Rückmeldung.

User Story 7: Anzeigen von Matches

US 7	Als Projektmitarbeiter:in muss ich alle Matches einsehen können, um Tauschbeziehungen anzubahnen.
AK 7	1. Nutzer:innen werden alle Matches in einer GUI angezeigt.

User Story 8: Änderung von Matches

US 8	Als Projektmitarbeiter:in muss ich den Status von Matches manuell verändern können, um den Status in der Anbahnung einer Tauschbeziehung zu dokumentieren.
AK 8	1. Nutzer:innen können den Status der Matches über ein GUI verändern.

Visualisierung von Anmeldungen

User Story 9 bezieht sich auf die Darstellung der teilnehmenden LDR und LDC auf einer digitalen Landkarte.

User Story 9: Geographische Visualisierung von Teilnehmenden

US 9	Als Projektmitarbeiter:in möchte ich die Teilnehmenden auf einer digitalen Karte visualisieren können, um einen Eindruck über die geographische Verteilung der Teilnehmenden zu bekommen.
AK 9	1. Nutzer:innen können über die GUI eine Karte generieren, auf der die Teilnehmenden dargestellt sind. 2. Auf der Karte werden die LDC und LDR in unterschiedlichen Farben dargestellt.

2.5.2 Nichtfunktionale Anforderungen

Unter dem Begriff der nichtfunktionalen Anforderungen werden häufig alle Anforderungen an ein Softwaresystem zusammengefasst, die nicht die Systemfunktionen betreffen. Hierbei handelt es sich zum Beispiel um Leistungsanforderungen, etwa Zeit- und Speichergrenzen, spezifische Qualitätsanforderungen wie Benutzbarkeit oder Wartbarkeit sowie Randbedingungen, etwa rechtlicher Art. (Balzert, 2009)

Die folgenden User Stories sind Unterabschnitten zugeordnet, die sich an den Qualitätsmerkmalen nach ISO/IEC 9126–1 orientieren, wie sie durch Balzert (2009) erläutert werden. Da sich nichtfunktionale Anforderungen in einigen Fällen schwieriger testen lassen als funktionale Anforderungen, sind die Akzeptanzkriterien der folgenden User Stories vielmehr als Leitlinie für die Entwicklung zu verstehen als dass Sie tatsächlich zur Abnahme überprüft werden können.

Portabilität

Ein Softwareprodukt weist eine hohe Portabilität auf, wenn dieses ohne größeren Anpassungsaufwand in andere Umgebung, z.B. Organisationen, Hardware oder Software übertragen werden kann (Balzert, 2009). Da unklar ist, welches Betriebssystem der Transferpartner verwendet, stellt dies eine wichtige Anforderung an das AniTa-Tool dar und wird in User Story 10 adressiert.

User Story 10: Portabilität des Anita-Tools

US 10	Als Transferpartner möchte ich die Anwendung auf verschiedenen Betriebssystemen nutzen können, damit ich direkt nach der Übernahme damit arbeiten kann.
AK 10	1. Die Anwendung ist auf Windows 10, macOS X und Ubuntu 18 ohne Einschränkungen in der Funktionalität lauffähig, ohne zuvor Änderungen am Quellcode vornehmen zu müssen.

Wartbarkeit

Wartbarkeit bezeichnet die Änderungsfähigkeit eines Softwaresystems im Sinne von Korrekturen, Verbesserungen oder Anpassungen (Balzert, 2009). Da sich die Anforderungen an das System, etwa aufgrund neuer Forschungsergebnisse, aber auch durch andere Geschäftsprozesse beim Transferpartner ändern können, ist es wichtig, dass das AniTa-Tool eine hohe Änderbarkeit aufweist. Die User Story 11 bezieht sich hierbei auf die Anpassung bestehender Funktionen

und Erweiterung um neue Funktionen. User Story 12 hingegen adressiert den Austausch gesamter Softwarekomponenten.

User Story 11: Änderbarkeit der Funktionalitäten des AniTa-Tools

US 11	Als Projektmitarbeiter:in und Transferpartner möchte ich die Anwendung durch neue Funktionalitäten erweitern können, um Sie an veränderte Anforderungen anpassen zu können.
AK 11	1. Die Entitäten können durch eine:n Programmierer:in mit Erfahrung im Technologiestack innerhalb von acht Arbeitsstunden durch neue Attribute erweitert werden. 2. Neue Anwendungsfälle können durch eine:n Programmierer:in mit Erfahrung im Technologiestack mit angemessenem Aufwand implementiert werden. 3. Es liegt eine Dokumentation der zentralen Module, Klassen und Funktionen der Anwendung vor.

User Story 12: Austausch von Komponenten

US 12	Als Transferpartner möchte ich die Möglichkeit haben, die Anwendung an meine präferierte Datenbank oder GUI anschließen zu können.
AK 12	1. Die Datenbank der Anwendung lässt sich durch eine:n erfahrene:n Programmierer:in innerhalb von einer Personenwoche austauschen. 2. Die GUI der Anwendung lässt sich durch eine:n erfahrene:n Programmierer:in innerhalb von einer Personenwoche austauschen.

Funktionalität

Funktionalität im Sinne einer Qualitätsanforderung beschreibt die Fähigkeit eines Softwaresystems Funktionen bereitzustellen, die festgelegte Bedürfnisse unter spezifizierten Bedingungen erfüllen (Balzert, 2009). Der Unteraspekt der Sicherheit wird in User Story 13 und User Story 14 berücksichtigt.

User Story 13: Übereinstimmung mit Datenschutzerklärung

US 13	Als Datenschutzbeauftragte:r der HAW möchte ich, dass die Anwendung den bestehenden Maßnahmen zum Datenschutz gerecht wird, um rechtlich abgesichert zu sein.
AK 13	1. Die Anwendung wird den vereinbarten technisch organisatorischen Maßnahmen gerecht. 2. Die Anwendung wird den im Verzeichnis der Verarbeitungstätigkeiten beschriebenen Tätigkeiten gerecht.

User Story 14: Schutz vor Datendiebstahl

US 14	Als Teilnehmende:r möchte ich, dass meine Daten nicht gestohlen werden können, damit meine Daten nicht zu anderen Zwecken als der SP verwendet werden.
AK 14	1. Die Daten der Teilnehmenden werden auf dem Netzwerklaufwerk der HAW gespeichert.

Zuverlässigkeit

Die Zuverlässigkeit eines Softwaresystems beschreibt die Fähigkeit ein festgelegtes Leistungsniveau zu bewahren (Balzert, 2009). Der Unteraspekt der Wiederherstellbarkeit in Bezug auf die Daten der Teilnehmenden wird in User Story 15 berücksichtigt.

User Story 15: Wiederherstellbarkeit der Stammdaten

US 15	Als Teilnehmende:r oder Projektmitarbeiter:in möchte ich, dass die Daten der Teilnehmenden nicht verloren gehen, damit die Funktionstüchtigkeit der Tauschbörse aufrechterhalten werden kann.
AK 15	1. Es wird mindestens täglich ein Backup der Daten der Teilnehmenden erstellt. 2. Der Datenbestand ist durch die Projektmitarbeiter:innen wiederherstellbar.

Benutzbarkeit

Unter Benutzbarkeit versteht man die Fähigkeit eines Softwaresystems von Anwender:innen verstanden und erlernt werden zu können (Balzert, 2009). Auf diese Aspekte bezieht sich die User Story 16.

User Story 16: Erlernbarkeit des AniTa-Tools

US 16	Als Projektmitarbeiter:in möchte ich, dass die Bedienung der Anwendung leicht zu erlernen ist, damit ich wenig Mehraufwand habe.
AK 16	1. Die Bedienung der Anwendung ist innerhalb von 15 Minuten unter Anleitung zu erlernen. 2. Es liegt ein Benutzerhandbuch vor, welches die zentralen Funktionalitäten und Bedienelemente erläutert.

3 Entwurf

Eine Softwarearchitektur beschreibt die Strukturen, Komponenten eines Systems sowie deren Schnittstellen und Beziehungen zueinander. Grundsätzlich kann die Architektur als Übergang von der Analyse zur Implementierung verstanden werden. Sie beschreibt, wie die Probleme, die sich durch die Anforderungen an das System ergeben unter Berücksichtigung der Rahmenbedingung gelöst werden sollen. Die Architektur dient zum einen als Grundlage für die Implementierung, wird allerdings ebenso auf Basis von Informationen, die sich durch die Implementierung ergeben, verfeinert und kann letztendlich auch als Form einer Dokumentation der Implementierung verstanden werden, welche jedoch von den Implementierungsdetails abstrahiert. (Starke, 2020a)

Die in diesem Kapitel dokumentierte Softwarearchitektur des AniTa-Tools ist Ergebnis mehrerer Iterationen aus Analyse der Anforderungen, Entwurf der Architektur, Implementierung und Evaluation. In den folgenden Abschnitten wird zunächst erläutert, was das AniTa-Tool ist und welche Rolle es im Kontext der bestehenden IT-Landschaft im Projekt einnimmt (Abschnitt 3.1) und ein Überblick über Architekturstil und -aufbau gegeben (Abschnitt 3.2). Anschließend werden die Fachdomänen- (Abschnitt 3.3), Präsentations- (Abschnitt 3.4) und Infrastrukturschicht (Abschnitt 3.5) sowie die Querschnittskonzepte (Abschnitt 3.6) erläutert. Hierbei werden neben den fachlichen und technischen Aspekten der Architektur auch die zentralen Designentscheidungen dargelegt sowie mögliche nächste Schritte der Weiterentwicklung der Architektur diskutiert.

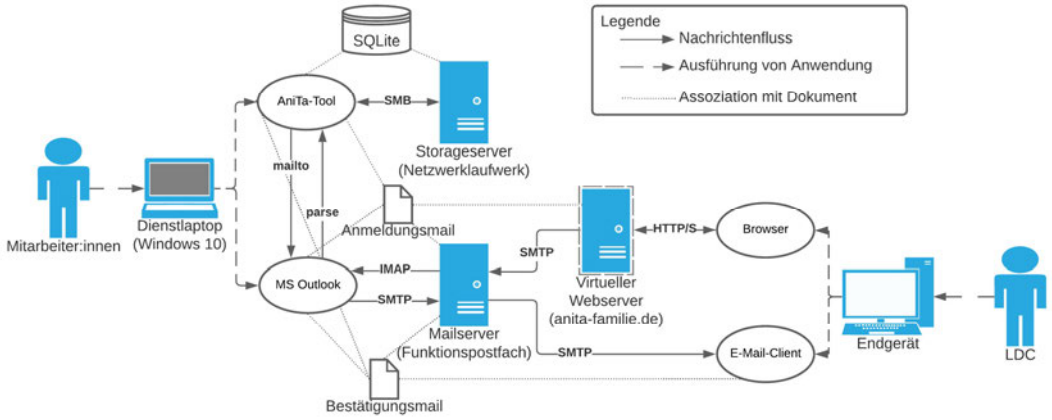
3.1 AniTa-Tool

Aufgrund der technischen sowie rechtlich-organisatorischen Rahmenbedingungen und der Notwendigkeit die Software an neue Forschungsergebnisse anpassen zu können (vgl. Abschnitte 2.2 und 2.3), wurde das AniTa-Tool als Individualsoftware konzipiert. Da keine

weiteren Plattformen zur Programmausführung zur Verfügung stehen, handelt es sich bei dem AniTa-Tool um eine Desktopanwendung, welche auf den Dienstlaptops der Projektmitarbeiter:innen ausgeführt wird. Mittels des AniTa-Tools lassen sich die Stammdaten der Teilnehmenden verwalten, das Matching der Teilnehmenden durchführen, Anmeldungen visualisieren und die Bearbeitung von Anmeldungen beschleunigen.

Wie in Abbildung 4 dargestellt ist, wird das Empfangen und Versenden von E-Mails hierbei weiterhin durch MS Outlook realisiert. Das AniTa-Tool ermöglicht es jedoch über einen Dialog die Stammdaten aus den Anmeldungsmails zu parsen und generiert aus den Stammdaten automatisch eine Bestätigungsmail, welche mittels des Uniform Resource Identifier Schemas ‚mailto' (Duerst et al., 2010) über MS Outlook versendet werden kann. Eine direkte Anbindung des AniTa-Tools an das Funktionspostfach via IMAP bzw. SMTP ist zwar technisch möglich, ist jedoch mit hohem Aufwand verbunden. Die Anforderung, dass die E-Mails individualisierbar sein sollten (vgl. User Story 1), hätte etwa die Implementierung einer Vielzahl von Funktionen notwendig gemacht, um annähernd den Funktionsumfang von MS Outlook zu erreichen. Da die Daten in der Anmeldemail clientseitig nur geringfügig geprüft werden und ein Single-Opt-In Verfahren genutzt wird, ist es zudem ratsam, die Daten vor der Weiterverarbeitung im AniTa-Tool manuell zu prüfen. Hierdurch können nur solche Anmeldungen in das System eingegeben werden, welche dem Augenschein nach valide sind. Ein direktes Abrufen der E-Mails via IMAP hätte auch die Möglichkeit einer solchen Prüfung durch das Anita-Tool notwendig gemacht.

Abbildung 4: Darstellung des AniTa-Tools im Kontext der Softwaresysteme und Kommunikationsprotokolle im Projekt AniTa



Da der einzige zentrale Ort in der technischen Infrastruktur der Storageserver darstellt, kam keine Datenbank mit Serverprozess in Frage. Stattdessen fiel die Entscheidung zugunsten der SQL-Datenbank-Engine SQLite, welche direkt auf eine Datei auf dem Netzwerklaufwerk schreibt (The SQLite Consortium, 2021). Hierdurch kann sichergestellt werden, dass alle Projektmitarbeiter:innen Zugriff auf den gleichen Datenbestand haben.

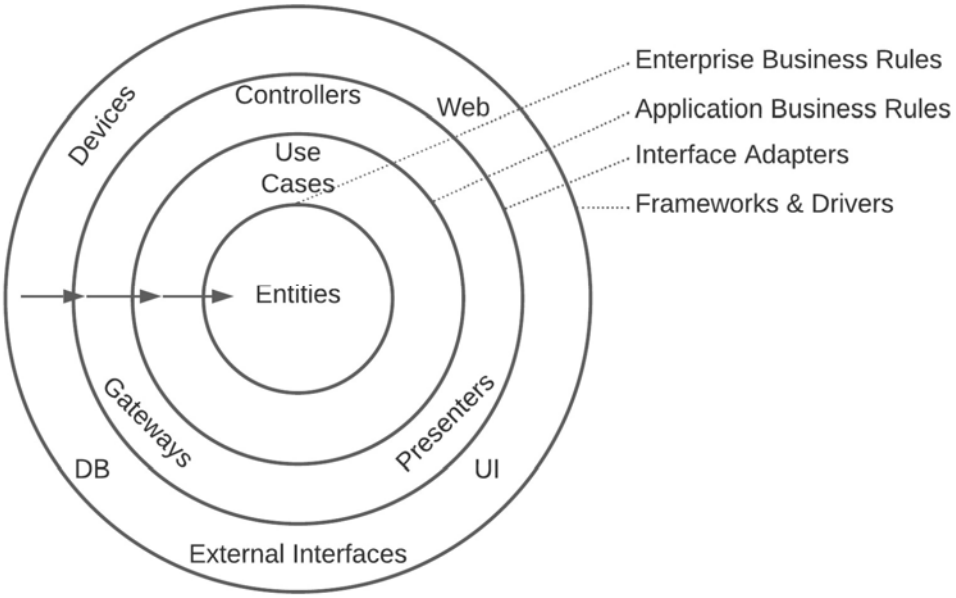
Das AniTa-Tool selbst basiert überwiegend auf der dynamisch typisierten, interpretierten Allzweck-Programmiersprache Python 3.8 (Python Software Foundation, 2021) sowie Bibliotheken und Frameworks des Python-Ökosystems. Die Entscheidung für Python als zentrale Programmiersprache ist durch mehrerer ihrer Eigenschaften begründet. Zum einen erlaubt die gute Lesbarkeit von Python ein schnelles Verständnis des Quellcodes, was die Wartbarkeit auch durch externe Entwickler:innen erhöhen kann (vgl. User Story 11). Zum anderen zeichnen sich Python-Programme durch eine hohe Portabilität (vgl. User Story 10) aus, da Python verschiedene Integrationsmechanismen unterstützt und Python-Programme somit in der Regel direkt auf verschiedenen Plattformen lauffähig sind. (Lutz, 2013)

3.2 Architekturüberblick

Die Software-Architektur des AniTa-Tools orientiert sich an dem Architekturmuster der Clean-Architecture (CA), welche mit anderen Architekturmustern etwa der Hexagonal Architecture oder der Onion Architecture verwandt ist. Bei allen dieser Muster soll durch eine Aufteilung

der Software in Schichten, welche über definierte Schnittstellen miteinander kommunizieren das Design Prinzip separation of concerncs (SoC) umgesetzt werden. Hierdurch sollen letztendlich Systeme entstehen, welche testbar und unabhängig von spezifischen Frameworks, User Interfaces, Datenbanken sowie sonstiger externer Agenten sind. Der zentrale Unterschied zu klassischen Schichtenarchitekturen liegt bei der CA in der sogenannten Dependency Rule, welche vorschreibt, dass Quellcodeabhängigkeiten nur nach innen, in Richtung der Objekte der Fachdomäne, zeigen dürfen. Um dieses Prinzip zu verdeutlichen, wird die CA in der Regel als Schalenmodell dargestellt (siehe Abbildung 5). (Martin, 2012)

Abbildung 5: Darstellung der Clean Architecture nach Martin (2012)



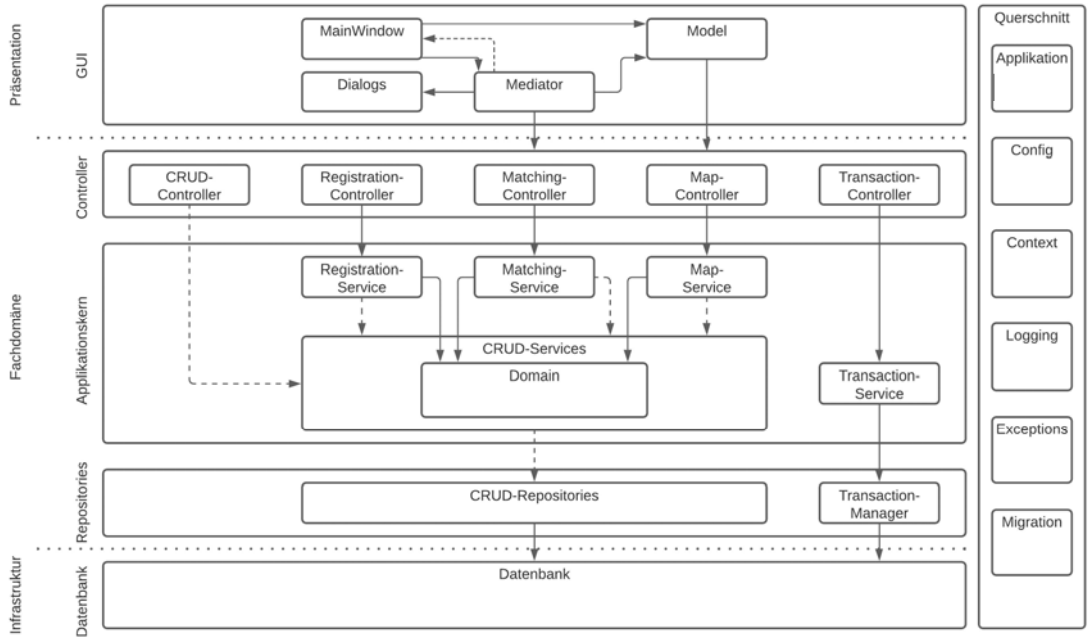
Anmerkung. Die Pfeile in der Abbildung symbolisieren die Verlaufsrichtung der Quellcode-Abhängigkeiten.

Den inneren Kern der CA bilden die Geschäftsobjekte der Fachdomäne, die sogenannten Entitäten, welche allgemeine Geschäftsregeln implementieren. Die in den Entitäten implementierten Geschäftsregeln werden von Use Cases genutzt, welche applikationsspezifische Geschäftsregeln implementieren. Schnittstellen-Adapter übersetzen die Daten vom Format der Use Cases in das Format der externen Frameworks und Treiber, etwa der Datenbank oder des User

Interfaces, und umgekehrt. In Fällen, in denen ein Use Case eine externe Technologie aufrufen muss, etwa um Daten aus der Datenbank zu lesen, wird das Dependency Inversion Principle (DIP) angewendet, um nicht gegen die Dependency Rule zu verstoßen. (Martin, 2012) Damit die Use Cases keine Abhängigkeiten zu Implementierungsdetails äußerer Schichten aufweisen, werden nach dem DIP häufig Interfaces zwischen Use Case und externer Technologie genutzt. Hierbei wird ein Interface definiert, was dem Abstraktionsniveau und der Domäne der Use Cases angemessen ist. Im Falle eines Datenbankaufrufes mittels SQL-Query durch einen Use Case, könnte etwa ein Repository genutzt werden, welches eine definierte Schnittstelle für den Use Case bereitstellt und so von der konkreten Datenbanktechnologie abstrahiert. (Schuchert, 2013)

Die Architektur des AniTa-Tools gliedert sich auf der obersten Ebene in die Fachdomänenschicht, Präsentationsschicht, Infrastrukturschicht und Querschnittskonzepte, welche nicht einer Ebene zugeordnet werden können (siehe Abbildung 6).

