HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät III
Medien, Information
und Design

Master-Thesis

E-PIX und FHIR HAPI MDM: Eine Gegenüberstellung der konzeptionellen und funktionalen Aspekte im Kontext des Record Linkage

Name: Aileen Stehn

Matrikelnummer: 1802102

Studiengang: Medizinisches Informationsmanagement (MMI)

Betreuer (Intern): Herr Prof. Dr. -Ing. Bott

Betreuer (Extern): Herr Dr. Bialke (Universitätsmedizin Greifswald)

Datum: 02.10.2025

Danksagung

Die vorliegende Arbeit entstand in Kooperation mit der Unabhängigen Treuhandstelle der Universitätsmedizin Greifswald, vertreten durch Herrn Dr. Bialke, sowie unter der Betreuung von Herrn Prof. Dr.-Ing. Bott an der Hochschule Hannover.

Zunächst möchte ich der Unabhängigen Treuhandstelle der Universitätsmedizin Greifswald meinen aufrichtigen Dank aussprechen, die mir die Möglichkeit geboten hat, meine empirische Arbeit in einem praxisnahen Unternehmensumfeld durchzuführen. Die gewährten Einblicke, Ressourcen und die kontinuierliche Unterstützung haben meine Arbeit in hohem Maße bereichert und mir wertvolle praktische Erfahrungen vermittelt. Darüber hinaus danke ich dem Krebsregister Mecklenburg-Vorpommern für die Bereitstellung der Testdaten.

Mein besonderer Dank gilt Herrn Dr. Bialke, der mich mit großem Engagement betreut hat. Er hat mir konstruktives Feedback gegeben und sich mit großer Sorgfalt und Geduld meiner Arbeit gewidmet. Sein stetiger Einsatz und seine Ermutigung, stets mein Bestes zu geben, haben den Fortschritt dieser Arbeit wesentlich gefördert.

Ein ebenso herzliches Dankeschön richte ich an Herrn Hampf, dessen kontinuierliche Unterstützung für mich von großem Wert war. Durch seine jederzeitige Hilfsbereitschaft und seine fachliche Expertise hat er die Erstellung dieser Arbeit in hohem Maße gefördert. Besonders hervorzuheben ist außerdem auch die Zeit, die er meiner Arbeit immer wieder mit großer Aufmerksamkeit gewidmet hat.

Ein herzliches Dankeschön gilt zudem Herrn Prof. Dr.-Ing. Bott, dessen stetiges Engagement mir während der gesamten Bearbeitungszeit wertvolle Orientierung geboten hat. Besonders seine Bereitschaft, meine regelmäßigen Zwischenstände aufmerksam zu begleiten und bei Rückfragen stets zeitnah mit konstruktiven Rückmeldungen zu reagieren, war für mich von großem Wert.

Abschließend möchte ich allen danken, die mich während der Erstellung dieser Arbeit unterstützt und mit hilfreichen Anregungen begleitet haben.

Abstract

Das Record Linkage bezeichnet die Identifikation und Zusammenführung von Datensätzen derselben Person. Dies ermöglicht die Verknüpfung von Forschungsergebnissen und die Beantwortung von Forschungsfragen, die mit isolierten Daten nicht möglich wäre. Das Record Linkage birgt jedoch das Risiko von Verknüpfungsfehlern, die zu Fehlinterpretationen von Forschungsergebnissen führen können. Um solche Fehler zu minimieren, ist daher eine Evaluation bestehender Record Linkage Lösungen notwendig. Ziel dieser Arbeit ist daher, die beiden Lösungen E-PIX und HAPI FHIR MDM hinsichtlich ihrer konzeptionellen Ansätze und ihrer Verknüpfungsqualität zu analysieren und gegenüberzustellen.

Zur Umsetzung wurden die Konzepte anhand ihrer Dokumentation analysiert und die Verknüpfungsqualität mit einem Echtdatensatz aus dem Krebsregister Mecklenburg-Vorpommern mit 2.729.204 Patientendatensätze gemessen. Beide Lösungen wurden in Testumgebungen mit Apache JMeter getestet, wobei verschiedene Matching-Szenarien konfiguriert und die Qualität mittels Precision, Recall und F1-Score bewertet wurden.

Die Ergebnisse zeigen, dass beide Lösungen die zentralen Komponenten eines Record Linkage Prozesses abdecken, sich jedoch in ihrem Konfigurationsumfang unterscheiden. Der E-PIX bietet größere Flexibilität in der Vorverarbeitung, den Blocking-Mechanismen, der Konfigruation von Matching-Variablen und im Umgang mit Multiple-Value-Feldern. HAPI FHIR MDM weist dagegen Vorteile bei der Auswahl von Matching-Algorithmen und der Klassifizierung von Possible Matches auf, zeigte jedoch zugleich deutliche Lücken in der Dokumentation. In den experimentellen Tests erreichte der E-PIX eine konstant hohe Verknüpfungsqualität von über 87% in allen Metriken, wobei die Multiple-Value-Feld-Funktion zu einer Qualitätssteigerung beitrug. Für HAPI FHIR MDM war eine gleichwertige Bewertung nicht möglich, da die Lösung bei der Verarbeitung des großen Datensatzes an ihre Grenzen stieß und lediglich 159 Fälle auswertete. Zwar lagen auch hier die Ergebnisse über 80%, jedoch sind diese aufgrund der geringen Datenbasis nicht mit den Resultaten des Enterprise Identifier Cross-Referencing (E-PIX) vergleichbar.