Abbildung 6: Darstellung der Softwarearchitektur des AniTa-Tools als Bausteinsicht



Anmerkung. Gestrichelte Linien symbolisieren Schichten, Kästen symbolisieren Komponenten und Subkomponenten; Pfeile symbolisieren Abhängigkeiten zwischen Komponenten, gestrichelte Pfeillinien symbolisieren Abhängigkeiten von einem Interface.

Die Präsentationsschicht besteht aus einem Graphical User Interface (GUI), über welches die Nutzer:innen das AniTa-Tool bedienen. Über die Schnittstellen, welche die Controller anbieten, können mittels des GUI die Funktionen der Services im Applikationskern genutzt werden. Die Repositories bieten dem Applikationskern Schnittstellen, um die Daten in der Datenbank der Infrastrukturschicht zu persistieren, ohne gegen die Dependency Rule zu verstoßen. Während Controller und Repositories Interface Adapter darstellen, liegen sowohl Application als auch Enterprise Business Rules im Applikationskern des AniTa-Tools.

3.3 Fachdomänenschicht

Die Fachdomänenschicht des AniTa-Tools gliedert sich in den Applikationskern, sowie die Controller und Repositories. In Bezug auf die CA umfasst der Applikationskern die Entitäten sowie Use Cases. Die Controller und Repositories repräsentieren die Schnittstellen-Adapter.

Abweichend von der in Abbildung 5 dargestellten Architektur, bilden die CRUD-Services in der Architektur des AniTa-Tools einen zusätzlichen Ring um die Entitäten der Fachdomäne, hier Domain (siehe Abschnitt 3.3.1). Die CRUD-Services verantworten hierbei die Modellierung der Geschäftsregeln, welche sich durch die Beziehungen der Entitäten untereinander ergeben und stellen die grundlegenden Operationen Create, Read, Update sowie Delete auf den Entitäten bereit (siehe Abschnitt 3.3.2). Während die CRUD-Services zum einen direkt zur Verwaltung der Stammdaten genutzt werden können, nutzen die übrigen Services die Funktionen der CRUD-Services, um die weiteren funktionalen Anforderungen zu erfüllen, wie sie in Abschnitt 2.5.1 beschrieben sind. Hierbei verantwortet der Registration Service (siehe Abschnitt 3.3.4) die Bearbeitung von Anmeldungen, der Matching-Service (siehe Abschnitt 3.3.6) das Matching der Teilnehmenden und der Map-Service (siehe Abschnitt 3.3.7) die Visualisierung der Anmeldungen. Der Transaction-Service nimmt wiederum eine Sonderrolle ein, da er Dienste bereitstellt, die den Zustand aller Entitäten zur Laufzeit betreffen (siehe Abschnitt 3.3.3). Während die GUI die jeweiligen Controller verwendet, um die Services zu nutzen (siehe Abschnitt 3.3.8.), verwenden die CRUD-Services die Repositories, um Operationen auf der Datenbank durchzuführen (siehe Abschnitt 3.3.3).

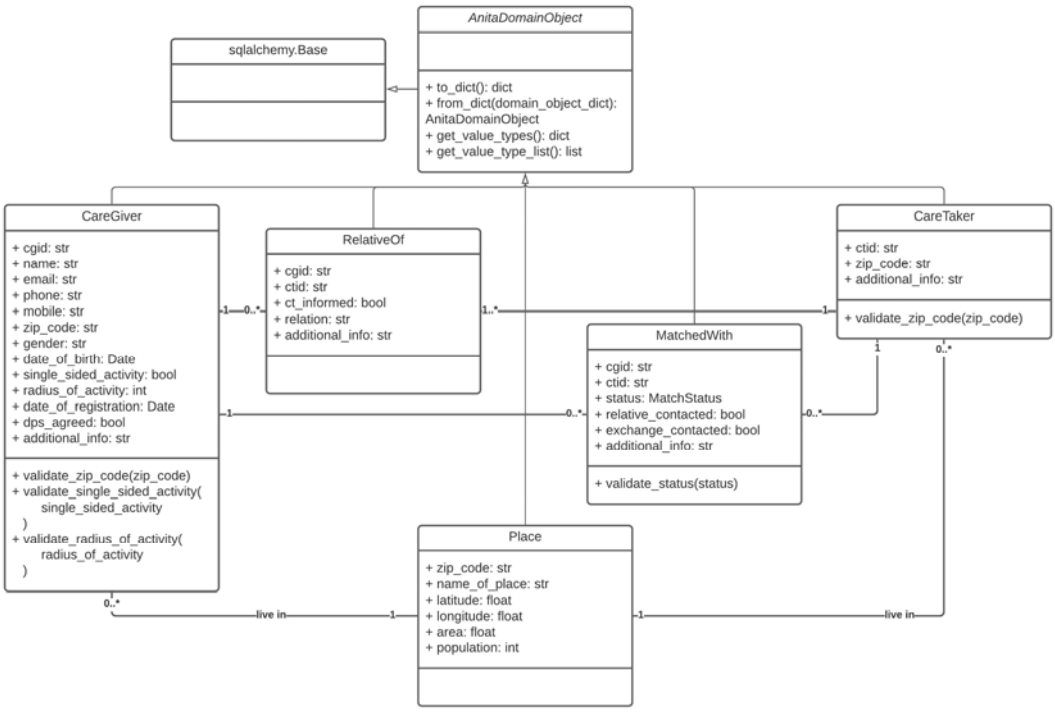
3.3.1 Domain

Die Fachdomäne wird durch die Entitäten CareGiver (CG), CareTaker (CT), RelativeOf (RO), MatchedWith (MW) und Place modelliert. Hierbei repräsentiert die Entität CareGiver die LDC und CareTaker die LDR. CareGiver und CareTaker können im Rahmen von AniTa über zwei verschiedene Arten in Beziehung zueinanderstehen. Sie können miteinander verwandt sein, was durch die Entität RelativeOf abgebildet wird oder es kann eine Tauschbeziehung – ein Match – zwischen ihnen bestehen, was durch die Entität MatchedWith abgebildet wird. Da sich einige LDC auf der Plattform anmelden, ohne selbst Unterstützung für einen LDR zu suchen und andere mehrere LDR anmelden, gilt, dass ein CareGiver mit keinem, einem oder mehreren CareTakern verwandt sein kann. Ein CareTaker jedoch muss mit mindestens einem CareGiver verwandt sein, da sich die LDR nicht selbst auf der Plattform anmelden können. Auch bei den Tauschbeziehungen muss berücksichtigt werden, dass sowohl ein CareGiver mit keinem, einem oder mehreren CareTakern gematcht sein kann und umgekehrt. Zum einen kann ein LDC mehrere LDR unterstützen, zum anderen können sich auch mehrere LDC die Unterstützung

eines LDR teilen. Zudem müssen nicht alle Tauschbeziehungen aktiv sein, sondern können sich z.B. auch in einem potenziellen oder inaktiven Status befinden.

Das auch die Beziehungen zwischen den CareGivern und CareTakern als Entitäten modelliert werden, ist fachlich dadurch begründet, dass es eben diese Beziehungen sind, die den Kern der Tauschbörse ausmachen. Auf technischer Eben wird zudem ohnehin ein weiteres Objekt bzw. eine Tabelle für die Modellierung der N:M-Beziehungen benötigt, zumal einige für den Betrieb notwendige Informationen nur der Ebene der Beziehungen sinnvoll zugeordnet werden können. Hierbei handelt es sich um die Informationen, welcher LDC welchen seiner LDR über die Teilnahme an der Tauschbörse informiert hat oder welchen Status eine Tauschbeziehung hat. Da zudem die Postleitzahlengebiete der LDC und LDR eine entscheidende Rolle im Rahmen des Matchings spielen, werden auch diese als die Entität Place modelliert. Alle konkreten Entitäten werden als Klassen mit spezifischen Attributen und Methoden modelliert. Einen Überblick über das Modell der Fachdomäne gibt Abbildung 7.

Abbildung 7: Übersicht über die Entitäten der Fachdomäne ihrer Relationen und Vererbungsbeziehungen als UML-Klassendiagramm



Die Entität CareGiver verfügt neben ihrer UID (cgid) über mehrere Attribute. Einige repräsentieren persönliche Informationen wie den Namen (name), das Geburstdatum bzw. -jahr (date_of_birth), das Geschlecht (gender) oder Kontaktinformationen (email, phone, mobile). Andere dienen administrativen Zwecken, halten etwa fest ob der Datenschutzvereinbarung zugestimmt wurde (dps_agreed) oder erlauben das Anlegen zusätzlicher Informationen (additional_info). Hinzu kommen die Informationen, welche für das automatisierte Matching von Relevanz sind: nämlich ob der LDC einseitig tätig wird, d.h. selbst keine Unterstützung sucht (single_sided_activity), in welchem Postleitzahlengebiet er wohnt (zip_code) und in welchem Radius er Hilfe anbietet, bzw. aktiv werden kann (radius_of_activity). Da diese Attribute eine besondere Rolle im automatisierten Matching spielen, liegen Methoden zur Validierung der Werte vor (validate_zip_code, validate_single_sided_activity, validate_radius_of_activity).

Die CareTaker werden nur durch ihre UID (cgid), ein Feld für zusätzliche Informationen (additional_info) und ihre Postleitzahl (zip_code) beschrieben, für welche wiederum eine Methode zur Validierung (validate_zip_code) vorliegt.

Neben den beiden UID (cgid, ctid) und den zusätzlichen Informationen (additional_info) verfügt die Beziehungs-Entität RelativeOf über ein Feld zur Beschreibung des Verwandtschaftsverhältnisses (relation) und eines, um zu hinterlegen, ob der assoziierte LDR vom LDC über die Teilnahme informiert wurde.

Die zweite Beziehungs-Entität MatchedWith verfügt ebenfalls über die beiden UID (cgid, ctid) und das Feld für zusätzliche Informationen (additional_info). Darüber hinaus beschreiben zwei Attribute, ob die Angehörigen des LDR über die Tauschmöglichkeit informiert wurden (relative_contacted) bzw. ob der potenzielle LDC zum Tausch kontaktiert wurde (exchange_contacted). Über das weitere Attribut status kann zudem festgehalten werden, in welchem Status sich ein Match befindet. Hierzu wird der fachliche Datentyp MatchStatus verwendet, der in Tabelle 4 genauer beschrieben ist.

Tabelle 4: Beschreibung des fachlichen Datentyps MatchStatus

Ausprägung	***Beschreibung***
awaiting_response	Die möglichen Tauschpartner:innen sind informiert, haben sich aber noch nicht zurückgemeldet.
potential	Die Grundvoraussetzung für eine Tauschbeziehung sind gegeben, aber die Tauschpartner:innen sind noch nicht informiert.
active	Die Tauschbeziehung ist aktiv.
denied	Einer der Tauschpartner:innen hat die Tauschbeziehung abgelehnt, bevor diese aktiv war.
cancelled	Die Tauschbeziehung war aktiv, wurde aber abgebrochen.

Bei der UID der Entität Place handelt es sich um die Postleitzahl (zip_code). Zudem werden ein möglicher Ortsname (name_of_place), die Flächengröße in Quadratkilometern (area) und die absolute Bevölkerungsgröße (population) hinterlegt. Über den Breiten- (latitude) und Längengrad (longitude) wird darüber hinaus ein zentraler Ort im jeweiligen Postleitzahlengebiet beschrieben.

Dadurch, dass alle Entitäten von der abstrakten Klasse AnitaDomainObject ableiten, erben sie die Implementierung der Hilfsmethoden to_dict, from_dict, get_value_types und get_value_type_list. Mit den Methoden to_dict und from_dict, lassen sich Entitäts-Python-Objekte in

Python-Dictionaries und zurück konvertieren. Diese Methoden können zur Konvertierung auf Ebene der Controller verwendet werden. Hierdurch müssen auf Ebene der Controller keine Anpassungen vorgenommen werden, wenn einer Entität neue Attribute hinzugefügt werden. Ähnliche Vorteile bieten die Methoden get_value_types und get_value_type_list, welche ein Python-Dictionary mit Attributnamen als Schlüsseln und dem assoziierten Datentyp als Wert, bzw. eine Liste der Datentypen zurückgeben. Die Informationen über die Datentypen können ebenfalls auf der Ebene der Controller genutzt werden, um die Daten, die aus der GUI übermittelt werden in die geforderten Datentypen zu konvertieren.

Da die Klasse AniTaDomainObject von der Klasse Base des Object Relational Mappers (ORM) SQLAlchemy (Bayer, 2012) ableitet und somit eine Abhängigkeit zu einem externen Framework erzeugt wird, findet eine Verletzung der Dependency Rule statt. Diese Verletzung wurde eingegangen, da SQLAlchemy plattformunabhängig ist und eine große Anzahl an Datenbanken unterstützt und es somit unwahrscheinlich ist, dass das ORM ausgetauscht werden muss. Zudem liegen in der Nutzung von SQLAlchemy viele Vorteile, die insbesondere in der Abstraktion von der konkreten Datenbank und der Fülle an angebotenen Features zu sehen ist.

Exkurs: Object Relational Mapping mit SQLAlchemy

Durch die Vererbungsbeziehung zur SQLAlchemy-Base-Klasse lässt sich innerhalb der Klassendefinition der Entitäten definieren, auf welche Tabelle in der Datenbank diese gemappt werden und welche Spalten mit welchen Datentypen diese Tabelle beinhaltet. Zudem lassen sich Beziehungen zu anderen Entitäten definieren. Ein Objekt einer Entitäts-Klasse kann so über seine jeweilige UID einer Zeile in der entsprechenden Tabelle zugeordnet werden. Innerhalb der Applikation lassen sich die Objekte dann wie Python-Objekte erstellen, verwenden und verändern. Die Änderungen an den Objekten werden durch SQLAlchemy automatisch in den Datenbanktabellen synchronisiert. (SQLAlchemy authors and contributors, n.d.)

Die Kommunikation zur Datenbank erfolgt über die SQLAlchemy-Session. Nach der Erzeugung ist ein Session-Objekt zunächst zustandslos und kann dazu genutzt werden Query-Objekte zu erzeugen, mittels derer Datenbankabfragen durchgeführt werden können. Hierzu benötigt die Session ein Engine-Objekt, welches die Verbindung zur Datenbank herstellt. Diese Verbindung repräsentiert eine Transaktion, die bestehen bleibt, bis ein Commit oder Roll-Back

der Session durchgeführt wird. Basierend auf dem Ergebnis der Query werden aus den Tabellenzeilen Objekte erstellt, die in der sogenannten Identity-Map verwaltet werden. Die Identity-Map ist eine Datenstruktur, die immer genau ein Objekt mit einem einzigartigen Primärschlüssel beinhaltet. Die Session verwaltet alle Änderungen an den Objekten so lange bis durch eine erneute Datenbankabfrage oder ein Commit der Session die Änderungen in die Datenbank „gespült“ werden. Zudem können innerhalb einer Session geschachtelte Transaktionen gestartet werden, für die wiederum separat ein Commit oder Roll-Back durchgeführt werden kann, ohne dass der Zustand vor der Transaktion verändert wird. (SQLAlchemy authors and contributors, n.d.)

3.3.2 CRUD-Services

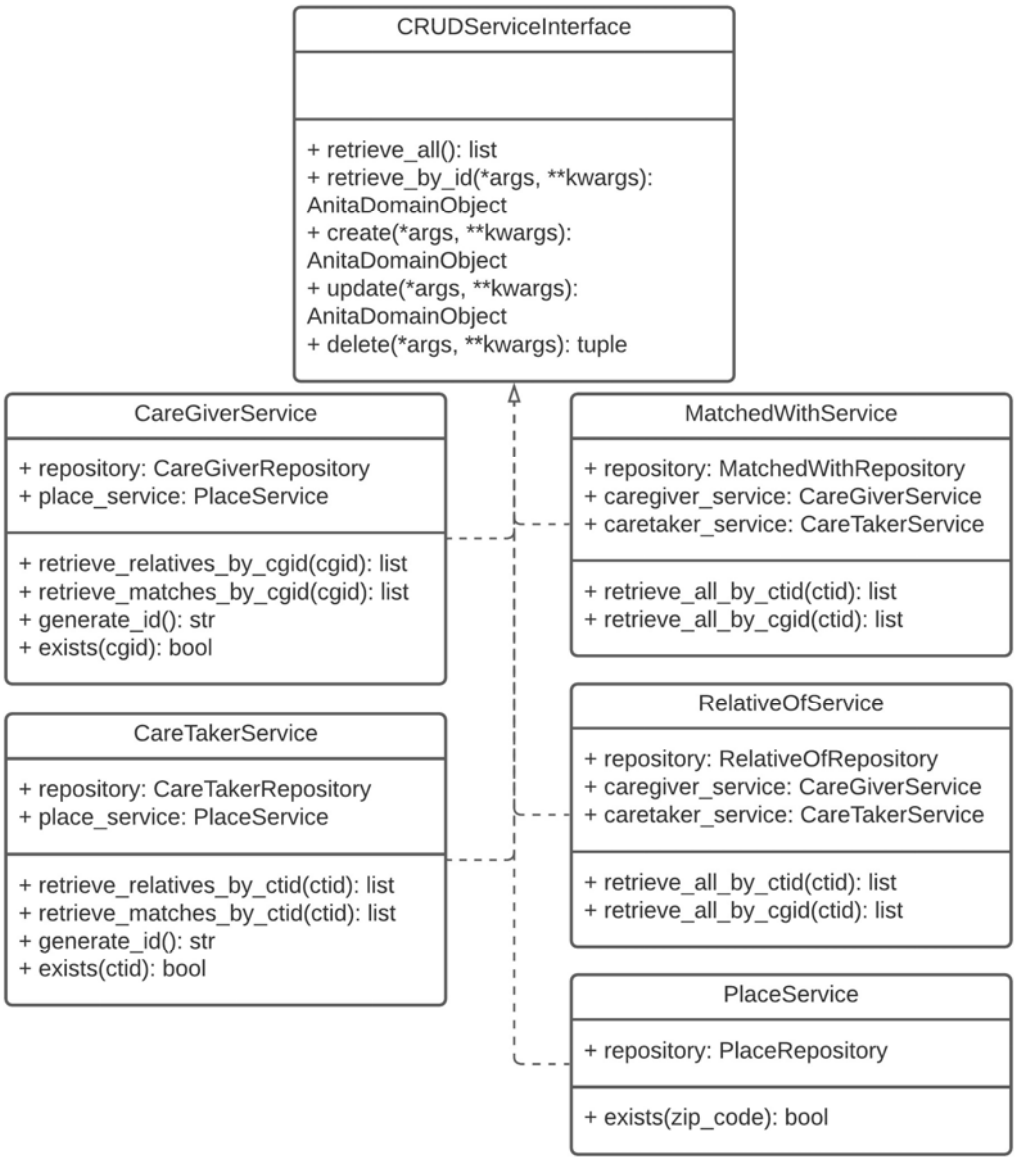
Wie das Akronym CRUD nahelegt, liegt die Verantwortung der CRUD-Services darin, die Operationen create, retrieve, update und delete für die Entitäten der Domain bereitzustellen. Die CRUD-Services ermöglichen es also, Entitäten zu erstellen (create), bestehende Entitäten abzurufen (retrieve) oder zu verändern (update) sowie diese zu löschen (delete). Hierbei berücksichtigen die CRUD-Services Geschäftsregeln, die nicht auf der Ebene einzelner Entitäten modelliert werden können. Die CRUD-Services bieten den CRUD-Controllern eine Schnittstelle, um die CRUD-Operationen zu nutzen und sind daher eine zentrale Komponente, um die Anforderungen der User Storys zur Verwaltung der Stammdaten zu erfüllen (s. User Story 3, User Story 4, User Story 5). Zudem werden die CRUD-Services von den übrigen Services genutzt, um auf die Entitäten zuzugreifen, diese anzulegen und zu verändern.

Für jede der in Abbildung 7 dargestellten Entitäten existiert ein CRUD-Service, welcher das CRUDServiceInterface implementiert (siehe Abbildung 8). Das CRUDServiceInterface definiert die grundlegenden CRUD-Operationen, welche von den jeweiligen konkreten CRUD-Services implementiert werden. Für den Zugriff auf die Datenbank nutzen die CRUD-Services wiederum ein entsprechendes Repository. Darüber hinaus stellen die einzelnen CRUD-Services zusätzliche Methoden bereit. Bis auf den PlaceService bieten alle CRUD-Services die Möglichkeit alle Matches bzw. Verwandtschaftsbeziehungen eines LDC bzw. LDR abzurufen. CareGiverService und CareTakerService nutzen die Methode generate_id(), um eine UID für die jeweilige Entität zu erzeugen und mittels ihrer Methode exists(uid) können der

MatchedWithService und RelativeOfService überprüfen, ob ein CareGiver bzw. CareTaker mit gegebener UID existiert, bevor eine Beziehungsentität angelegt wird. CareGiverService und CareTakerService nutzen wiederum die exists(zip_code) Methode des PlaceService, bevor eine Postleitzahl in einer der Entitäten angelegt wird.

Diese Prüfungen mittels der jeweiligen exists-Methode entsprechen der Herstellung eines Foreign-Key-Constraints, welcher ebenso auf Ebene der Datenbank hergestellt werden könnte. Durch die Handhabung der Constraints innerhalb des Applikationskerns, wird jedoch verhindert, dass Geschäftslogik in die Infrastrukturschicht eindringt und somit starke Abhängigkeiten zur jeweiligen Datenbanklösung entstehen.

Abbildung 8: Übersicht der CRUDServices als UML-Klassendiagramm



3.3.3 Transaction-Service

Die Aufgabe des Transaction-Services ist es innerhalb des Applikationskerns eine Schnittstelle zum Transaction-Manager bereitzustellen, um Informationen über den aktuellen Zustand der Session zu erhalten, die aktuelle Session zu speichern bzw. zurückzusetzen oder eine

geschachtelte Transaktion zu starten. Die Schnittstellen, die durch die Klasse TransactionService in Form von Methoden für den TransactionController bereitgestellt werden, sind in Tabelle 5 dargestellt.

Tabelle 5: Beschreibung der Methoden des TransactionServices

Methode	***Beschreibung***
save()	Speichert aktuellen Zustand der Entitäten in der Datenbank.
reset()	Verwirft ungespeicherte Änderungen der Entitäten.
get_changes()	Gibt alle geänderten, neuen, und gelöschten Entitäten zurück.
get_number_of_changes()	Gibt die Anzahl der ungespeicherten Änderungen zurück.
is_session_dirty()	Gibt an, ob ungespeicherte Änderungen vorliegen.
start_transaction()	Startet eine geschachtelte Transaktion.

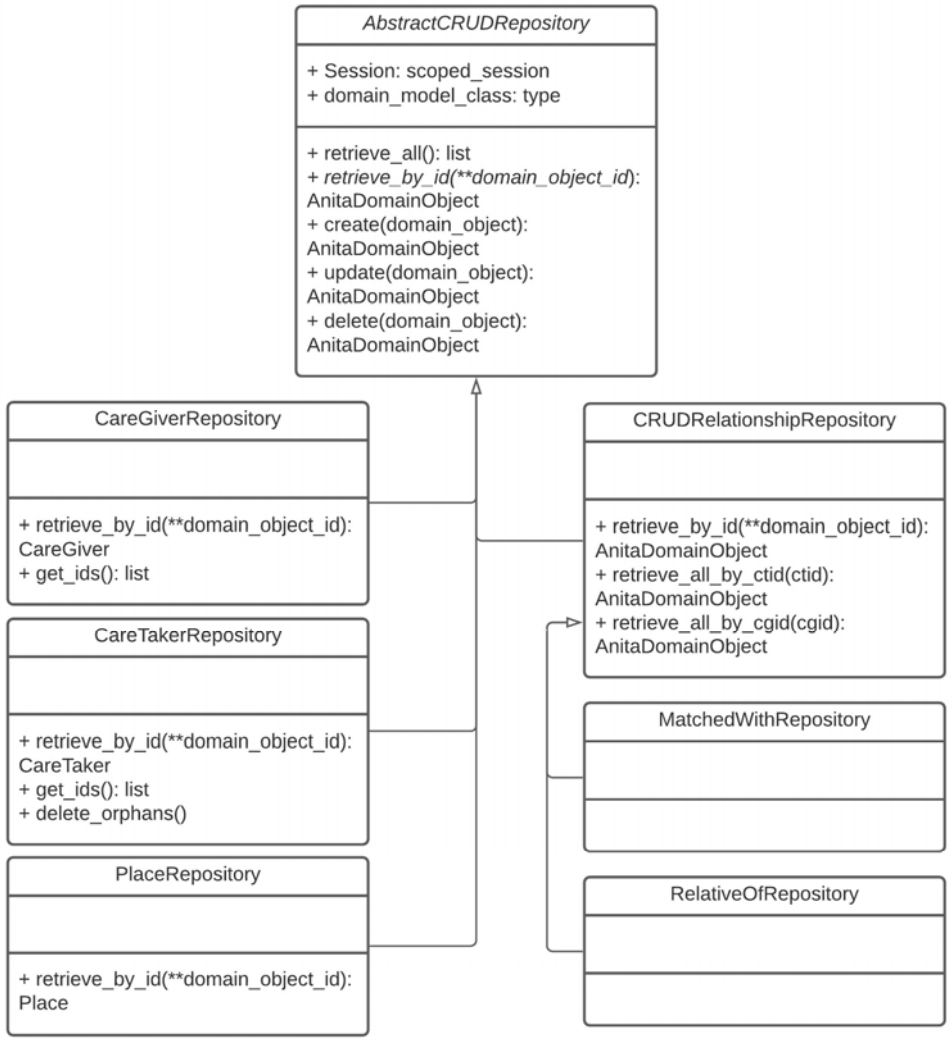
Dadurch, dass Änderungen explizit gespeichert oder zurückgesetzt werden können, erhalten Nutzer:innen die Möglichkeit fälschliches Löschen von Entitäten zu korrigieren (siehe User Story 5).

3.3.4 Repositories

Die Repositories stellen die Schnittstellen-Adapter zur Persistenzschicht bzw. Datenbank dar und nutzen die SQLAlchemy-Session. Sie umfassen die CRUD-Repositories und den Transaction-Manager. Während die CRUD-Repositories von den CRUD-Services genutzt werden, um auf die Datenbank zuzugreifen, verwendet der Transaction-Service den Transaction-Manager, um Informationen über die aktuelle Session abzurufen, eine verschachtelte Transaktion zu starten oder Änderungen zu speichern bzw. zurückzusetzen.

Wie in Abbildung 9 dargestellt ist, steht an oberster Stelle in der Vererbungshierarchie der Repositories das AbstractCRUDRepository, welches als Attribute eine SQLAlchemy-Session und den Datentyp der jeweiligen Entität festlegt und bereits für die meisten Methoden die Implementierung bereitstellt. Die meisten Methoden spiegeln hierbei die Methoden der in Abschnitt 3.3.2 beschriebenen CRUD-Services wider.

Abbildung 9: Übersicht über die CRUD-Repositories als UML-Klassendiagramm



Eine Besonderheit stellt die Methode delete_orphans() des CareTakerRepositories dar. Durch Aufruf dieser Methode werden alle CareTaker gelöscht, die weder über eine MatchedWith-Entität noch über RelativeOf-Entität mit einem CareGiver assoziiert sind. Hierdurch soll verhindert werden, dass LDR im Matchingprozess berücksichtigt werden, die nicht mehr über ihre LDC kontaktiert werden können. Aufgerufen wird die Methode immer dann, wenn ein CareGiver gelöscht wird. Der Aufruf erfolgt eventgetrieben durch Nutzung der SQLAlchemy event-API.

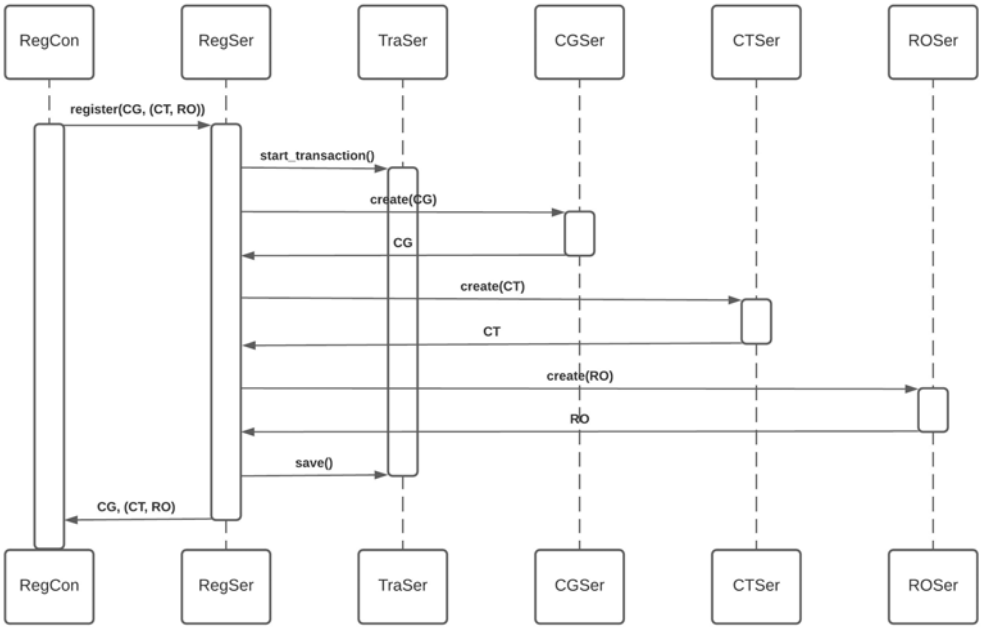
Die Methoden der Klasse TransactionManager entsprechen den Methoden der Klasse TransactionService, welche in Tabelle 5 dargestellt sind, mit dem Unterschied, dass diese die entsprechenden Funktionalitäten der SQLAlchemy-Session aufrufen.

3.3.5 Registration-Service

Der Registration-Service ist die Komponente, welche primär für die Bearbeitung von Anmeldungen und der Umsetzung der User Story 1 und User Story 2 verantwortlich ist. Hierzu stellt der Registration-Service zwei Schnittstellen bereit: Die Methode register und die Methode send_mail.

Die Methode register behandelt die beiden Standardfälle der Neuanmeldungen bei der SP, welche vom Anmeldeformular abgedeckt werden. Nämlich die Anmeldung eines einzelnen LDC, der lediglich Unterstützung anbietet oder die Anmeldung eines LDC und eines LDR, für den Unterstützung gesucht wird. Sie verfügt daher über den Pflichtparameter caregiver und die beiden optionalen Parameter caretaker und relative_of. Wie in Abbildung 10 zu sehen ist, startet der RegistrationService nach Aufruf der Methode register durch den RegistrationController eine geschachtelte Transaktion und nutzt anschließend die CRUD-Services der entsprechenden Entitäten, um diese auf Einhaltung der Geschäftsregeln zu prüfen und ggf. der Session hinzuzufügen. Ist die Prüfung für alle Entitäten erfolgreich, wird ein Commit der geschachtelten Session durchgeführt, andernfalls wird diese zurückgesetzt und der Nutzer erhält die Möglichkeit die Angaben zu korrigieren.

Abbildung 10: UML-Sequenzdiagramm der Registration-Service-Methode register



Anmerkung. RegCon = RegistrationController, RegSer = RegistrationService, TraSer = TransactionService, CGSer = CareGiverService, CTSer = CareTakerService, ROSer = RelativeOfService

Die Methode send_mail nimmt als Parameter eine CareGiver-Entität entgegen und nutzt eine Instanz der Klasse MailGenerator, um basierend auf den Stammdaten des LDC und einem E-Mail-Template eine Bestätigungsmail zu erstellen. Hierzu stehen Templates für die Anmeldungen mit LDR und ohne LDR zur Verfügung. Diese Bestätigungsmail wird in Form einer mailto-URL an den RegistrationController zurückgegeben, wodurch diese im Standardmailprogramm des ausführenden Systems geöffnet wird.

Dadurch, dass die mailto-URL auf Ebene des Registration-Service erstellt wird, erfolgt eine Verletzung der Dependency Rule, da an dieser Stelle innerhalb des Applikationskerns angenommen wird, dass eine Standard-E-Mail-Applikation installiert ist und angesprochen werden kann. Diese Verletzung kann in einer nächsten Iteration dadurch behoben werden, dass der

RegistrationService lediglich den E-Mail-Text generiert und die mailto-URL auf Ebene der Controller erstellt wird.

3.3.6 Matching-Service

Die Aufgabe des Matching-Service liegt im automatischen Matching der Teilnehmenden, welches durch die User Story 6 adressiert wird. Hierzu stellt der MatchingService die zwei Methoden get_potential_matches und persist_matches als Schnittstellen bereit. Die Methode persist_matches nimmt eine Liste von MatchedWith-Objekten und persistiert diese mittels des MatchedWithService. Das automatische Matching hingegen wird durch die Methode get_potential_matches gestartet, welche die Parameter since_date und deviation entgegennimmt. Übergibt man für den Parameter since_date ein Datum werden nur Anmeldungen berücksichtig, die seit diesem Datum erfolgt sind. Der Parameter deviation spielt in der Modellierung des Matchings eine Rolle, welche im Folgenden beschrieben wird. Die folgenden Ausführungen basieren hierbei zu einem großen Teil auf dem Unterkapitel „Vorläufiger und automatisierter Matchingprozess“ des Abschlussberichts des Projekts AniTa (Woock et al., 2020).

Da sich das Matching von n CareGivern zu m CareTakern als Maximum-Matching-Problem für bipartite Graphen, einer Form der Netzwerkflussprobleme, modellieren lässt, lässt sich der Ablauf des Matching-Algorithmus in zwei Schritte unterteilen (Tarjan, 1983).

Im ersten Schritt wird ein bipartiter Graph konstruiert. Dieser Graph G sei definiert als eine Menge von Knoten V und eine Menge von Kanten E, wobei eine Kante durch ein Tupel zweier unterschiedlicher Knoten repräsentiert wird. Die Knotenmenge besteht aus der Menge aller CareGiver und CareTaker und die Kantenmenge stellt die möglichen CareGiver-CareTaker-Paarungen dar. Welche Paarungen möglich sind, wird durch die Prüffunktion p bestimmt und die Gewichtungsfunktion w ordnet jeder Kante eine positive reelle Zahl zu. Dieses Kantengewicht ist so zu interpretieren, dass ein höheres Gewicht eine bessere Passung der entsprechenden CareGiver und CareTaker repräsentiert. Eine genauere Beschreibung, wie der bipartite Graph konstruiert wird und wie die Prüf- und Gewichtungsfunktion definiert sind, findet sich im Unterabschnitt „Konstruktion des Graphen“ wieder.

Im zweiten Schritt wird die Anzahl der Kanten reduziert. Hierzu wird ein Matching M auf dem Graphen G berechnet. M ist hierbei eine Menge von Kanten, in der sich keine zwei Kanten einen Knoten teilen. Ziel ist es, das Matching zu finden, bei dem die Summe der Gewichte der Kanten in M maximal ist. Hierfür wird im Abschnitt Matching-Algorithmus die Ungarische Methode beschrieben, die die Grundlage für viele weitere Algorithmen zur Berechnung von Matchings mit maximalem Gewicht darstellt (Kuhn, 1955).

Während die Konstruktion des Graphens im Matching-Service durch die Klasse Connector unter Einsatz des Python-Paketes NetworkX (NetworkX developers, 2021) umgesetzt wird, ist der Matching-Algorithmus lediglich im Simulator implementiert (siehe Abschnitt 6.3). Dies soll eine Prüfung der Konstruktion des Graphens im laufenden Betrieb der Tauschbörse und eine manuelle Bearbeitung aller potenzieller Matches ermöglichen.

Konstruktion des Graphen

Damit ein CareGiver-CareTaker-Tupel der Kantenmenge E des Graphen G hinzugefügt werden kann, müssen folgende drei zentrale Bedingungen erfüllt sein:

a. Weder der CareGiver noch der CareTaker befinden sich in einer aktiven Tauschbeziehung, d.h. es existiert keine MatchedWith-Entität mit Status active, in der die UID des CareGivers bzw. CareTakers vorkommt.
b. CareGiver und CareTaker sind nicht miteinander verwandt, d.h. es existiert keine RelativeOf-Entität in der die UID des CareGivers und CareTakers gemeinsam vorkommen.
c. Die Entfernung zwischen den Wohnorten des CareGivers und des CareTakers ist kleiner oder gleich dem Aktivitätsradius (radius_of_activity) des CareGivers.

Während sich die Bedingung a und Bedingung b durch eine Überprüfung der MatchedWith-, bzw. RelativeOf-Entitäten überprüfen lässt, wird Bedingungen c anhand der folgenden Ungleichung geprüft:

$$O \leq R + \sqrt{\frac{A_{cg}}{\pi}} * d + \sqrt{\frac{A_{ct}}{\pi}} * d$$

Hierbei ist O die Orthodrome der durch Latitude und Longitude beschriebenen zentralen Punkte in den jeweiligen Postleitzahlengebieten der CareGiver und CareTaker. Die Orthodrome ist die kürzeste Verbindung zweier Punkte auf einer Kugeloberfläche und lässt sich anhand der Haversine-Formel berechnen. R ist der Aktivitätsradius des CareGivers, A ist die Fläche des Ortes des CareGivers (A_{cg}) bzw. des CareTakers (A_{ct}) und d ist eine Konstante, die über den bereits erwähnten Parameter deviation kontrolliert werden kann. Durch die Division der Gebietsfläche durch π und anschließendes Ziehen der Wurzel, ergibt sich der Radius der Gebietsfläche. Im rechten Teil der Ungleichung wird also der Aktivitätsradius des CareGivers um die Radien der Gebietsflächen von CareGiver und Caretaker verlängert, welche jeweils mit d multipliziert werden. Unter der Annahme kreisförmiger Postleitzahlengebiete, lässt sich so deren Fläche in der Prüfung berücksichtigen und die Ungenauigkeiten in der Lokalisation der CareGiver und CareTaker ausgleichen. Zudem lässt sich durch den Faktor d anpassen, ob man eine liberale oder konservative Prüfung vornehmen möchte.

Die Prüffunktion ist letztendlich definiert als:

$$p(v_i \in Caregiver, v_j \in Caretaker):$$
$$= 1\ wenn\ Bedingung\ a, b\ und\ c\ erfüllt\ sind, sonst\ 0$$

Um die Qualität der der Passung von möglichen Paarungen zu repräsentieren, wird allen Kanten ein Gewicht zugeordnet. Für die Bestimmung der Qualität der Passungen sind unterschiedliche Kriterien denkbar, z.B. ähnliche Interessen von CareGiver und CareTaker. Da jedoch keine relevanten Informationen zu den Teilnehmenden vorliegen, beschränkt sich die Gewichtungsfunktion auf bestehende Tauschbeziehungen und ist wie folgt definiert:

$$w(E) \coloneqq 2\ wenn\ CareTaker\ in\ E\ mit\ aktivem\ CareGiver\ verwandt\ ist, sonst\ 1$$

Dadurch, dass Kanten ein höheres Gewicht erhalten, wenn der CareTaker mit einem CareGiver verwandt ist, der sich bereits in einer aktiven Tauschbeziehung befindet, werden diese Kanten durch den Matching-Algorithmus bevorzugt. So können aktive LDC schneller für ihre Vorleistung mit einer Gegenleistung belohnt werden.

Matching-Algorithmus

Damit die Ungarische Methode auf dem Graphen G angewendet werden kann, wird dieser durch Hinzufügen von sogenannten Dummyknoten und Dummykanten (mit einem Kantengewicht von null) zu dem balancierten, vollständig bipartiten Graphen G' umgeformt. G' zeichnet sich nun dadurch aus, dass beide Partitionen aus gleich vielen Knoten bestehen und dass jeder Knoten, der einen Partition mit allen Knoten der anderen Partition verbunden ist.

Es folgt eine Beschreibung des Ablaufs der Ungarischen Methode in einer graphentheoretischen Abwandlung, welche sich an den Ausführungen anderer Autoren orientiert (Asratian et al., 1998; Becker et al., 2015; Galil, 1986). In dieser graphentheoretischen Abwandlung werden in Teilgraphen von G' so lange Augmentationswege bestimmt bis das Matching komplett ist. Augmentationswege sind alternierende Pfade, bei denen weder Anfangs- noch Endknoten gematcht sind. Alternierende Pfade sind Sequenzen von Kanten, die abwechselnd im Matching M und der Menge $E \backslash M$ enthalten sind. Knoten sind gematcht, wenn sie Bestandteil einer Kante in M sind, sonst frei.

Entfernt man nach Abschluss des Algorithmus die Dummykanten ist das Ergebnis das Matching mit dem maximalen Gewicht. Bei mehrmaliger Durchführung des Algorithmus auf dem gleichen Graphen kann das Ergebnis zwar unterschiedlich ausfallen, ist jedoch stets optimal (Becker et al., 2015).

Für die Implementierung des Matching-Algorithmus im Simulator (siehe Abschnitt 6.3) wurde ebenfalls die Python-Bibliothek NetworkX (NetworkX developers, 2021) genutzt, in welcher die Ungarische Methode umgesetzt ist. Um diese Implementierung nutzen zu können, müssen dem konstruierten Graphen zwar Dummykanten aber keine Dummyknoten hinzugefügt werden.

Listing 1: Beschreibung der Ungarischen Methode in einer graphentheoretischen Abwandlung

1. Ordne jedem Knoten eine Markierung l zu, sodass für alle durch eine Kante verbundenen Knotenpaare (v_i, v_j), $l(v_i) + l(v_j) \geq w\left((v_i, v_j)\right)$ gilt. Dies erreicht man durch Markierung der CareGiver-Knoten (CGK) mit dem höchsten Gewicht seiner Kanten und Markierung der CareTaker-Knoten (CTK) mit 0.
2. Konstruiere den Gleichheitsgraphen
   $G_g = (V, E_g)$ mit $E_g = \{(v_i, v_j) \in E \mid l(v_i) + l(v_j) = w\left((v_i, v_j)\right)\}$.
3. Initialisiere das leere Matching $M = \emptyset$
4. Setze die Menge besuchter Knoten aus CGK $B_{CG} = \emptyset$ und CTK $B_{CT} = \emptyset$
5. Wenn das Matching perfekt ist, beende den Algorithmus.
6. Bestimme Augmentationsweg P.
   a. Finde einen Knoten $v \in CGK \wedge v \notin B_{CG}$ als Wurzel eines alternierenden Pfades.
   b. Konstruiere schrittweise unter Aktualisierung von B_{CG} und B_{CT} einen alternierenden Pfad in G_g bis Augmentationsweg gefunden wurde oder es keine passenden Kanten mehr gibt.
   c. Wenn Augmentationsweg gefunden wurde, dann aktualisiere das Matching als $M' = M \backslash P \cup P \backslash M$ und gehe zu Schritt 4, sonst gehe zu Schritt 7.
7. Erstelle neuen Gleichheitsgraph durch Anpassung der Markierungen
   $l(v)$ durch $l'(v)\begin{cases} l(v) - \Delta \; wenn \; v \in B_{CG} \\ l(v) + \Delta \; wenn \; v \in B_{CT} \\ l(v) \; sonst \end{cases}$
   mit $\Delta = \min_{i,j}\left(l(v_i \in B_{CG}) + l(v_j \in CTK \backslash B_{CT}) - w\left((v_i, v_j)\right)\right)$.
   Gehe zu Schritt 4.