Für den E-PIX hat der konzeptionelle Vergleich gezeigt, dass es sinnvoll wäre, Possible Matches künftig auch über das FHIR-Gateway auszugeben, da diese Funktion bislang noch nicht verfügbar ist. Zudem deutet die Qualitätssteigerung durch die Multiple-Value-Feld-Funktion im E-PIX darauf hin, dass eine vergleichbare Funktion auch in HAPI FHIR MDM integriert werden sollte. Die Analysen verdeutlichen zugleich die aktuellen Grenzen von HAPI FHIR MDM sowie Defizite in der Dokumentation. Es erscheint daher sinnvoll, die Ergebnisse mit dem Hersteller ”Smile Digital Health” zu teilen, um Möglichkeiten zur Evaluation mit sehr großen Datenmengen zu entwickeln, als auch eine Verbesserung der Dokumentation zu ermöglichen.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

API Application Programming Interface

ASCII American Standard Code for Information Interchange

ASTNA Audit Trail and Node Authentication

BIH Berlin Institute of Health

BPPC Basic Patient Privacy Consents

CDA Clinical Document Architecture

CDR Clinical Data Repositories

CHI Community Health Index

DFG Deutsche Forschungsgemeinschaft

DICOM Digital Imaging and Communications in Medicine

DKMS Deutsche Knochenmarkspenderdatei

DMIMS Domain Message Information Models

DSGVO Datenschutz-Grundverordnung

DSUB Document Subscription

DZGs Deutsche Zentren für Gesundheit

DZHK Deutsche Zentrum für Herz-Kreislauf-Forschung

EHR Electronic Health Record

EID Enterprise Identifier

EM Erwartungsmaximierung

E-PIX Enterprise Identifier Cross-Referencing

EU Europäische Union

FHIR Fast Healthcare Interoperability Resources

HATEOAS Hypermedia as the Engine of Application State

HIMSS Management Systems Society

HL7 Health Level 7

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IDAT identifizierende Daten

ICD International Classification of Diseases

IHTSDO International Health Terminology Standards Development Organisation

IHE Integrating the Healthcare Enterprise

ISO International Organization for Standardization

JDK Java Development Kit

JPA Java Persistence API

JSON JavaScript Object Notation

KI Künstliche Intelligenz

LOINC Logical Observation Identifiers Names and Codes

MHD Mobile Health Documents

MDM Master Data Management

MII Medizininformatik-Initiative

MITA Medical Imaging & Technology Alliance

MPI Master Patient Index

MPQ Modality Performed Procedure Step Query

NHS National Health Service

NUM Netzwerk Universitätsmedizin

NYSIIS New York State Identification and Intelligence System

OAuth Open Authorization

OID Object Identifier

PACS Picture Archiving and Communication Systems

PDQ Patient Demographics Query

PIX Patient Identifier Cross-reference Service

PPRL Privacy-Preserving Record Linkage

RDF Resource Description Framework

RFH Resources for Health

RIM Reference Information Model

RMIM Refined Message Information Model

RSNA Radiological Society of North America

SGML Standard Generalized Markup Language

SNM Sorted Neighborhood Method

SOAP Simple Object Access Protocol

SNOMED Systematized Nomenclature of Medicine

SQL Structured Query Language

THS Treuhandstelle

TTP Trusted Third Party

UUID Universally Unique Identifier

WHO World Health Organization

XCA Cross-Community Access

XDM Cross-Enterprise Document Media

XDR Cross-Enterprise Document Reliable Interchange

XDS Cross-Enterprise Document Sharing

XML Extensible Markup Language

1. Einleitung

1.1. Gegenstand und Motivation

Das Record Linkage bezieht sich auf den Prozess der Identifikation von Datensätzen, die zu derselben Identität gehören. Die Datensätze können sich entweder innerhalb derselben Quelle oder über verschiedene Quellen hinweg verteilt befinden [7, S.1]. Der Prozess des Record Linkage spielt eine entscheidende Rolle in der wissenschaftlichen Forschung insbesondere im Bereich der Gesundheitsforschung. Hier ermöglicht die Identifikation und Verknüpfung von Datensätzen aus verschiedenen Quellen die Generierung wertvoller Informationen und die Beantwortung komplexer Forschungsfragen, die mit Daten aus nur einer Quelle nicht beantwortet werden könnten [8, S.1]. Durch die Verknüpfung mehrerer Datenquellen können demnach wertvolle Informationen entstehen, die beispielsweise zu einer Verbesserung von Therapieansätzen beitragen oder die Identifizierung von Fall- und Kontrollgruppen begünstigen können. Daher kann mit Hilfe des Record Linkage die Versorgungsforschung unterstützt und die Qualität der medizinischen Versorgung verbessert werden [7, 8, S.1,1].