3.3.7 Map-Service

Aufgabe des Map-Services ist die Visualisierung der LDC und LDR auf einer digitalen Karte (siehe User Story 9). Hierfür stellt er die zwei Methoden build_map und get_map_path als Schnittstellen für den MapController bereit. Durch Aufruf der Methode build_map wird die Karte auf Basis der aktuellen Anmeldungen konstruiert bzw. aktualisiert. Hierzu wird mittels der Python-Bibliothek Folium (Story, 2013) ein HTML-Dokument erstellt, in dem eine Leaflet-

Karte (Agafonkin, 2021) eingebettet ist. Bei Aufruf der Methode get_map_path wird lediglich der der Pfad zum erzeugten HTML-Dokument zurückgegeben, welches durch die GUI dargestellt wird.

Durch die Weitergabe des HTML-Dokuments wird die Dependency Rule verletzt, da innerhalb der Fachdomäne Annahmen über technische Details der Darstellung getroffen werden. In einer nächsten Iteration könnte der Map-Service insofern angepasst werden, als dass er den einzelnen LDC und LDR ihre Geokoordinaten zuordnet und die GUI diese über den entsprechenden Controller anfordert. Die Logik zur Erzeugung der Karte könnte entsprechend auf Ebene der GUI implementiert werden.

3.3.8 Controller

Die Controller stellen der GUI eine Schnittstelle bereit, um die Services des Applikationskerns zu steuern und Daten mit diesem auszutauschen. Da die Entitäten in der GUI als Python-Dicts und im Applikationskern als Python-Objekte repräsentiert werden, muss auf Ebene der Controller eine Transformation der Daten stattfinden. Hierzu implementiert die Klasse AbstractCrudController die Methoden encode sowie decode und die anderen Controller-Klassen, welche mit Entitäten arbeiten, erben diese. Um eine Entität aus dem Objekt-Format in ein Python-Dict zu übersetzten, nutzt die Methode encode die Methode to_dict der AnitaDomain-Objects. Zur Transformation einer Entität aus dem Dict-Format in ein Python-Object wird die Klassen-Methode from_dict der AniTa-Domain-Objects aufgerufen. Werden Entitäten im GUI durch den Nutzer verändert, werden die neuen Werte zunächst als Zeichenkette interpretiert. Daher müssen diese vor der Transformation in ein Python-Objekt wieder in die erwarteten Datentypen übersetzt werden. Hierfür nutzt die Methode encode die Methode get_value_type_dict der AniTaDomainObjects.

Wie in Tabelle 6 zu sehen ist, leiten alle Controller, die mit Entitäten arbeiten, vom AbstractCrudController ab. Während die meisten Controller, lediglich die Transformation der Daten verantworten und eine Schnittstelle bereitstellen, ermöglicht der RegistrationController zusätzlich das Parsing der Anmeldemails. Der MainController hingegen bietet keine Funktionalität und dient lediglich als zentraler Ort zur Instanziierung der verwendeten Controller.

Tabelle 6: Überblick über die Controller im AniTa-Tool

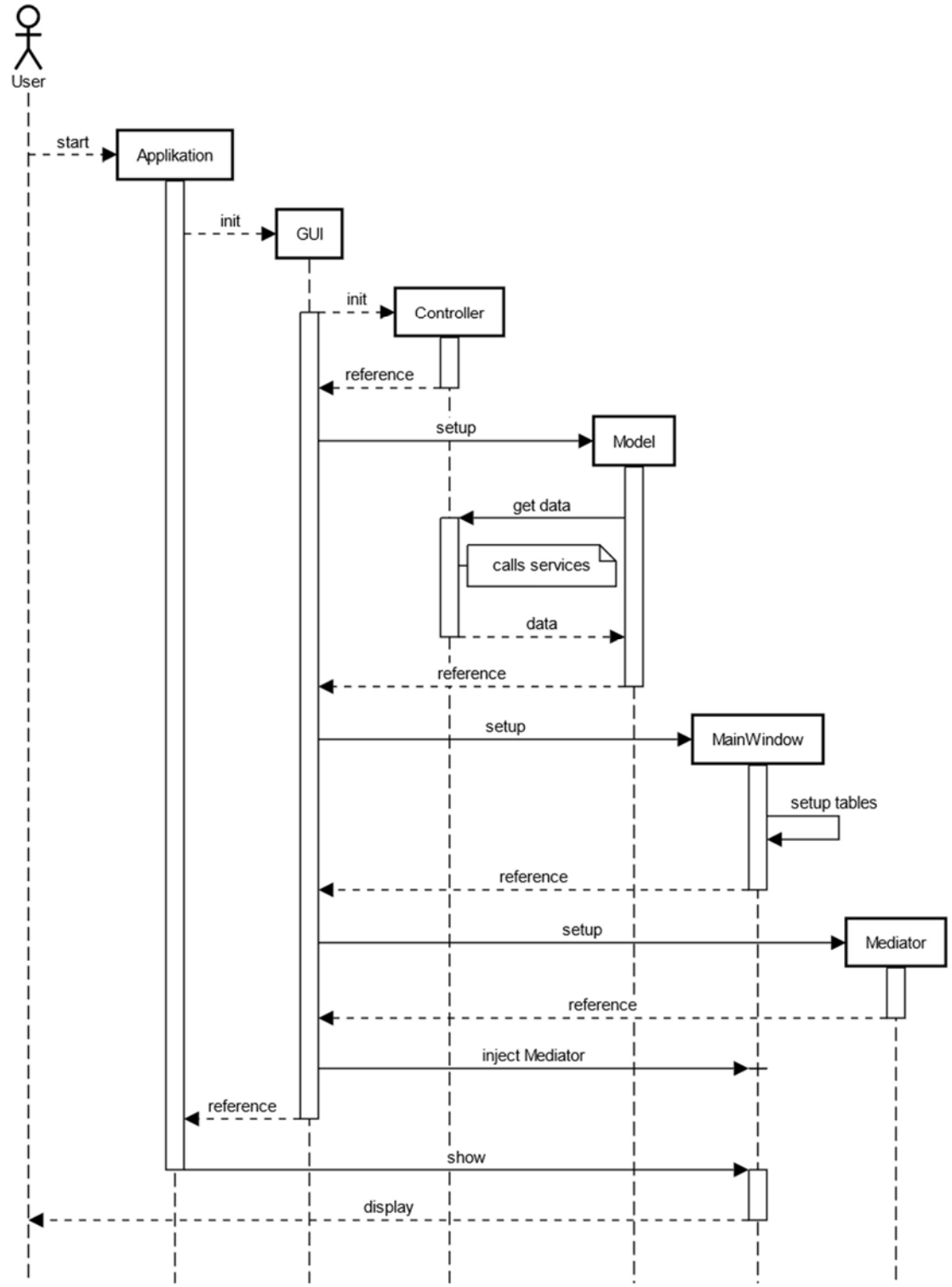
Name	***Parent***	***Funktion***
AbstractCrudController	-	Implementierung und Vererbung der Methoden decode und encode zum Dekodieren bzw. Enkodieren von Entitäten
CareGiverCrudController	AbstractCrudController	Schnittstelle zum CareGiverService
CareTakerCrudController	AbstractCrudController	Schnittstelle zum CareTakerService
MatchedWithCrudController	AbstractCrudController	Schnittstelle zum MatchedWithService
RelativeOfCrudController	AbstractCrudController	Schnittstelle zum RelativeOfService
MatchingController	AbstractCrudController	Schnittstelle zum MatchingService
RegistrationController	AbstractCrudController	Parsing der Anmeldemails und Schnittstelle zum RegistrationService
MapController	-	Schnittstelle zum CareGiverService
TransactionController	-	Schnittstelle zum TransactionService
MainController	-	Zentrale Instanziierung aller Controller

3.4 Präsentationsschicht

Wie in Abbildung 6 dargestellt ist, besteht die Präsentationschicht bzw. die GUI aus den Komponenten MainWindow (siehe Abschnitt 3.4.1), Model (siehe Abschnitt 3.4.2), Mediator (s. Abschnitt 3.4.4) und Dialogs (siehe Abschnitt 3.4.3). Hierbei ist das MainWindow die grafische Oberfläche, über welche die Nutzer:innen mit dem AniTa-Tool interagieren und die ihnen die Stammdaten der Teilnehmenden darstellt. Der Baustein Model definiert das Datenmodell der GUI und nutzt die Schnittstellen der CRUD-Controller, um die Stammdaten der Teilnehmenden zu laden, zu ändern oder zu löschen. Beim Baustein Dialogs handelt es sich um verschiedene Dialoge zur Benachrichtigung der Nutzer:innen oder zum Abfragen von Eingaben, etwa zur Nutzung der Services des Anita-Tools. Der Mediator stellt die zentrale Schnittstelle zwischen MainWindow, Model und Dialogs dar und realisiert die Kommunikation zwischen diesen.

Starten Nutzer:innen das Anita-Tool, wird über die Applikation die GUI initialisiert (siehe Abbildung 11). Die Komponente GUI verantwortet das Setup der Benutzeroberfläche. Sie richtet also die Komponenten der Präsentationsschicht ein und stellt die Verbindungen zwischen diesen und zur Fachdomäne her. Nachdem die GUI die Controller initialisiert hat, werden die Datenmodelle eingerichtet. Im Rahmen ihres Setups werden den Modellen die zugehörigen Controller via Dependency Injection (DI) eingespeist (siehe Abschnitt 4.3). Mittels dieser Controller werden die Datenmodelle mit den Daten der Teilnehmenden aus dem Applikationskern gefüllt. Anschließend wird durch die GUI das Setup des MainWindow durchgeführt. Nachdem das MainWindow die eigenen Datentabellen eingerichtet hat, folgt die Einrichtung des Mediators durch die GUI. Erst nach erfolgreicher Einrichtung des Mediators wird dieser dem MainWindow eingespeist. Hierdurch werden Probleme, welche sich aus der zirkulären Abhängigkeit zwischen Mediator und MainWindow ergeben könnten, vermieden. Ist das Setup durch die GUI abgeschlossen, wird durch die Applikation die Anzeige des MainWindow für die Nutzer:innen angefordert. Die Initialisierung bzw. das Setup der Dialoge erfolgt, sobald eine Nutzeraktion getätigt wurde, die einen Dialog zur Folge hat.

Abbildung 11: UML-Sequenzdiagramm des Starts des AniTa-Tools

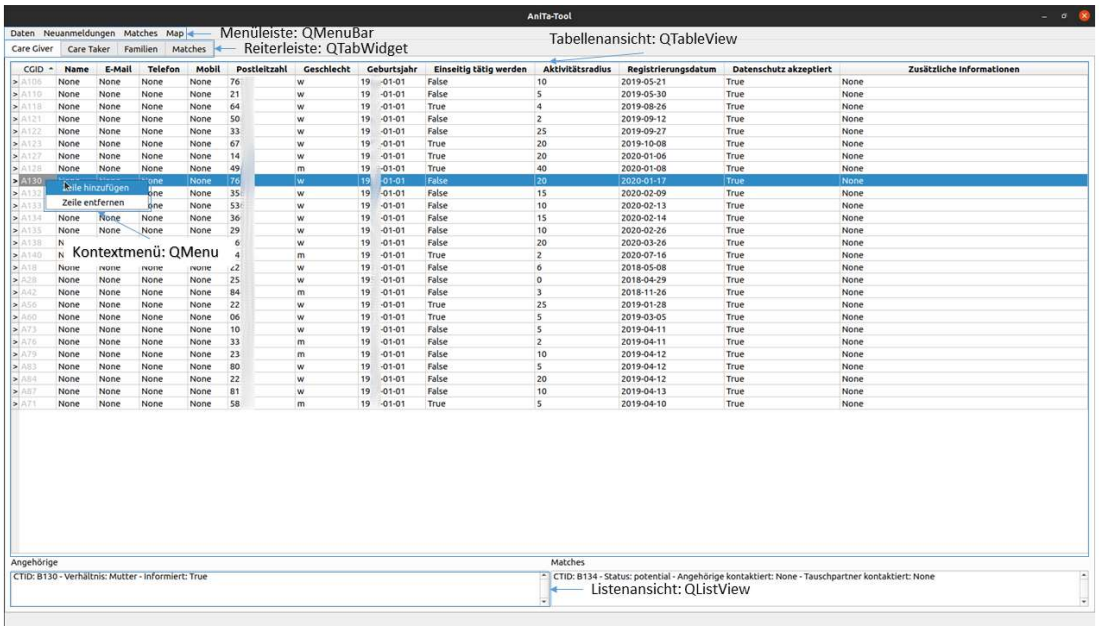


Als zentrale Technologie für die Umsetzung der Komponenten des GUI wurde PyQT5 gewählt. PyQT5 umfasst eine Reihe von Python-Bindings für das auf Framework QT, welches auf C++ basiert und unter anderem Windows, Linux und macOS unterstützt (Riverbank Computing, 2021a). Zum Erstellen und Testen der Layouts des GUI kann zudem das Tool Qt Designer genutzt werden, welches die Layouts in Form von XML-Dateien (auch .ui-Dateien) exportiert (The Qt Company, 2021a). Diese Dateien können mittels des uic-Moduls von PyQT5 in Python Code umgewandelt werden (Riverbank Computing, 2021b). PyQT beinhaltet mehrere Klassen, die einer Model/View-Architektur folgen, welche Ähnlichkeiten zum Design-Pattern Model-View-Controller aufweist, jedoch View und Controller in der View kombiniert. Um auf Ebene einzelner Datenelemente modifizieren zu können, wie diese bearbeitet und dargestellt werden, können sogenannte Delegates verwendet werden. (The Qt Company, 2021b)

3.4.1 MainWindow

Wie im vorherigen Abschnitt erläutert, stellt das MainWindow die zentrale Benutzeroberfläche des AniTa-Tools dar. Das MainWindow des AniTa-Tools ist in Abbildung 12 dargestellt. Es basiert auf der PyQT5-Klasse QMainWindow und setzt sich aus fünf Hauptkomponenten zusammen: Einer Menüleiste, einer Reiterleiste, Tabellenansichten mit Kontextmenü und Listenansichten.

Abbildung 12: Screenshot des MainWindow des AniTa-Tools mit Bezeichnungen der Hauptkomponenten und unterliegender PyQT5-Klassen



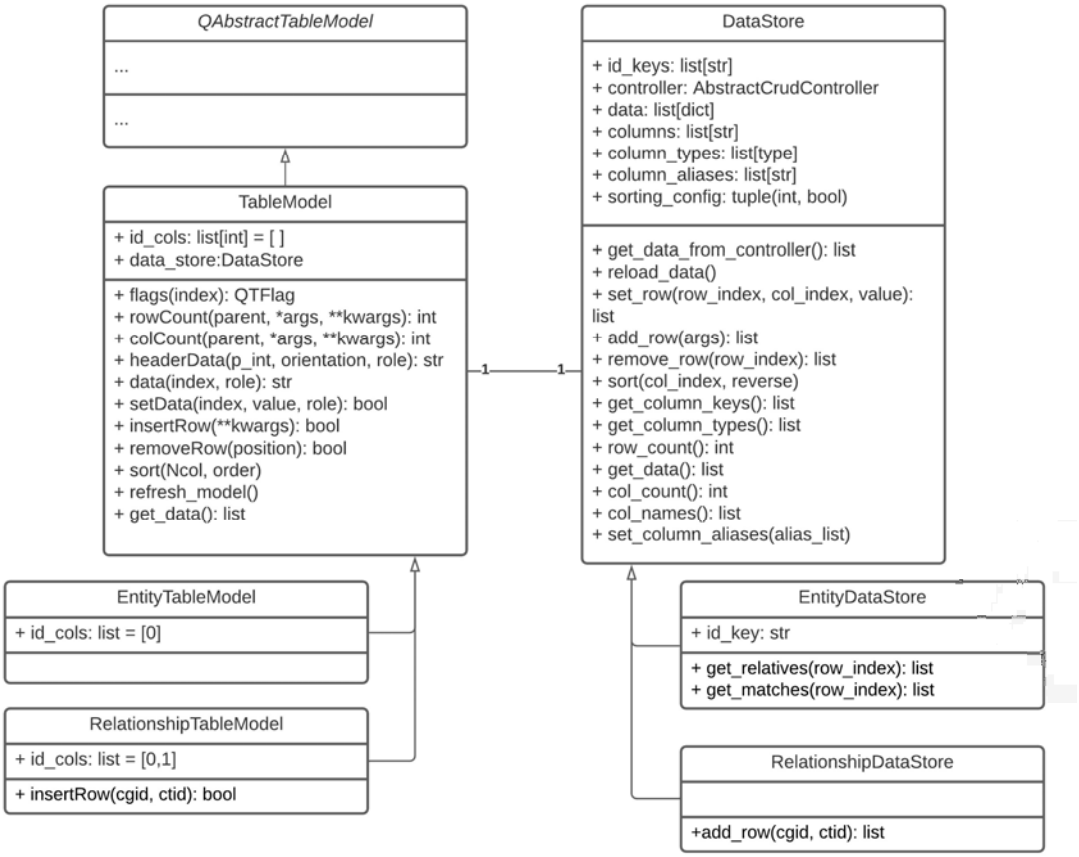
Über die Menüleiste, welche direkt auf der PyQT5-Klasse QMenu basiert, können die Nutzer:innen die Funktionalitäten des Transaction-Service, Registration-Service, Matching-Service und Map-Service nutzen. Die Reiterleiste basiert auf der PyQT5-Klasse QTabWidget und ermöglicht es den Nutzer:innen zwischen verschiedenen Tabellenansichten für die Entitäten bzw. Entitäts-Beziehungen zu wechseln (siehe Abschnitt 3.3.1). Die einzelnen Tabellenansichten basieren auf der PyQT5-Klasse QTableView, welches die Tabellen darstellt, welche wiederum auf dem TableModel basieren (siehe Abschnitt 3.4.2). Indem Nutzer:innen auf eine Zelle in der Kopfzeile der jeweiligen Tabelle klicken, werden die Datensätze auf Basis dieser Spalte sortiert. Durch Doppelklick auf eine Tabellenzeile lassen sich die entsprechenden Werte unter Nutzung der CRUD-Services verändern. Abhängig vom Datentyp der Spalte werden die Nutzer:innen hierbei durch ein Delegate unterstützt (siehe Abschnitt 3.4.2). Bei Rechtsklick auf eine Zeile öffnet sich ein Kontextmenü, welches auf der PyQT5-Klasse QMenu basiert und das Hinzufügen bzw. Entfernen von Zeilen ermöglicht. Durch Markierung einer Zeile werden die Listenansichten im unteren Bereich des MainWindow aktualisiert. Diese zeigen für den markierten CG bzw. CT die assoziierten Angehörigen bzw. Matches an und basieren auf der PyQT5-Klasse QListView.

Die Darstellung der Stammdaten in Tabellenform wurde gewählt, da die Nutzer:innen eine entsprechende Darstellung durch die Arbeit mit Microsoft Excel gewohnt sind. Anders als es bei Excel die Regel ist, wird jedoch durch die Delegates sichergestellt, dass nur Daten des vorgesehenen Datentyps in die entsprechende Zeile eingefügt werden.

3.4.2 Model

Das Datenmodell des Anita-Tools wird in der Präsentationsschicht durch die Klasse TableModel repräsentiert, welche direkt von der PyQT5-Klasse QAbstractTableModell erbt. TableModel ist wiederum die Superklasse des EntityTableModels und RelationshipTableModels, deren Unterscheidung lediglich darin begründet ist, dass die Zeilen des RelationshipTableModel über zwei ID-Spalten verfügen. Wie in Abbildung 13 zu sehen ist, folgt auch der DataStore diesem Schema, indem er die Superklasse für den EntityDataStore und den RelationshipDataStore bildet. Während das TableModel die notwendigen Methoden des QAbstractTableModels implementiert und so die Integration in das PyQT5-Framework gewährleistet, stellt der DataStore die Verbindung zu den Controllern der Fachdomänenschicht dar und stellt Methoden zur Bearbeitung der Daten bereit.

Abbildung 13: UML-Klassendiagramm des TableModels und des DataStores



Delegates

Zwar nehmen Delegates in der Model/View-Architektur eine Sonderrolle ein, werden im A-nita-Tool jedoch dem Model zugeordnet, da sie primär dazu dienen, dass die eingegebenen Daten den Datentypen des Modells entsprechen. Tabelle 7 zeigt die vier verwendeten Delega-tes, bei welchem Datentyp sie zum Einsatz kommen und welchen Effekt ihre Verwendung hat. Die Zuordnung der Delegates zu den Spalten des Datenmodells erfolgt während des Setups der Datentabellen durch das MainWindow.

Tabelle 7: Übersicht der Delegates des AniTa-Tools

Name	***Datentyp***	***Effekt***
InlineEditDelegate	String	Ermöglicht das Bearbeiten des bestehenden Textes in einen Textfeld.
ComboDelegate	Enum	Öffnet eine Combobox zur Auswahl von Enum-Werten.
DateDelegate	Date	Öffnet einen Kalender zur Auswahl eines Datums.
SpinBoxDelegate	Numeric	Öffnet eine SpinBox zur Eingabe numerischer Werte.

3.4.3 Dialogs

Dialoge stellen eigenständige Fenster dar, welche im AniTa-Tool dazu verwendet werden, um die Nutzer:innen über ein Ereignis zu informieren oder Eingaben dieser entgegenzunehmen. Bei den angeforderten Nutzereingaben kann es sich hierbei um eine einfache Bestätigung, wie etwa beim ResetDialog oder um Formulare mit mehreren Eingabefeldern, wie bei RegistrationFormDialog handeln. Tabelle 8 beschreibt die Dialoge, welche im AniTa-Tool verwendet werden.

Tabelle 8: Übersicht der im Anita-Tool verwendeten Dialoge

Name	***Beschreibung***
ChooseIdsDialog	Dialog mit zwei Comboboxen zur Auswahl einer CGID und einer CTID
ExitDialog	Dialog mit drei Buttons zur Auswahl, ob vorhandene Änderungen gespeichert bzw. zurückgesetzt werden sollen oder das Schließen der Applikation abgebrochen werden soll.
MailParserDialog	Dialog mit Textfeld zur Eingabe des Inhaltes der Anmeldemails.
PotentialMatchesDialog	Dialog zur Anzeige des Ergebnisses des Matchings, mit zwei Buttons, um diese zu übernehmen oder zu verwerfen.
RefreshMapDialog	Dialog zur Information, dass die Landkarte aktualisiert wird, mit Button zum Öffnen der aktualisierten Karte
RegistrationConfirmationDialog	Dialog, der den erfolgreichen Abschluss einer Registrierung zurückmeldet und die Möglichkeit anbietet eine Bestätigungsmail zu verschicken.
RegistrationFormDialog	Dialog mit mehreren Eingabefeldern, zur Eingabe und Kontrolle der Stammdaten von Neuanmeldungen.
RemoveRowDialog	Dialog mit zwei Buttons, zur Bestätigung, ob ein Datensatz gelöscht werden soll.
ResetDialog	Dialog mit zwei Buttons, zur Bestätigung, ob Änderungen in den Daten zurückgesetzt werden sollen.
StartMatchingDialog	Dialog zum Starten des Matchings, mit der Möglichkeit nur Anmeldungen ab einem bestimmten Datum zu berücksichtigen.

3.4.4 Mediator

Die Aufgabe des Mediators ist es die Kommunikation der Komponenten der Präsentationsschicht sicherzustellen und an einem Ort zu bündeln. Darüber hinaus verantwortet der Mediator bei komplexeren Folgen von Dialogen den Dialogfluss. Eine Übersicht der Methoden des Mediators einschließlich ihrer Effekte und der daran beteiligten Komponenten ist in Tabelle 9 dargestellt.

Die Aufgaben des Mediators in der Steuerung des Dialogflusses sollen im Folgenden am Beispiel der Bearbeitung einer Neuanmeldung verdeutlicht werden. Der Dialogfluss beginnt durch

Aufrufen der Aktion Abrufen im Menüpunkt Neuanmeldung und gliedert sich in drei Schritte. Im ersten Schritt wird durch den Mediator der MailParserDialog geöffnet und anschließend die Benutzereingabe an den RegistrationController weitergegeben. Konnten die Anmeldedaten erfolgreich geparst werden, gibt der Mediator im zweiten Schritt die Daten an den RegistrationFormDialog weiter und öffnet diesen. Nachdem die Nutzer:innen die Daten kontrolliert haben, werden diese erneut an den RegistrationController weitergegeben, damit diese in der Fachdomänenschicht geprüft werden können. Bei erfolgreicher Prüfung öffnet der Mediator im dritten Schritt den RegistrationConfirmationDialog. Möchten die Nutzer:innen eine Bestätigungsmail versenden, ruft der Mediator die entsprechende Methode im RegistrationController auf und nutzt den QDesktopService, um den mailto-Link zu öffnen. Bei negativer Prüfung hingegen öffnet der Mediator eine Fehlernachricht und die Nutzer:innen erhalten die Möglichkeit die Daten im RegistrationFormDialog zu korrigieren.

Tabelle 9: Übersicht der Methoden des Mediators, ihres Effektes und der beteiligten Komponenten

Methode	***Effekt***	***Beteiligte Komponenten***
save	Speichert Änderungen in den Daten.	- TransactionController - MainWindow
reset	Öffnet Dialog für Bestätigung und setzt ggf. ungespeicherte Änderungen in den Daten zurück.	- TransactionController - ResetDialog - MainWindow
insert_row_into_model	Fügt einem TableModel eine Zeile hinzu	- EntityTableModel - RelationshipTableModel - MainWindow
choose_ids_and_insert_row	Öffnet Dialog zur Auswahl von UIDs und fügt TableModel eine entsprechende Zeile hinzu.	- CareGiverCrudController - CareTakerCrudController - ChooseIdsDialog
remove_row_from_model	Öffnet Dialog zur Bestätigung und entfernt ggf. die gewählte Zeile aus TableModel.	- TableModel - RemoveRowDialog - MainWindow
exit	Öffnet ggf. einen Dialog zur Auswahl ob Änderungen gespeichert oder zurückgesetzt werden sollen und schließt ggf. die Applikation.	- TransactionController - ExitDialog - MainWindow
register	Führt den Dialogfluss für das Einfügen von Neuanmeldungen aus.	- MailParserDialog - RegistrationController - RegistrationFormDialog - MainWindow - QErrorMessage - RegistrationConfirmationDialog - QDesktopServices
start_matching	Führt den Dialogfluss für das Matching aus.	- StartMatchingDialog - MatchingController - PotentialMatchesDialog - MainWindow - QErrorMessage
refresh_map	Aktualisiert die Landkarte anhand der bestehenden Anmeldungen.	- RefreshMapDialog - GenericWorkerThread - MainWindow
open_map	Öffnet die Karte der Anmeldungen.	- MapController - QWebEngineView - MainWindow

3.5 Infrastrukturschicht

Während andere Systeme auf Ebene der Infrastrukturschicht aus mehreren Komponenten bestehen können, besteht diese Schicht beim AniTa-Tool lediglich aus einer relationalen

Datenbank (siehe Abbildung 6), welche in Abschnitt 3.5.1 beschrieben wird. Da die Datenbasis, insbesondere bezüglich der Postleitzahlengebiete, eine entscheidende Rolle in der Funktionalität des AniTa-Tools spielt, wird auf diese in Abschnitt 3.5.2 eingegangen.

3.5.1 Datenbank

Wie bereits erwähnt war die Auswahl der Datenbank durch die technische Infrastruktur insofern eingeschränkt, als dass kein DBMS mit eigenem Serverprozess gewählt werden konnte (siehe Abschnitt 3.1.). Zudem war zu erwarten, dass nur eine geringe Menge an Daten persistiert werden muss und aufgrund der geringen Anwenderzahl schreibende und lesende Zugriffe nur in niedriger Frequenz stattfinden werden. Da neben den klassischen Eigenschaften von Datenbanksystemen – Atomarität, Konsistenz, Isolation und Dauerhaftigkeit – eine zentrale Anforderung an das Datenbanksystem ist, dass es die Daten in einer Datei auf dem Netzwerklaufwerk persistiert, wurde die Programmbibliothek SQLite als Datenbank-Engine gewählt. SQLite ist eine weit verbreitete eingebettete SQL-Datenbank, welche die Daten von einer plattformunabhängigen Datei liest und in dieser speichert. (The SQLite Consortium, n.d.) Zudem implementiert SQLite ein relationales Datenbankschema und wird von SQLAlchemy unterstützt. Bei Änderungen der Anforderungen an die Datenbank ließe sich diese daher mit geringem Aufwand gegen ein anderes relationales Datenbanksystem austauschen.

Das relationale Datenbankschema des AniTa-Tools entspricht dem im Abschnitt 3.3.1 erläuterten und in Abbildung 7 dargestellten Modell der Entitäten der Fachdomäne. Wie in Tabelle 10 dargestellt ist, wird jeder Entität eine Datenbanktabelle zugeordnet. Die Verknüpfung der Tabellen erfolgt über die jeweiligen Fremdschlüssel.

Tabelle 10: Übersicht der Tabellen der relationalen Datenbank und ihrer Schlüssel

Tabellenname	***Primärschlüssel***	***Fremdschlüssel***
caregiver	cgid	zip_code
caretaker	ctid	zip_code
relative_of	cgid, ctid	cgid, ctid
matched_with	cgid, ctid	cgid, ctid
place	zip_code	-

3.5.2 Datenbasis

Während die bestehenden Daten der LDC und LDR nach einer manuellen Anpassung der Excel-Dateien an das Datenbankschema mittels einfacher Python-Skripte in die Datenbank migriert werden können, mussten die Daten zu den deutschen Postleitzahlengebieten über externe Quellen bezogen werden.

Hierzu wurden zwei Datensätze genutzt. Zum einen wurde der Datensatz mit „georef-germany-postleitzahl“ von opendatasoft (Opendatasoft, 2019) und der Datensatz „plz-5stellig-daten“ von suche-postleitzahl.org (Schwochow, 2019) bezogen. Beide Datensätze sind unter der Open Database Licence (Open Knowledge Foundation, n.d.) veröffentlicht. Während der Datensatz „georef-germany-postleitzahl“ die Postleitzahlen, Ortsnamen, und Latitude und Longitude eines zentralen Punktes im Postleitzahlengebiet enthält, stellt der Datensatz „plz-5stellig-daten“ Informationen zur Gebietsfläche in Quadratkilometern und Einwohnerzahl des jeweiligen Postleitzahlengebietes bereit. Nachdem beide Datensätze bereinigt und fehlende Werte manuell ergänzt wurden, wurden sie über die Postleitzahlen zusammengeführt. Der so entstandene Datensatz stellt die Datenbasis der Entität Place dar.

3.6 Querschnittskonzepte

Die Querschnittskonzepte des AniTa-Tools sind technische Komponenten, die mit anderen Komponenten auf verschiedenen Schichten interagieren. Die jeweilige Funktion der Querschnittkonzepte Applikation, Config, Context, Exceptions, Logging und Migration sind in Tabelle 11 dargestellt.

Tabelle 11: Querschnittskomponenten und ihre Funktion im AniTa-Tool

Komponente	***Funktion im AniTa-Tool***
Applikation	Zentraler Einstiegspunkt in das AniTa-Tool, der die Initialisierung der einzelnen Komponenten anstößt.
Config	Bereitstellungen von Parametern aus einer YAML-Datei, zur Konfiguration einzelner Aspekte anderer Komponenten.
Context	Singleton zur Definition der Datenbankverbindung und Bereitstellung der SQL-Alchemy-Session und Definition von Event-Listenern zum Tracking des Zustandes der Session.
Exceptions	Definition von Exceptions zur Fehlerbehandlung im AniTa-Tool.
Logging	Logging von nicht-behandelten Fehlern in einer Log-Datei, zum Debugging des AniTa-Tools.
Migration	Erstellen der Tabellen in der Datenbank und importieren der bestehenden Daten aus CSV-Dateien.

Während die Applikation der zentrale Einstiegspunkt in das AniTa-Tool ist, definiert der Context die Datenbankanbindung und stellt mittels des Singleton-Patterns sicher, dass alle Komponenten die gleiche SQLAlchemy-Session nutzen. Wo immer Fehler im AniTa-Tool antizipiert werden können, werden diese mit spezifischen Exceptions signalisiert und entsprechend behandelt. Treten dennoch Fehler auf, die nicht behandelt werden, werden diese durch das Logging in eine Datei geschrieben, sodass diese im Nachhinein nachvollzogen und behandelt werden können. Da die Datenbank initialisiert werden muss bevor sie genutzt werden kann, wurde das Migrations-Tool Alembic (Bayer, n.d.) genutzt, um die Datenbanktabellen anhand des festgelegten Schemas erstellen zu können. Bei Änderungen am Modell der Fachdomäne lassen sich mittels Alembic auch die Tabellen der Datenbank anpassen und etwa fehlende Werte bei neuen Feldern mit Default-Werten füllen. Über die Config lassen sich verschiedene Einstellungen am AniTa-Tool vornehmen, ohne in den Programmcode eingreifen zu müssen. Dies gelingt durch Änderungen von Werten in einer YAML-Datei, welche dann zur Laufzeit ausgelesen und durch die jeweiligen Komponenten importiert wird. Tabelle 12 gibt einen Überblick über die Konfigurationsmöglichkeiten.

Tabelle 12: Übersicht über die Konfigurationsmöglichkeiten

Konfiguration	***Beschreibung***
Profil	Einstellung, ob die Applikation im Profil DEVELOPMENT zum Entwickeln und Testen oder im Profil PRODUCTION zur Nutzung im laufenden Betrieb läuft.
Datenbank	Konfiguration des Pfades zur Datenbank, abhängig davon, ob die Applikation im DEVELOPMENT oder PRODUCTION Profil läuft.
Mailing	Konfiguration der Pfade zu den Templates für die Bestätigungsmails und Einstellung einzelner Aspekte der Formulierung.
Map	Konfiguration der Pfade zum HTML-Dokument der Karte, sowie einer Overlay-Datei und Einstellung von Darstellungsaspekten.
Matching	Einstellung des Faktors d, in der Prüfung, ob ein Match zwischen einem LDC und LDR potenziell möglich ist.
GUI	Konfiguration der Pfade zu den Layout-Dateien.
Logging	Konfiguration des Pfades zur Log-Datei

4 Implementierung

Die Implementierung des AniTa-Tools ist zum einen als Umsetzung der im vorherigen Kapitel beschriebenen Architektur zu verstehen; zum anderen liefert die inkrementelle Umsetzung Erkenntnisse, anhand derer die Architektur weiterentwickelt werden kann. In den folgenden Abschnitten werden anhand von Quellcodebeispielen zentrale Aspekte und wiederkehrende Muster in der Implementierung des AniTa-Tools beschrieben. Hierbei wird erläutert, wie die Frameworks SQLAlchemy und PyQT5 in das AniTa-Tool integriert sind (s. Abschnitte 4.1 und 4.4) und welche Rolle Interfaces und die Dependency Injection in der Implementierung spielen (s. Abschnitte 4.2 und 4.3).

4.1 SQLAlchemy

Damit die Entitäten der Fachdomäne auf die Tabellen der Datenbank gemappt werden können müssen diese von einer SQLAlchemy Base-Klasse ableiten. Listing 2 zeigt, wie über den Aufruf der Funktion declarative_base eine Base-Klasse konstruiert wird, welcher ein MetaData-Objekt zugewiesen wird. Dieses MetaData-Objekt kann als eine Fassade um ein Python-Dictionary verstanden werden, das verschiedene Informationen über die Datenbank beinhaltet, welche wiederum durch die Entitäten beschrieben wird (SQLAlchemy authors and contributors, 2021). Dadurch, dass das MetaData-Objekt auf Modul-Ebene erzeugt wird, registrieren sich alle Objekte, welche von der Base-Klasse ableiten, bei demselben MetaData-Objekt. Da das AnitaDomainObject keine Entität im fachlichen Sinne darstellt, wird durch das setzten der Klassenvariable __abstract__ auf den Wahrheitswert True sichergestellt, dass sich das AnitaDomainObject nicht bei dem MetaData-Objekt registriert.

Listing 2: Auszug aus der Klassendefinition des AnitaDomainObject

```
from sqlalchemy import MetaData
from sqlalchemy.ext.declarative import declarative_base

metadata = MetaData()
Base = declarative_base(metadata=metadata)

class AnitaDomainObject(Base):
    __abstract__ = True

    def to_dict(self) -> Dict[str, object]:
        """

        Returns
        -------
        A dict representing the object with attribute names as
            keys and attribute values as values
        """
        dict_ = {}

        for column in self.__table__.columns:
            dict_[column.name] = getattr(self, column.name)

        return dict_
```

Die Methode to_dict der AnitaDomainObject-Klasse zeigt, wie durch die Ableitung von der Base-Klasse die Klassenvariable __table__ genutzt werden kann, um auf eine generische Weise Objekte der Kindklassen in Python-Dictionaries zu transformieren.

Wie bereits zuvor erläutert, kommuniziert SQLAlchemy mit der Datenbank über Session-Objekte, welche so lange existieren, bis die Änderungen in die Datenbank gespült werden. Die Anbindung der Session an die Datenbank erfolgt wiederum über ein Engine-Objekt. Damit keine unvorhergesehenen Änderungen an den Stammdaten in der Datenbank erfolgen, wurden zwei Stages eingeführt. Die Stage DEVELOPMENT für die Entwicklung und das Testen des AniTa-Tools und die Stage PRODUCTION für den Produktivbetrieb. Welche Stage genutzt wird, kann in der config.yaml-Datei festgelegt werden, aus der auch der Pfad zur SQLite-Datei für die jeweilige Stage ausgelesen wird. In Listing 3 wird dargestellt, wie in der Config in Abhängigkeit der Stage die engine_url erzeugt wird.

Listing 3: Auszug aus der Config zur Erzeugung der enigne_url

```
CURRENT_PROFILE = config['profile']
"""Build path to database"""
if CURRENT_PROFILE == 'DEVELOPMENT':
    db_path = config['db']['development']['url']
elif CURRENT_PROFILE == 'PRODUCTION':
    db_path = config['db']['production']['url']
else:
    raise ValueError('Unknown Profile {}'.format(CURRENT_PROFILE))

sqlite_prefix = 'sqlite:///'
current_db_path = build_path(db_path, 'absolute')
engine_url = sqlite_prefix + current_db_path
```

Wie in Listing 4 dargestellt wird, importiert die Komponente Context die in der Config erzeugte engine_url, um mittels dieser ein Engine-Objekt für die SQLAlchemy-Session zu erzeugen. Die Funktion sessionmaker erzeugt eine Session-Factory, mittels derer Session-Objekte erstellt werden können, welche die gleiche Konfiguration haben. Im Falle des AniTa-Tools beschränkt sich die Konfiguration auf das erzeugte Engine-Objekt und das Setzen des Parameters autoflush auf False. Andere Komponenten importieren nicht direkt die durch die Variable session_factory referenzierte Session-Factory, sondern das durch die Variable Session referenzierte scoped_session-Objekt, welches wiederum die Session-Factory aufruft. Das scoped_session-Objekt kann als eine Registry verstanden werden, welche so lange dasselbe Session-Objekt zurückgibt, bis dieses explizit entfernt wird. Hierdurch wird sichergestellt, dass überall im AniTa-Tool die gleiche Session verwendet wird, ohne diese explizit weitergeben zu müssen.

Listing 4: Auszug aus der Komponente Context zur Erzeugung der SQLAlchemy-Session

```
from sqlalchemy import create_engine
from sqlalchemy.event import listen
from sqlalchemy.orm import sessionmaker, scoped_session

import anita.config as conf

ENGINE_URL = conf.engine_url

"""Create global scoped sessionmaker"""
engine = create_engine(ENGINE_URL)
session_factory = sessionmaker(bind=engine, autoflush=False)

Session = scoped_session(session_factory)
```

Da durch das Setzen von autoflush auf False, Änderungen an den Entitäten nicht mehr automatisch in die Datenbank „gespült“ werden, muss dies manuell erfolgen. Hierzu wird auf Ebene der Repositories die Methode flush des Session-Objektes explizit aufgerufen (siehe Listing 5). Dies ermöglicht es, Änderungen an Entitäts-Objekten vorzunehmen und diese auf Ebene des Applikationskerns zu überprüfen, bevor diese in die Datenbank gespült werden.

Listing 5: Auszug aus dem AbstractCRUDRepository: Nutzung der SQLAlchemy -Session

```
from anita.context import Session
…
…
def create(self, domain_object: AnitaDomainObject)
-> AnitaDomainObject:

    session = self.Session()
    session.add(domain_object)
    session.flush()

    return domain_object
```

So wird etwa bei Änderungen an CareGiver-Objekten geprüft, ob die geänderte Postleitzahl in der Datenbank vorhanden ist (siehe Listing 6). Diese Prüfungen könnten auch auf Ebene der Datenbank stattfinden, dies würde jedoch bedeuten, dass Geschäftslogik in die Persistenzschicht ausgelagert wird und der Applikationskern somit nicht mehr unabhängig von konkreten Datenbanktechnologien ist.

Listing 6: Auszug aus dem CareGiverCRUDService: Update von CareGiver-Objekt

```
def update(self, caregiver: CareGiver) -> CareGiver:
    if not self.place_service.exists(caregiver.zip_code):
        # Check Foreign-Key-Constraint
        raise ConstraintError(
            "zip_code must be german PLZ, but was {}".format(
                caregiver.zip_code
                )
            )

    return self.repository.update(caregiver)
```

4.2 Interfaces

Wie bereits in Abschnitt 3.2 erläutert, soll durch das DIP sichergestellt werden, dass nur Abhängigkeiten von Abstraktionen und nicht von konkreten Implementierungsdetails bestehen. Umgesetzt wird das DIP in der Regel dadurch, dass Schnittstellen oder abstrakte Klassen definiert werden, deren Methoden durch die abhängende Komponente benutzt werden. Die konkreten Implementierungen hängen wiederum von den Schnittstellen bzw. abstrakten Klassen durch die Vererbungsbeziehung ab.

Anders als bei statisch typisierten Sprachen, wie etwa Java, bei denen eine Typenprüfung vor der Laufzeit stattfindet, ist es in Python nicht zwingend erforderlich Schnittstellen oder abstrakte Klassen explizit zu definieren. Vielmehr folgt Python dem Prinzip des duck-typings, wonach Schnittstellen durch die Methoden definiert werden, die durch die jeweilige Klasse angeboten werden. Erst durch den Aufruf der jeweiligen Methode zur Laufzeit durch die abhängende Komponente stellt sich heraus, ob diese Methode tatsächlich angeboten wird. (Anaya, 2020)

Mit der Einführung der abstract base classes (abc) in PEP-3119 stellt Python Entwickler:innen ein Tool zur Verfügung, um durch die Definition abstrakter Klassen und abstrakter Methoden festlegen zu können, welche Methoden von ableitenden Klassen implementiert werden müssen (Anaya, 2020). Wie in Listing 7 dargestellt ist, leitet das AbstractCRUDRepository von der Klasse ABC ab und versieht die Methode retrieve_by_id mit dem Decorator abstractmethod.