Innerhalb des Record Linkage stellen Verknüpfungsfehler, wie sogenannte Homonym- und Synonymfehler, ein bedeutendes Problem dar. Synonymfehler treten auf, wenn Datensätze derselben Person nicht korrekt erkannt und daher nicht zusammengeführt werden. Ein Homonymfehler entsteht hingegen, wenn Datensätze unterschiedlicher Personen fälschlich zusammengeführt werden. Eine hohe Anzahl solcher Fehler kann schwerwiegende Folgen für die Forschung haben, darunter Selektionsverzerrung, Informationsverzerrung, Fehlklassifizierungen oder Messfehler [9, 10, S.2051,654].

Inzwischen wurden verschiedene Record Linkage Lösungen entwickelt, die eine effektive Verknüpfung von Datensätzen ermöglichen [11, S.25,27]. Eine solche Lösung ist der E-PIX, der durch die Universitätsmedizin Greifswald im Rahmen des GANI-MED-Projekts entwickelt und 2014 erstmals unter Open-Source-Lizenz veröffentlicht wurde [12]. Der E-PIX zeichnet sich unter anderem durch die Unterstützung der Integrating the Healthcare Enterprise (IHE) Profile PIX & PDQ aus und verfügt über eine SOAP sowie eine Health Level 7 (HL7)-Fast Healthcare Interoperability Resources (FHIR)-Schnittstelle [2, S.7,9]. Diese Record Linkage Lösung realisiert ein Identitätsmanagement, das die Erkennung von Duplikaten in Personendaten ermöglicht [13, S.2]. Insbesondere im Rahmen nationaler Forschungsprojekte hat sich der E-PIX als etabliertes Instrument zur Verknüpfung von Patientendaten bewährt. Der E-PIX wird unter anderem von ausgewählten Krebsregistern der Bundesländer, bestimmten Krankenkassen, den Deutschen Zentren für Gesundheit (DZGs), wie beispielsweise dem Deutschen Zentrum für Herz-Kreislauf-Forschung (DZHK), einem Projekt der Deutschen Knochenmarkspenderda-

tei (DKMS), dem Berlin Institute of Health (BIH) der Charité - Universitätsmedizin Berlin, der größten Langzeistudie Deutschlands „NAKO Gesundheitsstudie“ sowie von einer Vielzahl der 36 Universitätsstandorte des Netzwerk Universitätsmedizin (NUM) und der Medizininformatik-Initiative (MII) genutzt [5].

Neben dem E-PIX existiert auch eine Record Linkage Lösung, die auf den Prinzipien des von HL7 entwickelten Standards FHIR basiert und ebenfalls öffentlich zugänglich ist. FHIR wurde entwickelt, um den Austausch und die Interoperabilität von Gesundheitsdaten zu fördern. Dieser Standard umfasst unter anderem die Definition standardisierter Datenstrukturen (Ressourcen) sowie eine Schnittstelle für den Datenaustausch [4]. Länder wie die USA, Australien, Neuseeland, Großbritannien sowie die Mitgliedsstaaten der G7, die Europäische Union (EU) und zahlreiche asiatische Nationen haben FHIR als Standard anerkannt, um die Interoperabilität im Gesundheitswesen zu fördern und einen nahtlosen Austausch von Gesundheitsdaten zu ermöglichen [14–18].

HAPI FHIR ist eine Java-Implementierung des FHIR-Standards, die von dem Gesundheitstechnologieunternehmen Smile Digital Health entwickelt wurde. Das Ziel des Unternehmens ist es, Barrieren zwischen Information und Versorgung zu reduzieren. Hierzu wird die Leistungsfähigkeit offener Standards genutzt, um IT-Lösungen für das Gesundheitswesen zu entwickeln [14]. Eine spezifische Komponente ist das HAPI FHIR Master Data Management (MDM) Modul, eine Record Linkage Lösung, die Funktionen zur automatischen Identifizierung und Verknüpfung von doppelten Patientendaten bietet. Dadurch können Verknüpfungen zwischen FHIR-Ressourcen erstellt und verwaltet werden, sodass ersichtlich wird, ob verschiedene Datensätze auf dieselbe (reale) Ressource verweisen [3].

Im Bereich der Medizin wächst der Bedarf an Informationen aus verschiedenen Institutionen, wie z.B. Leistungsdaten, klinischen Informationen von Gesundheitseinrichtungen, nationalen Sterbedaten und Registereinträgen. Aus diesem Grund sind Methoden zur Datenverknüpfung und die Sicherstellung der Verknüpfungsqualität wesentliche Forschungsgebiete, um eine verlässliche Nutzung verknüpfter Daten zu gewährleisten [19, S.8]. Demnach ist es notwendig, bestehende Record Linkage Lösungen kontinuierlich zu evaluieren und an die Ansprüche neuer Herausforderungen, wie beispielsweise dem standortübergreifenden Record Linkage, anzupassen, um die Anzahl von Verknüpfungsfehlern zu minimieren und negative Auswirkungen auf die Validität von Forschungsergebnissen zu vermeiden. Angesichts der weit verbreiteten Anwendung des FHIR-Standards und der etablierten Rolle des E-PIX in der Forschung sowie dessen Bereitstellung von FHIR-basierten Funktionen zur Daten- und Identitätsverwaltung [6], erscheint ein Vergleich der beiden Lösungen sinnvoll. Durch die Implementierung gemeinsamer Algorithmen in beiden Lösungen lassen sich standardisierte Testumgebungen schaffen, die eine vergleichbare Evaluierung ermöglichen. Gleichzeitig bestehen konzeptionelle Unterschiede zwischen den Lösungen, die auf diese Weise hinsichtlich ihrer Wirksamkeit und Fehlerresistenz überprüft werden können.