Solange diese Methode nicht durch die Kindsklassen des AbstractCRUDRepository implementiert wird, können von diesen keine Objekte instanziiert werden.

Listing 7: Auszug aus dem AbstractCRUDRepository: Definition abstrakter Klassen und Methoden

```
from abc import ABC, abstractmethod
from typing import Optional

…

class AbstractCRUDRepository(ABC):
    """ Abstract Repository for accessing data """

    @abstractmethod
    def retrieve_by_id(self, **domain_object_id)
        -> Optional[AnitaDomainObject]:
        pass
```

Die Nutzung der abc erzwingt zwar, dass abstrakte Methoden durch die Kindsklassen implementiert werden, definiert jedoch nicht, welche Datentypen die übergebenen Argumente bzw. die Rückgabewerte haben müssen. Python erlaubt allerdings die Annotation von Funktionen und Methoden mittels sogenannter Type Hints. Diese Annotationen führen nicht zu einer Typenprüfung zur Laufzeit, können aber durch IDEs genutzt werden, um Abhängigkeiten und Verletzungen von Schnittstellenverträgen auf Ebene des Quellcodes darzustellen. (van Rossum et al., 2014) In Listing 5 ist die annotierte Methode create des AbstractCRUDRepository dargestellt, deren Type Hints darauf hinweisen, dass sowohl das Argument domain_object und der Rückgabewert vom Typ AnitaDomainObject sind.

Durch die Nutzung der abc und des Type Hintings, lassen sich einige Nachteile impliziter Schnittstellen ausgleichen, während die Vorteile erhalten bleiben. Hier ist insbesondere die geringere Menge an notwendigem Quellcode aufgrund impliziter Definition zu nennen, wodurch sich eine höhere Entwicklerproduktivität ergeben kann (Lutz, 2013).

4.3 Dependency Injection

Bei der Nutzung von Schnittstellen stellt sich das Problem, wie abstrakten Schnittstellen zur Laufzeit eine passende konkrete Instanz zugeordnet werden kann. Trägt die Klasse, welche die Schnittstelle nutzt, hierfür die Verantwortung, entsteht eine enge Kopplung zur konkreten

Implementierung der Schnittstelle. Um dies zu vermeiden, kann das Entkopplungsmuster Dependency Injection verwendet werden. Nach diesem Muster übergibt eine dritte Komponente, der sogenannte Assembler, die Referenz auf die konkrete Implementierung der Schnittstelle. Dies erfolgt in der Regel über Setter-Methoden (Setter Injection) oder über den Konstruktor der Klasse (Constructor Injection). (Starke, 2020b)

Wie in Listing 8 am Beispiel des RegistrationService zu sehen ist, wird im AniTa-Tool im Standardfall auf einen Assembler verzichtet. Vielmehr erfolgt die Dependency Injection in der Konstruktor-Methode der jeweiligen Klasse durch die Nutzung von Default-Argumenten. Hierdurch kann ein Objekt des RegistrationService erzeugt werden und Referenzen zu den anderen genutzten Services werden automatisch erzeugt, ohne dass weitere Argumente übergeben werden müssen. Die genutzten Services nutzen wiederum Default-Argumente, um ihre Abhängigkeiten herzustellen. Durch dieses Vorgehen kann für den Standardfall, auf den zusätzlichen Aufwand verzichtet werden, den die Implementierung eines Assemblers mit sich bringen würde. Durch die Übergabe anderer Argumente lassen sich die Objekte der Klasse jedoch weiterhin konfigurieren, ohne etwas an deren Quellcode ändern zu müssen. Wie dies auch im Rahmen des Testens genutzt wurde, wird in Abschnitt 5.1 genauer erläutert. Zwar sollte die Nutzung sogenannter Mutable Default Arguments in Python vermieden werden, da dies zu unvorhergesehenem Verhalten führen kann, im Falle zustandsloser Objekte und nicht-paralleler Verarbeitung, ist dies jedoch gefahrlos möglich.

Listing 8: Auszug aus dem RegistrationService: Konstruktor-Methode der Klasse

```
def __init__(self,
   caregiver_service: CareGiverService = CareGiverService(),
   caretaker_service: CareTakerService = CareTakerService(),
   relative_of_service: RelativeOfService = RelativeOfService(),
   transaction_service: TransactionService = TransactionService(),
   mail_generator: MailGenerator = MailGenerator()):

      self.caregiver_service = caregiver_service
      self.caretaker_service = caretaker_service
      self.relative_of_service = relative_of_service
      self.transaction_service = transaction_service
      self.mail_generator = mail_generator
```

4.4 PyQT5

Wie in Abschnitt 3.4 erläutert wird, bietet das Tool QT Designer die Möglichkeit Layout-Dateien zu erzeugen, welche zur Laufzeit durch das uic-Modul in Python-Code umgewandelt werden können. Listing 9 zeigt den gesamten Klassendefinition des ChooseIdsDialog zur Auswahl je einer CGID und CTID (s. Abbildung 14Abbildung 14: ChooseIdsDialog).

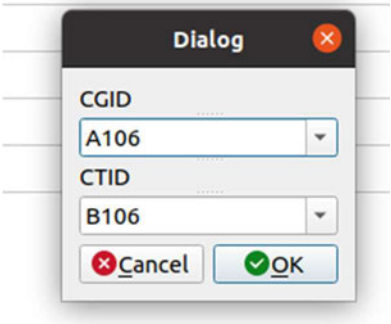


Abbildung 14: ChooseIdsDialog

Neben dem Laden der Layout-Datei muss lediglich die Vererbungsbeziehung zur Klasse QDialog definiert und die Combo-Boxen mit Werten befüllt werden. Alle weiteren Anpassungen können an der Layout-Datei vorgenommen werden, solange die Combo-Boxen weiterhin den Namen cgidComboBox und ctidComboBox tragen.

Listing 9: Auszug aus dem ChooseIdsDialog: Klassendefinition

```
class ChooseIdsDialog(QtWidgets.QDialog):
    def __init__(self, cgid_list, ctid_list):

        super().__init__()
        uic.loadUi(gui_config['choose_ids_dialog'], self)

        self.cgidComboBox.addItems(cgid_list)
        self.ctidComboBox.addItems(ctid_list)
```

Instanziiert wird der ChooseIdsDialog in der Methode choose_ids_and_insert_row der Komponente Mediator (siehe Listing 10). Die CGIDs bzw. CTIDs werden in dieser Methode über die jeweiligen Controller aus dem Applikationskern abgerufen und über den Konstruktor an den Dialog weitergegeben. Durch Aufruf der Methode exec wird der Dialog ausgeführt. Da der

Dialog in der Variable dialog referenziert wird, können die ausgewählten Werte ausgelesen werden, auch wenn das Dialogfenster nicht mehr angezeigt wird.

Listing 10: Auszug aus Mediator: Hinzufügen von Zeilen

```
def choose_ids_and_insert_row(self, model):
    """Opens a Dialog to Choose Ids and a Row with Given Ids
       into model
    """

    """Get all CGIDS and CTIDS"""
    cgids = [cg['cgid'] for cg in
       self.main_controller.caregiver_controller.retrieve_all()]
    ctids = [ct['ctid'] for ct in
        self.main_controller.caretaker_controller.retrieve_all()]

    """Init and open Dialog"""
    dialog = ChooseIdsDialog(cgids, ctids)

    if dialog.exec():
        cgid = dialog.cgidComboBox.currentText()
        ctid = dialog.ctidComboBox.currentText()
        self.insert_row_into_model(model, [cgid, ctid])
```

5 Qualitätssicherung

Die Maßnahmen zur Qualitätssicherung in der Entwicklung des AniTa-Tools gliedern sich in das Testen (siehe Abschnitt 5.1) und die Dokumentation (siehe Abschnitt 5.2). Während das Testen dabei unterstützen soll, die Funktionsfähigkeit des AniTa-Tools zu gewährleisten, dient die Dokumentation dazu sowohl externen Entwickler:innen als auch den Nutzer:innen den Umgang mit dem AniTa-Tool zu erleichtern.

5.1 Testen

Die Teststrategie des AniTa-Tools gliedert sich in drei aufeinander folgende Stufen. Auf der ersten Stufe sind die Unit-Tests angesiedelt. Im Rahmen der Unit-Tests werden die einzelnen Komponenten des AniTa-Tools in Isolation getestet. Auf der zweiten Stufe werden im Rahmen eines Integrations-Tests, die verschiedenen Komponenten in ihrem Zusammenspiel getestet. Auf der dritten Stufe stehen die Akzeptanztests, welche eine komplexe Folge von Aktionen testen, wie sie von den Nutzer:innen des AniTa-Tools durchgeführt werden. Während die Tests der ersten Stufe automatisch durchgeführt werden, werden die Akzeptanztests der dritten Stufe anhand definierter Szenarien manuell durchgeführt.

Durch die implementierten Unit- und Integrations-Tests wird eine Testabdeckung von 81% des Quellcodes der Fachdomäne erreicht. Da in den Integrations-Tests eine SQLite-Datenbankdatei eingebunden wird, wird so auch die Infrastrukturschicht in den Tests abgedeckt. Die Präsentationsschicht wird hingegen lediglich in den Akzeptanztests berücksichtigt.

Sowohl für die Unit-Tests als auch die Integrations-Tests wurde das Framework pytest verwendet (Krekel, n.d.). Um im Rahmen der Unit-Tests Komponenten in Isolation testen zu können, die über Abhängigkeiten zu anderen Komponenten verfügen, wurde das Python-Modul unittest.mock eingesetzt.

Listing 11 zeigt, wie Mock-Objekte eingesetzt werden, um die Methode create des CareGiver-Services zu testen. Hierfür wird je ein Mock-Objekt für das CareGiverRepository und den PlaceService erstellt. Für diese Mock-Objekte werden Mock-Methoden erstellt, die bei Aufruf durch den CareGiverService zuvor definierte Werte zurückgeben. Da in der Implementierung konsequent das Muster der Dependency Injection verfolgt wurde (s. Abschnitt 4.3), können die Objekte über den Konstruktor in das CareGiverService-Objekt injiziert werden.

Listing 11: Auszug aus den Tests des CareGiverServices: Einsatz von Mock-Objekten

```
@patch(
    'anita.repositories.caregiver_repository.CareGiverRepository'
)
@patch('anita.services.crud.place_service.PlaceService')
def test_caregiver_service_put(self, MockRepo, MockService):
    def side_effect(value):
        return value

    def side_effect2(zip_code):
        return True if zip_code is None \
            or zip_code is "22085" else False

    place_service = MockService()
    place_service.exists = Mock(side_effect=side_effect2)
    repo = MockRepo()
    repo.create = Mock(side_effect=side_effect)
    repo.get_ids.return_value = ["A100"]

    service = CareGiverService(
        repository=repo,
        place_service=place_service
    )
```

Während die Unit-Tests in der Struktur und Namensgebung den Komponenten des AniTa-Tools entsprechen, wurde für den Integrationstest ein größeres Szenario entwickelt, bei dem verschiedene Entitäten über die CRUDController, CRUDServices, CRUDRepositories in der Datenbank angelegt, verändert, abgerufen und gelöscht werden.

Für die manuellen Akzeptanztests wurden drei fiktive Anmeldemails und ein Szenario erzeugt, welches schrittweise durchgeführt wird. Dieses Szenario beinhaltet verschiedene Stories, die verschiedene notwendige Prozessschritte in der Bearbeitung von Anmeldungen bei der SP repräsentieren. Für jede Story werden Vorbedingungen (Given), eine durchzuführende Aktion

(When) und ein erwartetes Ergebnis (Then) definiert. Die für die Akzeptanztests erstellte Tabelle findet sich in Anhang A.

5.2 Dokumentation

Für das AniTa-Tool wurde neben der Readme sowohl eine technische Dokumentation als auch ein Dokumentation für die Benutzer:innen erstellt. Während die Readme allgemeine Informationen zum Erstellen des Builds und Starten des AniTa-Tools enthält, hält die technische Dokumentation Informationen zu technischen Details vor. Die technische Dokumentation wurde mit der Python-Bibliothek pdoc3 erstellt, mittels derer aus dem bestehenden Quellcode und Multiline-Kommentaren, sogenannten docstrings, eine Dokumentation im HTML-Format erstellt werden kann (kernc, n.d.). In Listing 12 ist exemplarisch der docstring in numpydoc-Stil (numpydoc maintainers, 2019) der Methode get_potential_matches des MatchingService dargestellt.

Listing 12: Auszug aus dem MatchingService: docstring der Methode get_potential_matches

```
"""
Returns a list of all potential Matches
If since_date is given, potential mathches are only checked for
CareGivers with date of registration >= since date

Parameters
----------
since_date: A datetime.date object
deviation: A float used in the Connector as a factor for
the Radius

Returns
-------
A List of all potential Matches
"""
```

Bei der Dokumentation für die Benutzer:innen handelt es sich um ein Benutzerhandbuch, im PDF-Format, welches anhand von Beschreibungen und Screenshots, die einzelnen Funktionen des AniTa-Tools erläutert. Das Benutzerhandbuch befindet sich in Anhang B.

6 Evaluation

Während die Evaluation der Architektur und einzelner Designentscheidungen im Laufe des Kapitels 3 besprochen wurde, werden in den folgenden drei Abschnitten der Erfüllungsgrad der Anforderungen (Abschnitt 6.1), der finale Stand der Prozessautomatisierung (Abschnitt 6.2) sowie der Matching-Algorithmus (Abschnitt 6.3) evaluiert.

6.1 Anforderungen

Zur Beschreibung der Anforderungen des AniTa-Tools wurden neun User-Stories zu funktionalen und sieben User-Stories zu nichtfunktionalen Anforderungen verfasst. Während die funktionalen Anforderungen größtenteils vollständig oder zumindest teilweise erfüllt werden konnten, war es für einige Akzeptanzkriterien nicht möglich, diese zu überprüfen. Einen Überblick darüber, wie viele Akzeptanzkriterien für die jeweilige User Story erfüllt sind, gibt Tabelle 13. Dass ein Großteil der Anforderungen erfüllt werden konnte, ist nicht zuletzt dadurch bedingt, dass die Anforderungen im Verlauf der Entwicklungstätigkeiten an die neuen Erkenntnisse angepasst wurden.

Tabelle 13: Übersicht über den Erfüllungsgrad der einzelnen Anforderungen

User Story	***Beschreibung***	***Akzeptanzkriterien erfüllt?***			
		Anzahl	Voll	Teilweise	Nicht
US1	Erzeugung von Bestätigungsmails	4	4	0	0
US2	Einpflegen von Anmeldedaten	4	4	0	0
US3	Anzeigen von Stammdaten	2	2	0	0
US4	Änderung von Stammdaten	3	2	1	0
US5	Löschen von Stammdaten	3	2	1	0
US6	Automatisches Matching von Teilnehmenden	4	3	1	0
US7	Anzeigen von Matches	1	1	0	0
US8	Änderung von Matches	1	1	0	0
US9	Visualisierung von Anmeldungen	2	2	0	0
US10	Portabilität des AniTa-Tools	1	0-1	0-1	0-1
US11	Änderbarkeit der Funktionalitäten des AniTa-Tools	3	1-3	0-2	0-2
US12	Austausch von Komponenten	2	0-2	0-2	0-2
US13	Übereinstimmung mit Datenschutzerklärung	2	2	0	0
US14	Schutz vor Datendiebstahl	1	1	0	0
US15	Wiederherstellbarkeit der Stammdaten	2	2	0	0
US16	Erlernbarkeit des AniTa-Tools	2	1-2	0-1	0-1

Da sich Änderungen an Stammdaten bzw. das Löschen von Teilnehmenden nur auf den letzten gespeicherten Stand und nicht im Sinne einzelner Aktionen zurücksetzen lässt, wurde das entsprechende Akzeptanzkriterium der US4 und US5 nur teilweise erfüllt. Zudem wurden im Rahmen des Matchings angrenzende Postleitzahlengebiete nur indirekt über die jeweilige Fläche des Postleitzahlengebietes berücksichtigt (siehe User Story 6).

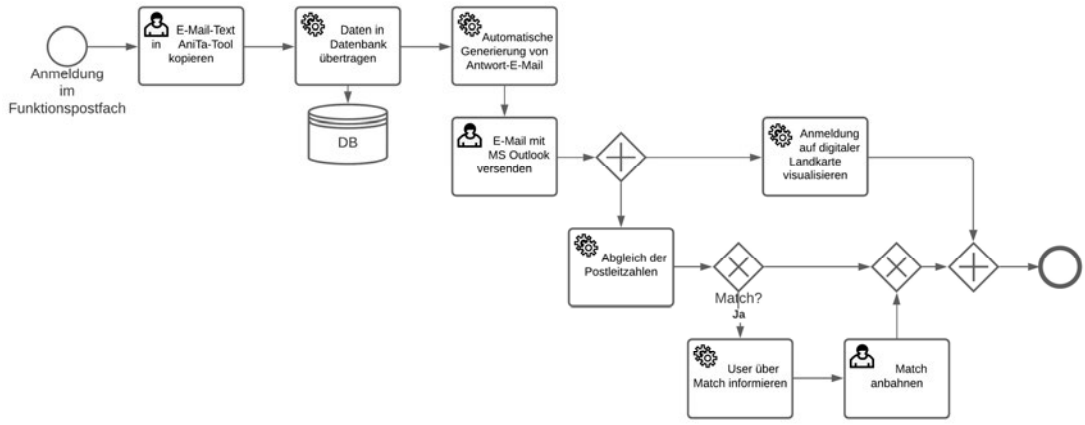
Seitens der nichtfunktionalen Anforderungen wurde nicht überprüft, ob das AniTa-Tool auf anderen Betriebssystemen als Window 10 lauffähig ist, wie in US10 gefordert. Ebenso konnte nicht geprüft werden, ob sich in dem geforderten Zeitaufwand Funktionalitäten ändern (US11) oder Komponenten austauschen lassen (US12). Zudem konnte nicht überprüft werden, ob das AniTa-Tool leicht zu erlernen ist, wie in US16 gefordert wird. Dies ist dadurch begründet, dass das AniTa-Tool nicht mehr im Betrieb der SP eingesetzt wurde. Die Übereinstimmung mit der Datenschutzerklärung (US13), sowie der Schutz vor Datendiebstahl (US14) und die

Wiederherstellbarkeit der Stammdaten (US15) konnten hingegen dadurch erfüllt werden, dass auf die vorhandenen Systeme der HAW zurückgegriffen wurde.

6.2 Prozessautomatisierung

Vergleicht man den in Abbildung 2 dargestellten Soll-Prozess im Betrieb der SP mit dem in Abbildung 15: Darstellung des potenziellen neuen Prozesses unter Einsatz des AniTa-ToolsAbbildung 15 dargestellten potenziellen Prozess unter Verwendung des AniTa-Tools, wird deutlich, dass durch das AniTa-Tool keine vollständige Automatisierung des Prozesses erreicht wird. Vielmehr sind an verschiedenen Stellen Aktionen der Nutzer:innen notwendig. Hierbei handelt es zum einen um die manuellen Aktionen, die notwendig sind, um den Text der Anmeldemail in das AniTa-Tool zu kopieren und die Bestätigungsmail mittels MS Outlook zu versenden. Zum anderen müssen die Anmeldedaten durch die Mitarbeiter:innen darauf geprüft werden, ob diese augenscheinlich korrekt sind. Da im Rahmen des Matching-Algorithmus alle potenziellen Matches erfasst werden, liegt es zudem in der Verantwortung der Projektmitarbeiter:innen die vielversprechendsten Tauschpaare priorisiert zu kontaktieren, sollten Teilnehmende in mehreren potenziellen Matches vorkommen.

Abbildung 15: Darstellung des potenziellen neuen Prozesses unter Einsatz des AniTa-Tools



Zwar konnte keine vollständige Automatisierung der Geschäftsprozesse erzielt werden, es ist jedoch davon auszugehen, dass sich durch die Nutzung des AniTa-Tools die Effizienz in der Bearbeitung der Anmeldungen steigern lässt. Zudem ist anzunehmen, dass auch sich auch die

Ergebnisqualität verbessert, da Übertragungsfehler reduziert und weniger potenzielle Matches übersehen werden. So konnten durch Einsatz des implementierten Matching-Algorithmus auf den Daten von 27 CG und 20 CT 12 potenzielle Matches identifiziert werden, die beim manuellen Abgleich nicht entdeckt wurden.

6.3 Matching-Algorithmus

Da lediglich die Konstruktion des Graphen als Basis für den Matching-Algorithmus im AniTa-Tool implementiert ist, wurde eine Simulation durchgeführt, um erste Anhaltspunkte über das Verhalten des gesamten Matching-Algorithmus, wie er in Abschnitt 3.3.6 beschrieben wird, zu erhalten. Hierzu wurde der Betrieb der SP modelliert. Da die Komplexität der Realität des Betriebes der SP zu hoch ist, um diese in einem Modell zu erfassen, wurden hierfür vereinfachende Annahmen getroffen. So werden nur Anmeldungen von CareGiver-CareTaker-Paaren berücksichtigt und jedes Match, welches im Sinne der Simulation angebahnt wird, kommt auch zu Stande. Teilnehmende scheiden also weder mit der Zeit aus der Tauschbörse aus, noch lehnen sie ein Match ab.