1.2. Problemstellung

Verknüpfungsfehler können gravierende Auswirkungen auf die Validität von Forschungsergebnissen haben. Die Relevanz dieses Problems zeigt sich in einem Projekt der Deutsche Forschungsgemeinschaft (DFG): „Evaluierung eines indirekten Linkage-Ansatzes anhand einer Beispielstudie zum Risiko einer Krebsneuerkrankung und der Krebsmortalität bei Patienten mit Typ-2-Diabetes unter Behandlung mit verschiedenen Antidiabetika“. In dieser Beispielstudie wurden Daten aus einem Krebsregister verwendet, um Krankenkassendaten der pharmakoepidemiologischen Forschungsbank mit Informationen zum Tumorstadium und zur Todesursache zu ergänzen. Innerhalb dieses Projekts haben hohe Fehlerquoten zu einer Unterschätzung des Krebsrisikos geführt [8, S.31,89]. Dieses Beispiel verdeutlicht, dass eine hohe Anzahl an Verknüpfungsfehlern nicht nur die wissenschaftliche Aussagekraft von Studien beeinträchtigt, sondern im schlimmsten Fall auch gesundheitliche Konsequenzen für die Bevölkerung haben kann.

Die Häufigkeit solcher Fehler kann je nach eingesetzter Record Linkage Lösung und Konfigurationsmöglichkeiten variieren. Um die Qualität der Datenverknüpfung zu verbessern und mögliche Synergien zwischen verschiedenen Record Linkage Lösungen gezielt zu nutzen, ist eine systematische Evaluierung und Gegenüberstellung bestehender Ansätze erforderlich. Soweit ersichtlich, existiert bislang keine Analyse, die die konzeptionellen und funktionalen Unterschiede zwischen E-PIX und HAPI FHIR MDM untersucht. Aus den beschriebenen Sachverhalten ergeben sich folgende Problemstellungen:

Problem 1: Es gibt bisher keine Analyse der konzeptionellen und funktionalen Unterschiede zwischen den Record Linkage Lösungen E-PIX und HAPI FHIR MDM.

Problem 2: Es ist ungewiss, in welchem Maß sich die Häufigkeit von Homonym- und Synonymfehlern zwischen E-PIX und HAPI FHIR MDM unterscheidet und wie sich diese auf die Datenqualität auswirken.

1.3. Ziele

Das Ziel dieser Masterarbeit ist die Analyse sowie der Vergleich der beiden Record Linkage Lösungen E-PIX und HAPI FHIR MDM hinsichtlich ihres Konzepts, Flexibilität und Verknüpfungsqualität. Im Einzelnen wird das Erreichen folgender Ziele angestrebt:

Ziel 1: Eine Gegenüberstellung der funktionalen und konzeptionellen Aspekte zwischen E-PIX und HAPI FHIR MDM.

Ziel 2: Eine Analyse der Verknüpfungsqualität von E-PIX und HAPI FHIR MDM, insbesondere im Hinblick auf die Häufigkeit und Ursachen von Homonym- und Synonymfehlern.

1.4. Fragestellung

Folgende Fragen lassen sich von der Zielsetzung ableiten und sollen in dieser Arbeit beantwortet werden:

Zu Ziel 1:

- Frage 1: Welche konzeptionellen Unterschiede bestehen zwischen E-PIX und HAPI FHIR MDM?
- Frage 2: Inwiefern beeinflussen diese Unterschiede die Anwendbarkeit und Flexibilität beider Lösungen?

Zu Ziel 2:

- Frage 3: Wie unterscheidet sich die Verknüpfungsqualität zwischen E-PIX und HAPI FHIR MDM?
- Frage 4: Welche Faktoren führen zu Abweichungen in den Matching-Entscheidungen beider Lösungen?

1.5. Gliederung

Diese Arbeit gliedert sich in insgesamt sechs Kapitel. Zu Beginn der Arbeit dient das Kapitel „Grundlagen“ dazu, einen Rahmen für die nachfolgende Untersuchung zu schaffen. Zunächst werden die Grundlagen des Record Linkage erläutert, wobei verschiedene Verfahren, Algorithmen sowie die damit verbundenen Herausforderungen thematisiert werden. Anschließend wird der Interoperabilitätsstandard HL7 FHIR vorgestellt und in den Kontext bestehender Standards eingeordnet. Dieser Abschnitt soll ein Verständnis für die Funktionsweise von HAPI FHIR MDM und den FHIR-Schnittstellen der Record Linkage Lösungen vermitteln.

Das dritte Kapitel widmet sich der Vorstellung der beiden Record Linkage Lösungen. Dabei wird das jeweilige Konzept vorgestellt, indem Konfigurationsmöglichkeiten näher erläutert werden.

Im vierten Kapitel wird die Methodik der Arbeit beschrieben. Diese umfasst sowohl die durchgeführte Literaturrecherche als auch die experimentelle Analyse. In Bezug auf die Literaturrecherche wird erläutert, welche Datenbanken und Suchstrategien genutzt wurden und in welchem Zeitraum die Recherche stattfand. Die Methodik der experimentellen Analyse wird anhand des Testkonzepts zur Bewertung der Verknüpfungsqualität vorgestellt. Hierzu werden der verwendete Referenzdatensatz sowie die entwickelten Konfigurationen detailliert beschrieben. Ein weiteres Unterkapitel widmet sich der Evaluierung der Verknüpfungsergebnisse, wobei Metriken wie Precision, Recall und der F1-Score erläutert werden.

Das fünfte Kapitel präsentiert die Ergebnisse der Untersuchung. Als erstes erfolgt die Gegenüberstellung der konzeptionellen Unterschiede zwischen den beiden Record Linkage Lösungen, indem Gemeinsamkeiten und Unterschiede herausgearbeitet werden. Anschließend werden die Ergebnisse der experimentellen Analyse für jede Record Linkage Lösung dargestellt.

Im folgenden Fazit werden abschließend die zentralen Erkenntnisse dieser Arbeit zusammengefasst und die eingangs formulierten Forschungsfragen beantwortet. Zudem werden die Ergebnisse kritisch hinterfragt, mögliche Limitationen der Untersuchung aufgezeigt und Implikationen für zukünftige Forschung diskutiert.

2. Grundlagen

2.1. Grundlagen des Record Linkage

In diesem Kapitel werden die grundlegenden Konzepte und Verfahren des Record Linkage vorgestellt, um ein Verständnis für die nachfolgende empirische Arbeit zu schaffen.

2.1.1. Definition und Anwendung

Das Record Linkage bezieht sich auf den Prozess der Identifikation von Datensätzen, die zu derselben Identität gehören [7, 20, S.1,5]. Dadurch kann eine Verknüpfung von Daten aus verschiedenen Quellen ermöglicht werden, die sich auf dieselbe Person beziehen und zu einer neuen erweiterten Datenressource kombiniert werden [21, S.1]. In der Regel erfolgt diese Verknüpfung von Daten aus mindestens zwei unterschiedlichen Quellen [7, 10, S.648,1].

Biobanken, Forschungs- und Gesundheitseinrichtungen erfassen und speichern personenbezogene Daten aufgrund verschiedener Zwecke, wie beispielsweise zur Gesundheitsüberwachung oder für die Identifizierung von Nebenwirkungen. Dabei können in den unterschiedlichen Einrichtungen Informationen zu derselben Person enthalten sein. Durch die Zusammenführung dieser Datensätze aus den verschiedenen Datenbanken können umfassende Erkenntnisse über einzelne Personen gewonnen werden, die unter anderem zur Optimierung von Therapieansätzen, zur Identifikation von Fall- und Kontrollgruppen sowie zur Verfeinerung von Messwerten beitragen können, was wiederum die Versorgungsforschung unterstützt und die Qualität der medizinischen Versorgung verbessern kann [7, 8, 22, S.8,1,1].

Insbesondere die Verknüpfung von Biobanken, Registern und Gesundheitsdatenbanken mit externen Datenquellen bietet ein erhebliches Potenzial zur Erweiterung der Forschungs- und Analysemöglichkeiten [22, S.21]. So kann die Verknüpfung von Gesundheitsdaten mit Nicht-Gesundheitsdaten wichtige Kontextinformationen für Forscher liefern, die medizinische Fragestellungen untersuchen, deren Ursachen oder Folgen über das Gesundheitssystem selbst hinausgehen [20, S.6]. Beispielsweise könnte untersucht werden, ob der Familienstand Auswirkungen auf die Lebenserwartung hat, indem zivile Geburts-, Todes- und Heiratsurkunden verknüpft werden [23, S.1].

Durch das Verfahren des Record Linkage können Forscher auf bestehende Datenquellen zurückgreifen, ohne den erheblichen Zeit- und Kostenaufwand einer neuen Datenerhebung sowie die zusätzliche Belastung der Teilnehmer in Kauf nehmen zu müssen. Dadurch lassen sich Kohorten in einem größeren und detaillierteren Maßstab erstellen, was nicht nur kostengünstiger ist, sondern auch die Belastung für die Teilnehmer verringert [19, 21, 24, 25, S.1699,

1, 224, 2]. Das Verfahren des Record Linkage kann daher auch bei Fragestellungen von Nutzen sein, die entweder eine große Stichprobengröße erfordern (z.B. bei seltenen Krankheiten), eine umfassende Bevölkerungsabdeckung benötigen (z.B. zur Planung von Pandemie-Reaktionen) oder sich auf schwer erreichbare Personengruppen beziehen [21, S.1]. Ein weiterer Grund für die Verknüpfung von Datenquellen besteht darin, zusätzliche relevante Variablen zu erschließen, die erst durch eine Datenzusammenführung entstehen [8, 25, S.220,5].