Die Simulation soll folgende Fragestellungen adressieren:

Fragestellung 1: Wie entwickelt sich das Verhältnis zwischen potenziellen und aktiven Matches mit steigender Teilnehmendenzahl?

Fragestellung 2: Wie entwickelt sich das Verhältnis zwischen suchenden Familien und aktiven Matches mit steigender Teilnehmendenzahl?

Fragestellung 3: Wie verändert sich die durchschnittliche Zeit von der Anmeldung bis zum Match bzw. vom Match eines Familienmitglieds bis das andere gematched ist mit steigender Teilnehmendenzahl?

Fragestellung 4: Wie ist das Laufzeitverhalten der Konstruktion des Graphen und des Matching-Algorithmus?

Während sich die ersten drei Fragestellungen der übergeordneten Frage widmen, ob sich unter den Annahmen der Simulation ähnliche Dynamiken ergeben, wie sie für die

Plattformökonomie beschrieben werden (siehe Abschnitt 1.3), liegt der Fokus der vierten Fragestellung auf der technischen Umsetzung des Matching-Algorithmus.

6.4 Ablauf der Simulation

Die Simulation läuft als eine Abfolge von Ticks ab. Jeder Tick besteht aus den folgenden vier Schritten:

1. Generierung ein neues CareGiver-CareTaker-Paares.
2. Konstruktion des Graphen der potenziellen Matches aus allen CareGiver-CareTaker-Paaren.
3. Ausführung des Matching-Algorithmus auf dem Graphen der potenziellen Matches.
4. Setzen des Status aller Matches im Ergebnis des Matching-Algorithmus auf „aktiv“.

Während Schritt zwei und drei wie in Abschnitt 3.3.6 beschrieben ablaufen, erfolgt die Generierung der CareGiver-CareTaker-Paare wie in Listing 13 erläutert.

Listing 13: Ablauf der Generierung der CareGiver-CareTaker-Paare in der Simulation

```
1. Ziehe zufälliges Postleitzahlengebiet(plz) x für CG aus der
   Menge aller Postleitzahlengebiete (PLZ) mit Wahrscheinlich-
   keit w(x) = Population(x) / sum(Population(plz) für alle plz
   in PLZ)
2. Ziehe zufälligen Aktivitätsradius des CG mit Normalverteil-
   lung N(5, 15) im Intervall [0, 50].
3. Ziehe zufällige Entfernung e mit Wahrscheinlichkeit p aus
   [(e=100, p=.075), (200, .125), (300, .15), (400, .3), (500,
   .15), (600, .125), (700, .075)]
4. Ziehe zufälliges plz y für CT aus allen plz mit Entfernung e
   zu x (PLZex) mit Wahrscheinlichkeit w(y) = Population(y) /
   sum(Population(plz) für alle plz in PLZex). Wenn keine PLZex
   existieren, gehe zu 3.
```

Da wie beschrieben an verschiedenen Schritten der Simulation Zufallsziehungen durchgeführt werden, wird das Ergebnis einzelner Durchläufe durch den Zufall beeinflusst. Um diesen Einfluss zu reduzieren, wurden 20 Durchläufe der Simulation über je 1000 Ticks durchgeführt.

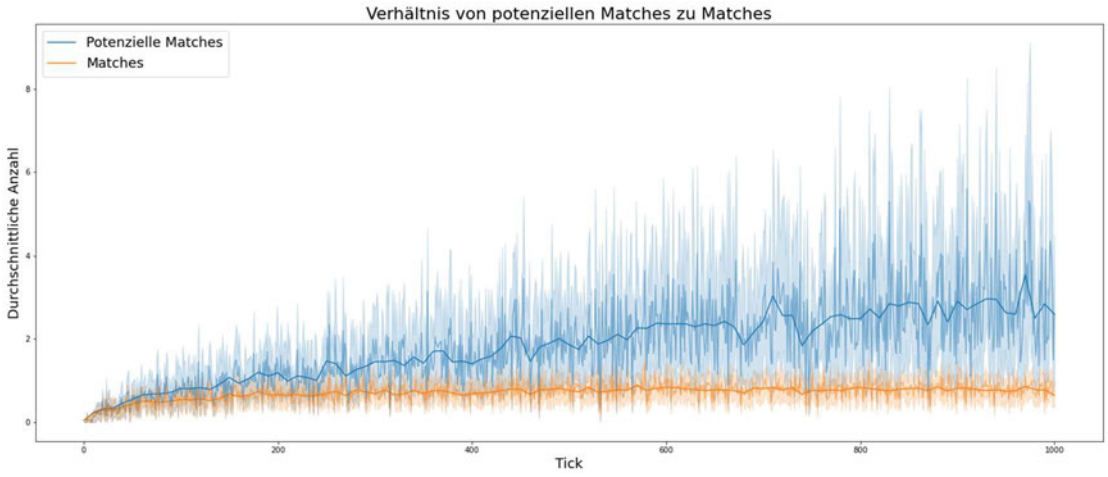
6.5 Ergebnisse der Simulation

Da die Ergebnisse auf mehreren Durchläufen der Simulation beruhen, werden die Kennwerte als Durchschnitt über die Durchläufe dargestellt.

Fragestellung 1

Wie in Abbildung 16 dargestellt ist, steigt die durchschnittliche Anzahl der potenziellen Matches mit zunehmenden Ticks bzw. Anmeldungen leicht an, übersteigt jedoch nur selten eine Anzahl von vier. Über alle Simulationen beträgt der Mittelwert der Anzahl der potenziellen Matches M = 1.82 mit einer Standardabweichung von SD = 2.92, bei einem Range von r = (0, 34). Während die Anzahl der angebahnten Matches zunächst parallel zur Anzahl der potenziellen Matches steigt, kommt es ab etwa 100 Ticks nicht mehr zu einem nennenswerten Anstieg. Im Durchschnitt werden pro Tick M = 0.71 (SD = 0.71) Matches angebahnt (r = (0,2)).

Abbildung 16: Verhältnis von potenziellen Matches zu aktiven Matches pro Tick



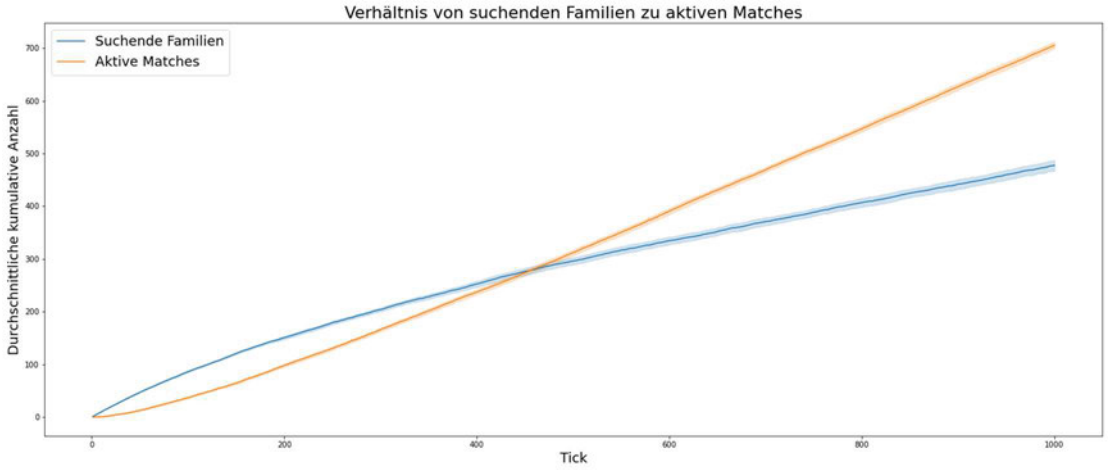
Anmerkung. Für beide Farben werden je drei Linien in unterschiedlicher Stärke dargestellt. Die stärkste Linie im Vordergrund stellt den über je 10 Ticks geglätteten Verlauf dar. Die mittelstarke Linie, stellt den nicht geglätteten Verlauf dar und die schwächste Linie im Hintergrund die Standardabweichung des nicht geglätteten Verlaufs.

Die beschriebenen Ergebnisse deuten darauf hin, dass zu den einzelnen Ticks in der Regel einzelne Teilnehmende in mehreren potenziellen Matches vorkommen und somit aus diesen nur wenige Matches angebahnt werden können.

Fragestellung 2

Als Ergebnis der Simulation zeigt sich in Abbildung 17, dass die Anzahl der suchenden Familien zu Beginn der Simulationen stärker ansteigt als die Zahl der aktiven Matches. Dies ändert sich jedoch bereits ab ca. 100 Ticks, sodass im Durchschnitt ab Tick 456 mehr aktive Matches als suchende Familien vorliegen.

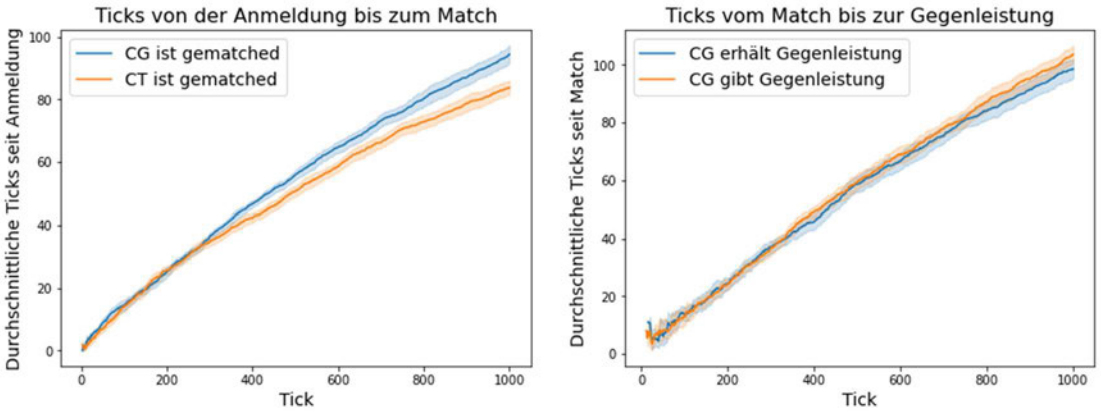
Abbildung 17: Entwicklung des Verhältnisses von suchenden Familien zu aktiven Matches



Fragestellung 3

In Abbildung 18 ist in der linken Grafik dargestellt, wie viele Ticks durchschnittlich vergehen, bis ein CG bzw. CT gematched ist. Die rechte Grafik der Abbildung zeigt, wie viele Ticks durchschnittlich vergehen bis ein bereits gematchter CG eine Gegenleistung erhält bzw. bis ein CG mit gematchtem CT in Gegenleistung geht. In beiden Grafiken zeigt sich, dass die durchschnittliche Anzahl von Ticks mit voranschreitender Simulationsdauer ansteigt, die Steigung jedoch abflacht. Es scheint daher nicht ausgeschlossen, dass bei steigender Simulationsdauer ein Plateau erreicht werden würde.

Abbildung 18: Ticks von der Anmeldung zum Match bzw. vom Match zur Gegenleistung

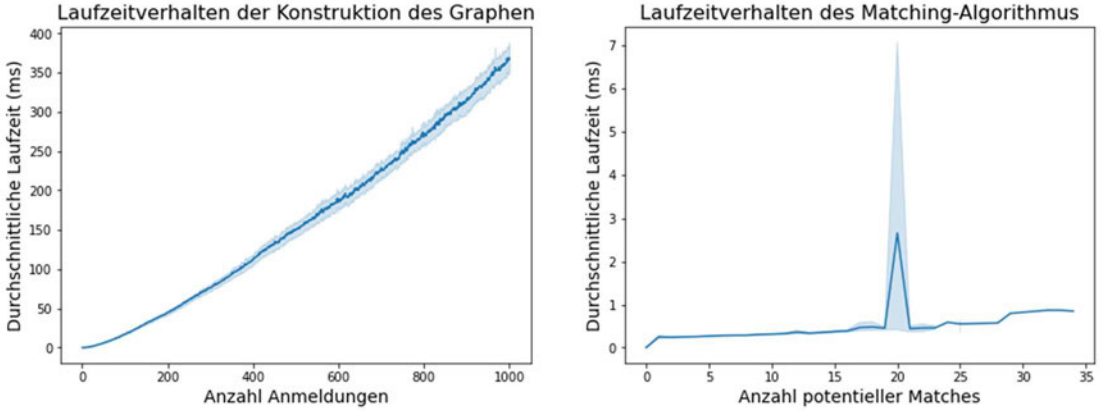


Anmerkung. Der linke Teil der Grafik zeigt, wie viele Ticks durchschnittlich zwischen Anmeldung und aktivem Match eines CG bzw. CT liegen. Der rechte Teil der Grafik zeigt, wie viele Ticks seit dem Match vergehen bis der CT eines bereits gematchten CG gematcht ist (CG erhält Gegenleistung) bzw. bis der CG eines bereits gematchten CT gematcht ist (CG gibt Gegenleistung).

Fragestellung 4

Das in der linken Grafik der Abbildung 19 dargestellte Laufzeitverhalten der Konstruktion des Graphen deutet auf ein quadratisches Laufzeitverhalten hin, was durchaus zur Implementierung im Rahmen der Simulation passt. Da lediglich ein Maximum von 34 potenziellen Matches erreicht wurde, kann zum Laufzeitverhalten des implementierten Matching-Algorithmus auf Basis der Simulation keine Aussage getätigt werden.

Abbildung 19: Laufzeitverhalten der Konstruktion des Graphen und des Matching-Algorithmus



6.5.2 Diskussion der Ergebnisse

Die Ergebnisse zu den ersten drei Fragestellungen deuten auf eine ähnliche Dynamik hin, wie sie in Abschnitt 1.3 für die Plattformökonomie beschrieben wurde. Für eine genauere Analyse der Dynamik sollte die Simulation in einem nächsten Schritt über eine größere Anzahl von Ticks laufen. Inwieweit die Ergebnisse der Simulation auf die Realität übertragbar sind, hängt im großen Maß davon ab, ob die vereinfachten Annahmen haltbar sind. So ist die Annahme, dass jedes Match zustande kommt, wohl sehr optimistisch, da erwartbar ist, dass Teilnehmende mit der Zeit nicht mehr aktiv an der Tauschbörse teilnehmen oder aus verschiedenen Gründen eine Tauschbeziehung ablehnen werden. Um die Simulationsergebnisse übertragen zu können, müssten daher in möglichst kurzer Zeit viele Anmeldungen an der Tauschbörse erfolgen und die Passung zwischen potenziellen Tauschpaaren möglichst präzise quantifiziert werden.

Bezüglich des Laufzeitverhaltens zeigt sich, dass ein Aufbau des gesamten Graphs für jeden Durchlauf des Matchings mit steigender Teilnehmendenzahl nicht optimal ist. Vielmehr sollte mit steigender Teilnehmendenzahl darüber nachgedacht werden, den Graphen zu persistieren und lediglich neue Anmeldungen als Knoten in diese bestehende Struktur einzufügen. Hierfür würden sich etwa Graph-Datenbanken anbieten. Um die Laufzeit des eigentlichen Matching-Algorithmus zu reduzieren, könnten zunächst die Zusammenhangskomponenten ermittelt und dann der Algorithmus auf je Zusammenhangskomponente durchgeführt werden.

7 Fazit

Im Rahmen dieser Arbeit wurde beschrieben, wie in einem iterativen Prozess von Anforderungsanalyse, Architekturentwurf, Implementierung und Evaluation ein Softwaresystem entwickelt wurde, welches die Projektmitarbeiter:innen des Projektes AniTa im Betrieb der SP AniTa unterstützt. Das ursprüngliche Ziel einer vollständigen Geschäftsprozessautomatisierung konnte hierbei aufgrund infrastruktureller sowie organisatorisch-rechtlicher Rahmenbedingungen nicht erreicht werden. Es kann jedoch davon ausgegangen werden, dass der Einsatz des AniTa-Tools die Bearbeitungen der Anmeldungen an der SP signifikant reduzieren kann. Das AniTa-Tool ist zudem so entworfen und implementiert worden, dass die bestehenden Komponenten so modifiziert bzw. erweitert werden können, dass auf geänderte Rahmenbedingungen reagiert und letztendlich auch eine vollständige Automatisierung erreicht werden kann.

Literaturverzeichnis

Agafonkin, V., 2021. Leaflet — an open-source JavaScript library for interactive maps [WWW Document]. URL https://leafletjs.com/ (accessed 8.18.21).

Anaya, M., 2020. The SOLID Principles, in: Clean Code in Python: Develop Maintainable and Efficient Code, Expert Insight. Packt, Birmingham - Mumbai, pp. 125–156.

Asratian, A.S., Denley, T.M.J., Häggkvist, R., 1998. Bipartite Graphs and Their Applications. Cambridge University Press.

Balzert, H., 2009. Anforderungen und Anforderungsarten, in: Balzert, H. (Ed.), Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering. Spektrum Akademischer Verlag, Heidelberg, pp. 455–474. https://doi.org/10.1007/978-3-8274-2247-7_16

Bayer, M., 2012. SQLAlchemy, in: Brown, A., Wilson, G. (Eds.), The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks. aosabook.org, Mountain View.

Bayer, M., n.d. Welcome to Alembic's documentation! — Alembic 1.6.5 documentation [WWW Document]. URL https://alembic.sqlalchemy.org/en/latest/ (accessed 8.21.21).

Becker, M.J., Riedl, W.F., Aleksejs Voroncovs, 2015. Die Ungarische Methode [WWW Document]. URL https://www-m9.ma.tum.de/graph-algorithms/matchings-hungarian-method (accessed 8.18.21).

Burtch, G., Ramaprasad, J., 2016. Assessing and Quantifying Local Network Effects in an Online Dating Market (SSRN Scholarly Paper No. ID 2848515). Social Science Research Network, Rochester, NY. https://doi.org/10.2139/ssrn.2848515

Crispin, M., 2003. INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1.

Duerst, M., Masinter, L., Zawinski, J., 2010. The "mailto" URI Scheme.

Freund, J., Rücker, B., 2019. Praxishandbuch BPMN 2.0: Mit Einführung in DMN, 6th ed. Carl Hanser Verlag GmbH & Co. KG, München. https://doi.org/10.3139/9783446461123

Galil, Z., 1986. Efficient algorithms for finding maximum matching in graphs. ACM Comput. Surv. CSUR 18, 23–38. https://doi.org/10.1145/6462.6502

Gartner, 2021. Definition of Business Process Automation (BPA) - Gartner Information Technology Glossary [WWW Document]. Gartner. URL https://www.gartner.com/en/information-technology/glossary/bpa-business-process-automation (accessed 7.26.21).

Geyer, J., Schulz, E., 2014. Who cares? Die Bedeutung der informellen Pflege durch Erwerbstätige in Deutschland. DIW Wochenber. 81, 294–301.

Hagiu, A., Wright, J., 2015. Multi-sided platforms. Int. J. Ind. Organ. 43, 162–174.

kernc, n.d. pdoc – Auto-generate API documentation for Python projects [WWW Document]. URL https://pdoc3.github.io/pdoc/ (accessed 9.12.21).

Klensin, J., 2008. Simple Mail Transfer Protocol.

Krekel, H., n.d. pytest: helps you write better programs — pytest documentation [WWW Document]. URL https://docs.pytest.org/en/6.2.x/ (accessed 8.25.21).

Kricheldorff, C., Franke, A., Bischofberger, I., Otto, U., 2019. „Distance caregiving" – Pflege bei räumlicher Distanz. Z. Für Gerontol. Geriatr. 52, 519–520. https://doi.org/10.1007/s00391-019-01612-5

Kuhn, H.W., 1955. The Hungarian method for the assignment problem. Nav. Res. Logist. Q. 2, 83–97.

Lutz, M., 2013. Part I. Getting Started, in: Learning Python: Powerful Object-Oriented Programming. O'Reilly Media, Incorporated, Sebastopol, UNITED STATES, pp. 1–92.

Martin, R.C., 2012. The Clean Architecture [WWW Document]. Clean Code Blog. URL https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html (accessed 8.4.21).

Microsoft Corporation, 2021. [MS-SMB2]: Server Message Block (SMB) Protocol Versions 2 and 3 [WWW Document]. URL https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-smb2/5606ad47-5ee0-437a-817e-70c366052962 (accessed 7.27.21).

Navidi, Z., Nagel, K., Winter, S., 2020. Toward identifying the critical mass in spatial two-sided markets. Environ. Plan. B Urban Anal. City Sci. 47, 1704–1724. https://doi.org/10.1177/2399808319842181

NetworkX developers, 2021. NetworkX — NetworkX documentation [WWW Document]. URL https://networkx.org/ (accessed 8.18.21).

Nowossadeck, S., Engstler, H., Klaus, D., 2016. Pflege und Unterstützung durch Angehörige.

numpydoc maintainers, 2019. Style guide — numpydoc v1.2.dev0 Manual [WWW Document]. URL https://numpydoc.readthedocs.io/en/latest/format.html (accessed 9.12.21).

Object Management Group, 2014. Business Process Model and Notation.

Open Knowledge Foundation, n.d. Open Data Commons Open Database License (ODbL) v1.0 — Open Data Commons: legal tools for open data [WWW Document]. URL https://opendatacommons.org/licenses/odbl/1-0/ (accessed 8.21.21).

Opendatasoft, 2019. Postleitzahlen - Germany [WWW Document]. URL https://public.opendatasoft.com/explore/dataset/georef-germany-postleitzahl/information/ (accessed 1.20.20).

Python Software Foundation, 2021. 3.8.11 Documentation [WWW Document]. URL https://docs.python.org/3.8/ (accessed 7.30.21).