In der Gesundheitsforschung hat sich das Record Linkage bereits in zahlreichen Projekten als wertvoll erwiesen. Ein Beispiel hierfür ist die Verknüpfung des niederländischen Krebsregisters mit einer Pathologiedatenbank, um das Risiko und die Prognose von Endometriumkrebs nach Tamoxifenbehandlung zu untersuchen [26]. Ein weiteres Beispiel ist die Erforschung des Einflusses von Thiazid-Diuretika und genetischen Variationen im Renin-Angiotensin-System auf das Diabetes-Risiko durch die Verknüpfung von Apothekenaufzeichnungen und Biobankdaten [27]. Zudem ergab die Verknüpfung von Diabetes- und Sterberegistern in Wales eine Untererfassung von Diabetes als Todesursache und zeigte eine um durchschnittlich sieben Jahre verkürzte Lebenserwartung für Betroffene, vor allem durch Herz-Kreislauf-Erkrankungen [28].

Darüber hinaus kann der Abgleich von Datensätzen dazu eingesetzt werden, Datenquellen auf potenzielle Duplikate zu prüfen und somit zu gewährleisten, dass eine Identität nicht mehrfach in separaten Datensätzen repräsentiert wird [10, S.648]. Denn es ist nicht nur bei der Zusammenführung von Datensätzen möglich, dass eine Person mehrfach erfasst wird. Auch in Registern, die darauf abzielen, Personen oder Ereignisse nur einmal zu dokumentieren, wie Geburten, Sterbefälle oder meldepflichtige Krankheiten, kann je nach System eine unterschiedlich hohe Anzahl an Duplikaten auftreten [25, S.220]. Wird das Record Linkage ausschließlich zur Daten-Deduplizierung eingesetzt, fördert es eine effiziente und sichere Handhabung der Informationen und unterstützt die Genauigkeit sowie die Integrität der Daten [1, S.10].

2.1.2. Datenaufbereitung

Auswahl der Matching-Variablen

Bevor eine Prüfung auf Duplikate oder eine Verknüpfung von Datenquellen erfolgen kann, müssen geeignete Matching-Variablen, auch Verknüpfungsvariablen genannt, festgelegt werden (siehe Abbildung 2.1). Diese Variablen dienen als gemeinsame Merkmale, anhand derer die Datensätze abgeglichen werden. Dabei werden Variablen verwendet, die in allen zu verknüpfenden Datenquellen vorhanden sind [22, S.4].

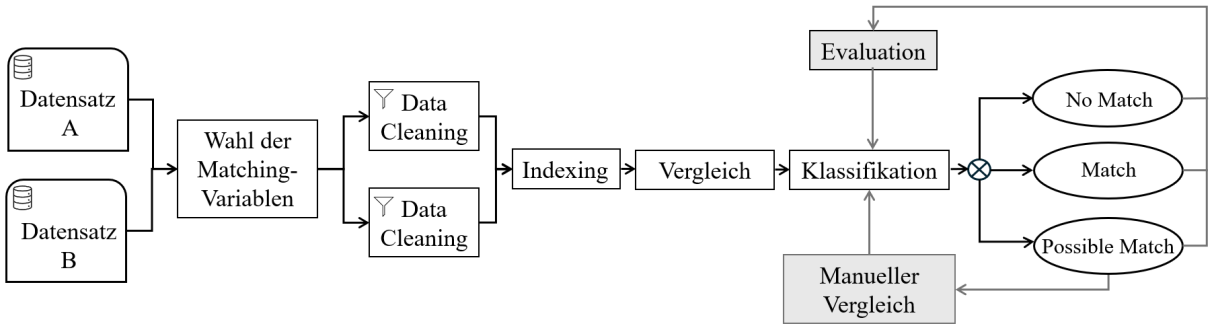


Abbildung 2.1.: Darstellung des Matching-Prozesses (entnommen aus: eigene Aufnahmen)

In einigen Ländern, wie den skandinavischen, existieren eindeutige persönliche Identifikationsnummern, die als unmissverständliche Identifikatoren fungieren und somit den Vergleich sowie die Verknüpfung von Identitäten erheblich vereinfachen [29, S.5]. Beispielsweise wird in England die National Health Service (NHS)-Nummer als zentrale Kennung für Gesundheitsdienste verwendet, während in Schottland der Community Health Index (CHI) als primäre Gesundheitsidentifikation dient. In Kanada, speziell in den Provinzen Ontario und Manitoba, kommen verschlüsselte Gesundheitskartennummern zum Einsatz, um die Verknüpfung von Gesundheitsdaten zu ermöglichen [30]. Solche nationalen Identifikationsnummern sind jedoch oft nicht nutzbar, da sie entweder bewusst nicht in Datensätzen enthalten sind oder aufgrund rechtlicher Vorgaben nicht verwendet werden dürfen [22, S.21]. Potenzielle eindeutige Identifikatoren wie die Sozialversichertennummer oder die Steueridentifikationsnummer könnten in Deutschland zwar als eindeutige Identifikatoren dienen, dürfen jedoch nur für spezifische, eng begrenzte Zwecke verwendet werden und sind daher nicht in allen Datenquellen verfügbar, was ihre Anwendbarkeit einschränkt [8, S.7,13].