R. Fielding, E., J. Reschke, E., 2014. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing.

Riverbank Computing, 2021a. Introduction — PyQt v5.15 Reference Guide [WWW Document]. PyQt V515 Ref. Guide. URL https://www.riverbankcomputing.com/static/Docs/PyQt5/introduction.html#pyqt5-components (accessed 8.7.21).

Riverbank Computing, 2021b. uic — PyQt v5.15 Reference Guide [WWW Document]. URL https://www.riverbankcomputing.com/static/Docs/PyQt5/api/uic/uic-module.html?highlight=uic#uic (accessed 8.8.21).

Rohweder, J.P., Kasten, G., Malzahn, D., Piro, A., Schmid, J., 2021. Informationsqualität – Definitionen, Dimensionen und Begriffe, in: Hildebrand, K., Gebauer, M., Mielke, M. (Eds.), Daten- und Informationsqualität: Die Grundlage der Digitalisierung. Springer Fachmedien, Wiesbaden, pp. 23–43. https://doi.org/10.1007/978-3-658-30991-6_2

Schuchert, B.L., 2013. DIP in the Wild [WWW Document]. martinfowler.com. URL https://martinfowler.com/articles/dipInTheWild.html (accessed 8.4.21).

Schwochow, M., 2019. Kostenlose Postleitzahlenkarte [WWW Document]. URL https://www.suche-postleitzahl.org/plz-karte-erstellen (accessed 1.20.20).

SQLAlchemy authors and contributors, 2021. Working with Database Metadata — SQLAlchemy 1.4 Documentation [WWW Document]. URL https://docs.sqlalchemy.org/en/14/tutorial/metadata.html (accessed 9.1.21).

SQLAlchemy authors and contributors, n.d. Object Relational Tutorial — SQLAlchemy 1.3 Documentation [WWW Document]. URL https://docs.sqlalchemy.org/en/13/orm/tutorial.html (accessed 8.15.21a).

SQLAlchemy authors and contributors, n.d. Session Basics — SQLAlchemy 1.3 Documentation [WWW Document]. URL https://docs.sqlalchemy.org/en/13/orm/session_basics.html#what-does-the-session-do (accessed 8.15.21b).

Starke, G., 2020a. Architektur und Architekten, in: Effektive Softwarearchitekturen. Carl Hanser Verlag GmbH & Co. KG, pp. 15–32. https://doi.org/10.3139/9783446465893.002

Starke, G., 2020b. Strukturentwurf, Architektur- und Designmuster, in: Effektive Softwarearchitekturen. Carl Hanser Verlag GmbH & Co. KG, pp. 141–183. https://doi.org/10.3139/9783446465893.002

Story, R., 2013. Folium — Folium 0.12.1 documentation [WWW Document]. URL http://python-visualization.github.io/folium/ (accessed 8.18.21).

Tarjan, R.E., 1983. Matchings, in: Data Structures and Network Algorithms. SIAM, Philadelphia, pp. 113–125.

The Qt Company, 2021a. Qt Designer Manual [WWW Document]. Qt Des. Man. URL https://doc.qt.io/qt-5/qtdesigner-manual.html (accessed 8.8.21).

The Qt Company, 2021b. Model/View Programming | Qt Widgets 5.15.5 [WWW Document]. URL https://doc.qt.io/qt-5/model-view-programming.html (accessed 8.11.21).

The SQLite Consortium, 2021. SQLite Home Page [WWW Document]. URL https://www.sqlite.org/index.html (accessed 8.3.21).

The SQLite Consortium, n.d. About SQLite [WWW Document]. URL https://www.sqlite.org/about.html (accessed 8.3.21).

van Rossum, G., Lehtosalo, J., Langa, L., 2014. PEP 484 -- Type Hints [WWW Document]. Python.org. URL https://www.python.org/dev/peps/pep-0484/ (accessed 7.30.21).

W3C, 2017. HTML 5.2: 4.10. Forms [WWW Document]. URL https://www.w3.org/TR/html52/sec-forms.html#constraints (accessed 7.28.21).

Wirdemann, R., Mainusch, J., 2017. User Stories, in: Scrum Mit User Stories. Carl Hanser Verlag GmbH & Co. KG, pp. 49–65. https://doi.org/10.3139/9783446450776.004

Woock, K., Meinert, N., Völtzer, L., Nordholt, P., Busch, S., 2020. AniTa - Angehörige von älteren Menschen mit Pflege-/Unterstützungsbedarf im (Aus-)Tausch. Eine überregionale Plattform zur Aktivierung ungenutzten Betreuungspotentials. Abschlussbericht.

Zentgraf, A., Jann, P.M., Myrczik, J., van Holten, K., 2019. Pflegen auf Distanz? Z. Für Gerontol. Geriatr. 52, 539–545. https://doi.org/10.1007/s00391-019-01607-2

A Anhang: Akzeptanztests

Seq	Story	Given	When	Then	Test Result
0	Registrieren eines neuen CG-CT Paares aus einer E-Mail	Software ist geöffnet	Nutzer verwendet Registrierungsworkflow mit Anmeldemail korrekt 1	In Tabelle Caregiver, Caretaker und RelativeOf sind die Daten korrekt abgelegt, E-Mail wird korrekt erstellt	
1	Registrieren eines neuen Cgs aus einer E-Mail	Seq 0	Nutzer verwendet Registrierungsworkflow mit Anmeldemail korrekt 2	In Tabelle Caregiver sind Daten korrekt abgelegt, E-Mail wird korrekt erstellt	
2	Schließen der Applikation	Seq 1	Nutzer Schließt die Applikation über x	Nutzer wird über ungespeicherte Änderungen Informiert	
3	Schließen der Applikation	Seq 2	Nutzer speichert und beendet Applikation	Applikation ist geschlossen	
4	Öffnen der Applikation	Seq 3	Nutzer öffnet Applikation	In Tabelle Caregiver, Caretaker und RelativeOf sind die	

				Daten korrekt abgelegt	
5	Registrierung aus E-Mail	Seq 4	Nutzer verwendet Registrierungsflow mit Anmeldemain inkorrekt 2	Nutzer wird auf inkorrekte Postleitzahlen aufmerksam gemacht	
6	Registrierung ohne PLZ	Seq 5	Nutzer leert Felder, Nutzer beendet Workflow	Daten sind in Cargiver, Caretaker und RelativeOf korrekt abgelegt.	
7	Speichern	Seq 6	Nutzer speichert Daten und schließt die Applikation	Applikation schließt	
8	Matching	Seq 7	Nutzer startet Matching	Nutzer wird über Match zwischen CG mit Zip 26789 und CT mit Zip 26835 Informiert	
9	Ändern von Daten	Seq 8	Nutzer öffnet Applikation	Daten liegen unverändert vor	
10	Ändern von Daten	Seq 9	Nutzer ändert PLZ von CG 26788 und von CT auf 20099	Änderungen werden in entsprechenden Tabellen angezeigt	
11	Matching	Seq 10	Nutzer bestehenden Match auf active	Änderung wird in Tabelle angezeigt	
12	Matching	Seq 11	Nutzer startet Matching	Match zwischen CG 22085 und 20099 wird angezeigt, sonst keine Matches	

13	Matching	Seq 12	Nutzer setzt aktiven Match auf cancelled	Änderung wird in Tabelle angezeigt	
14	CT hinzufügen	Seq 13	Nutzer fügt CG ohne CT einen CT in 22303 hinzu	Änderungen in Tabelle werden angezeigt	
15	Matching	Seq 14	Nutzer startet Matching	Match zwischen CG 22085 Radius 5 und CT 22303 wird angzeigt	
16	Schließen und speichern	Seq 15	Nutzer schließt die Applikation und speichert nach aufforderung	Die Applikation schließt	
17	Visualisieren	Seq 16	Nutzer öffnet Karte	Alle Cgs und Cts werden korrekt auf der Karte dargestellt	
18	Löschen	Seq 17	Nutzer löscht CG mit Zip 22085	Zugehörige Daten werden aus Caregiver, RelativeOf und Match-Tabelle gelöscht	
19	Wiederherstellen	Seq 18	Nutzer setzt die Änderung Zurück	Zugehörige Daten werden sind wieder in Caregiver, RelativeOf und Match-Tabelle	
20	Löschen	Seq 19	Nutzer löscht alle Datensätze, speichert und schließt die Applikation	Applikation schließt	

21	Öffnen	Seq 20	Nutzer öffnet die Applikation	Alle Tabellen sind leer	

Anmeldemail korrekt, 1: CG (Zip, 22085, Radius 0), CT (Zip, 26835)

Anmeldemail korrekt, 2: CG (Zip, 26789; Radius, 15)

Anmeldemail inkorrekt, 2: CG (Zip, 26788!, Radius 5), CT (Zip, 20090!)

B Anhang: Benutzerhandbuch

AniTa-Tool

- Benutzerhinweise -

Das AniTa-Tool wurde im Rahmen des Projektes AniTa an der Hochschule für Angewandte Wissenschaften in Hamburg zum internen Gebrauch entwickelt. Aufgrund der Covid-19-Pandemie, ist eine Einführung der Software im Projektbetrieb jedoch nicht erfolgt. Im Folgenden ist illustriert, wie das AniTa-Tool genutzt werden kann, um einzelne Prozessschritte der Anmeldung bei der Tauschbörse (www.anita-famile.de) und dem Matching der Teilnehmenden, zu unterstützen.

Lizenz

Einrichtung der Software

Vor dem ersten Start der Software müssen einige wenige Einstellungen vorgenommen werden. Siehe hierzu auch *Abbildung* .

1. **Die Sqlite-Datenbank muss angebunden werden:** Hierzu öffnet man im Dateiordner der gebuildeten Software den ordner anita/resources (1.). Hier befindet sich die Datei config.yml (2.) in dieser Datei kann man das Profil unter profile einstellen. Die Pfade zur Auswahl der Sqlite Tabelle müssen unter db:prodction:url bzw. db:development:url an das lokale System angepasst werden (3.). Zur Erzeugung einer Datenbank siehe README.MD im Order anita.

2. **Das Profil muss ggf. auf ‘‘PRODUCTION‘‘ gesetzt werden** (4.)

Start der Software

Um die Software zu Starten genügt ein Doppelklick auf die Datei „anita_tool.exe“ im Ordner „anita_tool“. Der Startvorgang kann ein wenig Zeit in Anspruch nehmen.

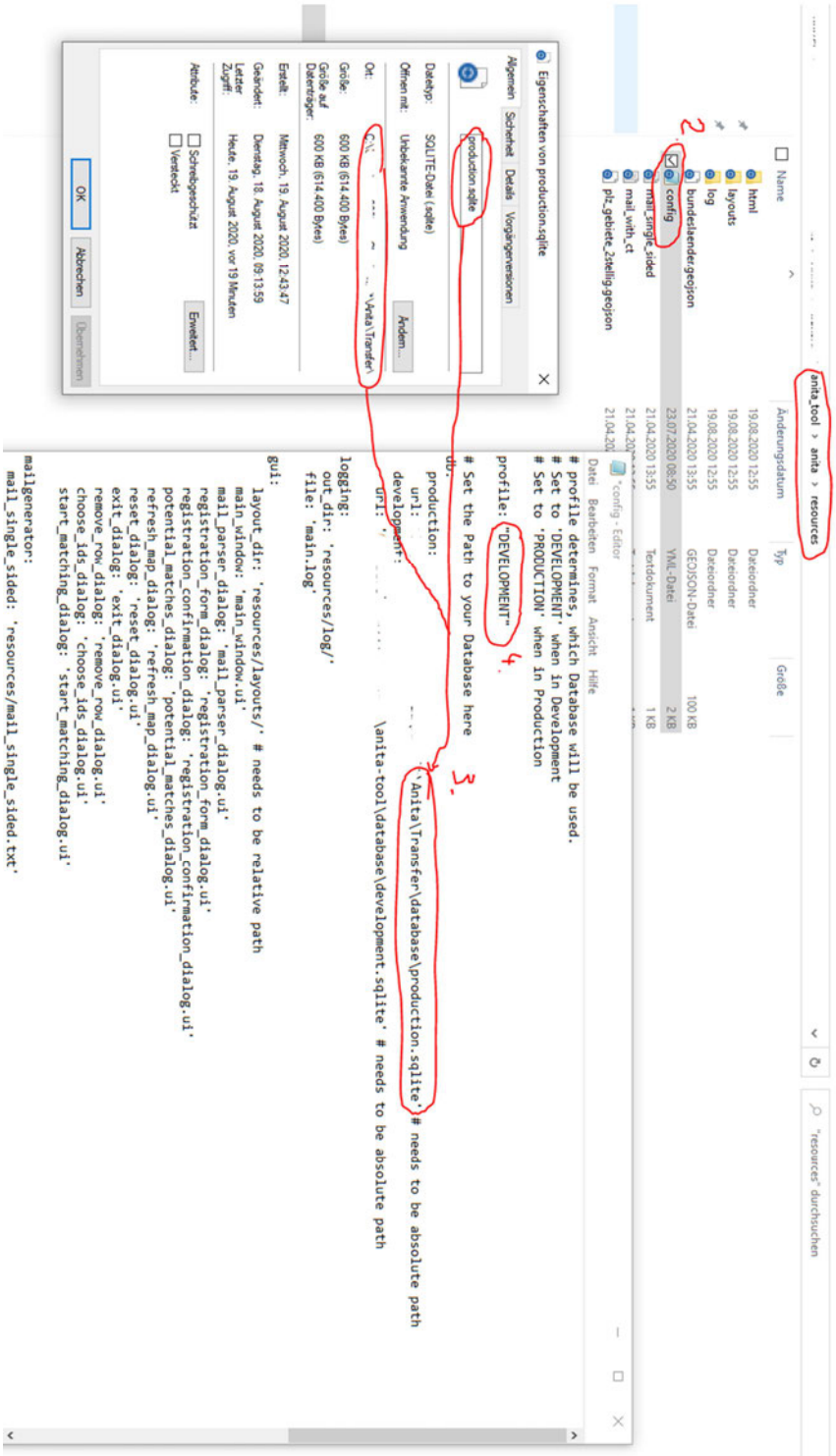


Abbildung 1. Anbinden der Datenbank

Oberfläche der Software

Aufbau (Abbildung 2)

Ganz oben befindet sich die Menü-Leiste (1.), über die verschiedenen Funktionen aufgerufen werden können, siehe Punkt Funktionen. Zentral befinden sich die Tabellen in denen sich die Daten der Care Giver und Care Taker sowie deren Assoziationen über das Verwandtschaftsverhältnis bzw. eines Matches, befinden (2.). Weiter Informationen hierzu sind unter dem Punkt Tabellen aufgeführt. Durch Rechtsklick auf einen Tabelleneintrag kann zudem ein Kontext Menü aufgerufen werden, welches das Hinzufügen, bzw. Entfernen von Einträgen erlaubt (3.).

Funktionen

- **Daten**: Speichern und Zurücksetzen von Änderungen in den Tabellen
  - Änderungen speichern: Speichern aller vorgenommenen Änderungen
  - Änderungen zurücksetzen: Zurücksetzen aller ungespeicherten Änderungen
- **Neuanmeldungen**:
  - Aus E-Mail auslesen: Öffnet einen Dialog, um die Daten aus dem Text einer Anmelde-Mail zu lesen
  - Manuell eingeben: Öffnet einen Dialog, um die Daten eines Care Givers und ggf. eines assoziierten Care Takers einzugeben
- **Matching**: Matching von Care Givern und Care Takern
  - Starten: Startet einen Dialog um das Matching zu starten
- **Map**: Aktualisierung und Anzeigen einer Karte mit allen Care Givern und Care Taker
  - Öffnen: Öffnet ein Fenster mit der Karte. Da teile der Karte aus dem Internet geladen werden, muss hierfür eine Internetverbindung bestehen. Außerdem kann es einen Moment dauern, bis die Karte dargestellt wird.
  - Aktualisieren: Erstellt die Karte anhand der aktuellen Datensätze

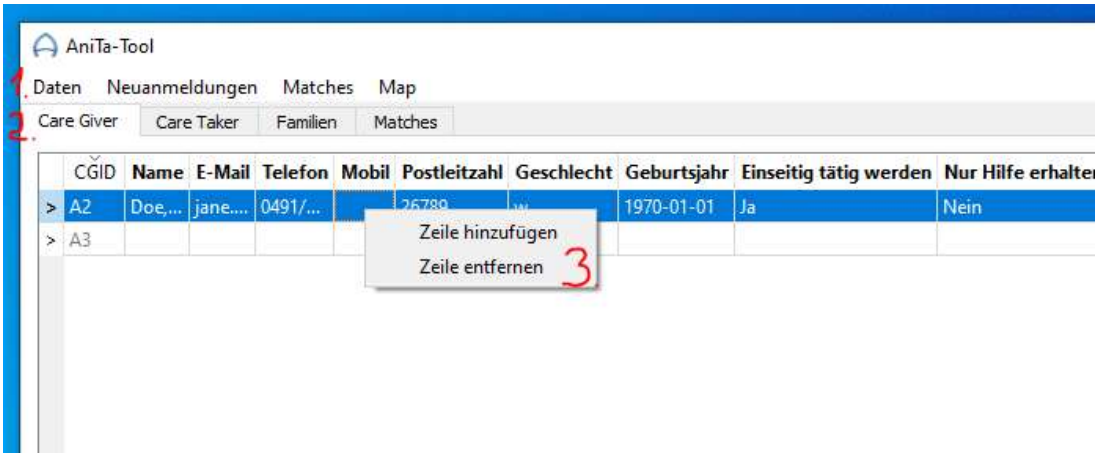


Abbildung 2. Bedienoberfläche des AniTa-Tools

Tabellen

Insgesamt können mit dem AniTa-Tool die Daten in den vier verschiedenen Tabellen, Care Giver, Care Taker, Familien und Matches verwaltet werden. Durch Doppelklick auf die Spalten lassen sich die Werte in diesen verändern. Die Bedeutung der einzelnen Tabellen und ihrer Spalten ist im Folgenden erläutert. Unterhalb der Tabellen Care Giver und Care Taker befinden sich die Felder Angehörige und Matches. Wählt man eine Tabellenzeile aus werden hier die Angehörigen, bzw. Matches des jeweiligen Care Givers, bzw. Care Takers angezeigt.

- **Care Giver**: Care Giver sind die erwachsenen Kinder, welche für eine nahestehende Person Hilfe suchen und ggf. Hilfe anbieten, bzw. nur Hilfe anbieten.
    - CGID: Der eindeutige Bezeichner eines Care Givers. Wird automatisch zugeteilt.
    - Name: Nachname und Vorname
    - E-Mail: E-Mail-Adresse
    - Telefon: Telefonnummer
    - Mobil: Mobiltelefonnummer
    - Postleitzahl: Die Postleitzahl des Wohnortes des Care Givers
    - Geschlecht: Das Geschlecht des Care Givers (m: männlich, w: weiblich, u: unbekannt)
    - Geburtsjahr: Geburtsjahr des Care Givers, als im Format JJJJ-MM-TT (Monat und Tag, standardmäßig als 1-1)
    - Einseitig tätig werden: Kennzeichnet Care Giver ohne assoziierten Care Taker
    - Nur Hilfe erhalten: Kennzeichnet Care Giver, die nicht bereit sind die Versorgung eines anderen Care Takers zu übernehmen
    - Aktivitätsradius: Der Radius in Kilometern um den Wohnort des Care Givers, in dem dieser bereit ist seine Hilfe anzubieten
    - Datenschutz akzeptiert: Kennzeichnet ob die Datenschutzbestimmung bei der Registrierung akzeptiert wurde.
    - Zusätzliche Informationen: Feld für sonstige Infos
- **Care Taker**: Care Taker sind die älteren Angehörigen der Care Giver, für welche diese eine Unterstützung suchen.
    - CTID: Der eindeutige Bezeichner eines Care Takers. Wird automatisch zugeteilt.
    - Postleitzahl: Die Postleitzahl des Wohnortes des Care Takers
    - Zusätzliche Informationen: Feld für sonstige Infos
    - Betreuung durch: Gibt an, ob die Betreuung durch einen Mann (m) oder eine Frau (w) gewünscht ist. Ist das Geschlecht unwichtig, kann die Option (u) gewählt werden.
- **Familien**: Diese Tabelle repräsentiert familiäre Beziehungen, bzw. ordnet Care Givern ihre Angehörigen (Care Taker) zu und umgekehrt.

    - CGID: Der eindeutige Bezeichner eines Care Givers.
    - CTID: Der eindeutige Bezeichner eines Care Takers.
    - Caretaker informiert: Kennzeichnet ob der Care Taker über die Anmeldung bei AniTa informiert ist
    - Beziehung: Die Beziehung zwischen dem Care Giver und Care Taker, z.B. Mutter oder Vater
    - Zusätzliche Informationen: Feld für sonstige Infos
- **Matches**: Diese Tabelle repräsentiert (potentielle) Betreuungsbeziehungen zwischen einem Care Giver und einem Care Taker
    - CGID: Der eindeutige Bezeichner eines Care Givers.
    - CTID: Der eindeutige Bezeichner eines Care Takers.
    - Status: Der aktuelle Status des Matches (awaiting_response: Warten auf Antwort von Angehörigen oder Tauschpartner, potential: Potentieller Match, noch keine weiteren Schritte eingeleitet, active: Aktiver Match zwischen Care Giver und Care Taker, denied: Match wurde durch eine der Parteien abgelehnt; cancelled: Match war aktiv, wurde aber abgebrochen)
    - Angehörige kontaktiert: Angehörige des Care Takers (CTID) wurden über möglichen Match informiert
    - Tauschpartner kontaktiert: Möglicher Tauschpartner (CGID) wurde über möglichen Match informiert
    - Zusätzliche Informationen: Feld für sonstige Infos

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original