Aus diesem Grund werden stattdessen sogenannte Quasi-Identifikatoren, wie Name, Nachname, Postleitzahl, Geburtsdatum, Geschlecht und Nationalität als Matching-Variablen herangezogen, um Identitäten miteinander zu vergleichen [31, S.4], [8, S.6,7], [1, S.9]. Das Verfahren des Record Linkage wird folglich häufig unter Verwendung von personenidentifizierenden Daten (IDAT) durchgeführt, die in ihren Kombinationen eine praktikable Alternative darstellen, um eine hinreichende Differenzierung zwischen Personen zu ermöglichen [22, S.5,8], [25, S.218], [21, S.1], [31, S.4,5]. Die Stärke dieser Kombinationen hängt maßgeblich von der Anzahl der verwendeten Identifikatoren sowie deren individueller Unterscheidungskraft ab [22, S.6]. Zwar könnte die Verknüpfungsgenauigkeit theoretisch durch die Berücksichtigung möglichst vieler Identifikatoren verbessert werden, jedoch sind diese in der Praxis häufig fehlerbehaftet oder verändern sich im Laufe der Zeit, was wiederum die Qualität der Datenverknüpfung negativ beeinträchtigen kann [22, S.6]. Matching-Variablen lassen sich in drei verschiedene Kategorien einteilen. String-Variablen umfassen beispielsweise Daten wie Namen, Adressen, Postleitzahlen oder die textuelle Darstellung eines Datums. Numerische Variablen beinhalten Größen wie Alter oder Messwerte, während kategorische Variablen Merkmale wie Geschlecht, ethnische Zugehörigkeit, Bildungsstand oder Familienstatus abbilden [22, S.9].

Bei der Auswahl der Variablen ist zu beachten, dass nicht alle Variablen die gleiche Informationsdichte aufweisen. Einige Identifikatoren, wie etwa die Adresse, besitzen eine geringere Aussagekraft, da die Möglichkeit besteht, dass die Person ihren Wohnort gewechselt hat [22, S.8,16], [31, S.3]. Der Name ist eine zentrale Matching-Variable für die Beurteilung von Duplikaten. Eine vollständige Übereinstimmung des Namens allein reicht jedoch nicht aus, um zweifelsfrei festzustellen, dass es sich um dieselbe Person handelt, da viele Menschen denselben Namen tragen, wie beispielsweise John Smith, Muhammad Hussein, Wong oder Hon Nguyen. Der Nachname ist außerdem sehr fehleranfällig, was seine Verwendung ebenfalls einschränkt [32, S.9], [22, S.6]. Personennamen sind in der Regel zeitlich stabil, wobei diese Stabilität nicht uneingeschränkt gilt, da viele Länder unter bestimmten Bedingungen eine Namensänderung ermöglichen [1, S.28]. Im Gegensatz dazu bietet das Geburtsdatum den Vorteil, dass es eine konstante Variable darstellt, die sich im Laufe der Zeit nicht verändert. Das Geschlecht ist hingegen eine weniger aussagekräftige Variable, da es nur begrenzte Unterscheidungskraft besitzt [1, S.27,28]. Aufgrund solcher Unterschiede ist es entscheidend, bei der Prüfung auf Duplikate mehrere Matching-Variablen sorgfältig auszuwählen, gezielt zu kombinieren und sinnvoll zu berücksichtigen [1, S.9].

Datenbereinigung

Da die Daten in unterschiedlichen Formaten und Qualitäten vorliegen können, ist nach der Wahl der Matching-Variablen eine sorgfältige Vorverarbeitung notwendig (vgl. Abbildung 2.1). Eine gründliche Bereinigung und Standardisierung der Daten erhöht die Genauigkeit sowie die Verlässlichkeit ihrer Nutzung und kann demnach zu einer hohen Verknüpfungsqualität führen. In diesem Prozess wird die Struktur der Daten, die Art ihrer Speicherung sowie ihre Vollständigkeit überprüft, wobei mögliche Unregelmäßigkeiten identifiziert und behoben werden können [7, S.30]. Hierzu können verschiedene Datenbereinigungsverfahren zum Einsatz kommen, darunter die Angleichung der Groß- und Kleinschreibung, die Entfernung von Satzzeichen sowie die Bereinigung überflüssiger Leerzeichen, einschließlich aufeinanderfolgender, führender und nachfolgender Leerzeichen. Zudem können Präfixe (z. B. Herr, Frau, Prof., Dr.) und Suffixe (z. B. Senior, Junior) eliminiert, Wortvertauschungen erkannt (z. B. „A Sayers“ statt „Sayers A“) und Spitznamen identifiziert werden. Darüber hinaus umfasst die Standardisierung auch Prozeduren wie die Vereinheitlichung des Datumsformats sowie die Identifikation fehlerhafter Datumsangaben, die zeitlich nicht plausibel erscheinen. Ebenso können Identifikatoren in strukturelle Bestandteile zerlegt werden, beispielsweise durch die Trennung einer Adresse in Bundesland und Postleitzahl [33, S.958], [7, S.31].

Blocking/Indexing

Bei großen Datensätzen ist es ratsam nach der Datenbereinigung den Vergleichsraum auf diejenigen übereinstimmenden Paare zu reduzieren, die bestimmte grundlegende Kriterien erfüllen

(siehe Abbildung 2.1) [7, S.33,34]. Bevor die eigentliche Verknüpfung der Datenquellen erfolgt, kann als letzter Schritt der Vorverarbeitung das sogenannte Blocking eingesetzt werden, auch bekannt als Filtering oder Indexing. Dieses Verfahren dient dazu, die Effizienz des Verknüpfungsprozesses zu steigern, indem nicht sämtliche Datensätze aus Quelle A mit allen Datensätzen aus Quelle B abgeglichen werden. Stattdessen erfolgt der Abgleich nur zwischen Datensätzen, die beispielsweise bei mindestens einer zuvor festgelegten Variable übereinstimmen (exaktes Blocking) oder ein zuvor festgelegtes Maß an Ähnlichkeit aufweisen. Diese Variablen, die als Blocking-Variablen bezeichnet werden, müssen im Vorfeld sorgfältig ausgewählt werden, um eine sinnvolle Reduzierung der Vergleichspaare zu gewährleisten [11, S.24,29]. Die Anzahl der möglichen Vergleichspaare entspricht dem Produkt der Datensatzanzahl beider Quellen. Enthalten die beiden Datenquellen beispielsweise jeweils eine Million Datensätze, ergibt sich eine Anzahl von einer Billion zu vergleichender Paare. Ein vollständiger Abgleich aller Datensätze wäre in einem solchen Fall ohne leistungsstarke Software und Hochleistungsrechner kaum umsetzbar, da dies zu erheblichen Kapazitätsproblemen führen würde [30, S.3], [29, S.4,5]. In diesem Fall könnte beispielsweise ein exaktes Blocking mit der Postleitzahl als Variable genutzt werden. Durch die Gruppierung der Datensätze anhand dieses Merkmals werden anschließend nur diejenigen Einträge miteinander verglichen, die dieselbe Postleitzahl aufweisen. Auf diese Weise würde sich die Gesamtzahl der erforderlichen Vergleiche erheblich verringern, was die Effizienz dieses Verknüpfungsprozesses steigern würde [8, S.15,16].

Datensatzpaare, die die zuvor festgelegten Kriterien der Blockierungsphase nicht erfüllen, werden automatisch als Nicht-Übereinstimmungen betrachtet und aus dem Abgleich ausgeschlossen [7, S.33,34]. Demnach besteht das Hauptziel dieses Verfahrens darin, jene Paare zu eliminieren, die mit hoher Wahrscheinlichkeit keine sinnvolle Verknüpfung darstellen [34]. Die Wahl einer geeigneten Blockierungsstrategie erfordert jedoch eine sorgfältige Abwägung, da diese maßgeblich die Qualität der Verknüpfung beeinflussen kann. Insbesondere unvollständige oder fehlerhafte Daten können dazu führen, dass potenziell zusammengehörige Datensätze nicht erkannt werden [7, S.33,34], [29, S.4,5]. Auch wenn idealerweise fehlerfreie Blocking-Variablen verwendet werden sollten, ist dies in der Praxis selten vollständig umsetzbar. Um dennoch eine hohe Erfassungsrate relevanter Datensätze zu gewährleisten, wird häufig ein mehrstufiges Blocking-Verfahren eingesetzt. In mehreren Durchgängen werden unterschiedliche Blocking-Variablen eingesetzt, um zuvor übersehene potenzielle Übereinstimmungen zu erkennen [22, S.27], [35].

Eine bekannte Blocking-Methoden ist unter anderem die Sorted Neighborhood Method (SNM) [22, S.14]. Die Methode der sortierten Nachbarschaft lässt sich in drei wesentliche Schritte unterteilen. Zunächst wird für jeden Datensatz ein Schlüssel gebildet, der aus der Extraktion relevanter Matching-Variablen oder ihrer Teilmengen besteht. Ein Beispiel hierfür könnte sein, dass die ersten drei Konsonanten des Nachnamens mit den ersten drei Buchstaben des Vornamens kombiniert werden, gefolgt von der Adressnummer und den Konsonanten des Straßennamens.

Die Wahl des Schlüssels spielt eine entscheidende Rolle für die Effektivität der Methode, da der Schlüssel als Sequenz von Attributen oder Teilzeichenfolgen innerhalb dieser Attribute definiert wird, die zur Sortierung der Datensätze verwendet werden. Im zweiten Schritt erfolgt die Sortierung der gesamten Datensatzliste anhand der zuvor erstellten Schlüssel. Das Ziel dieses Vorgangs ist es, gleichwertige oder übereinstimmende Datensätze so anzuordnen, dass sie in der finalen, sortierten Liste nahe beieinander liegen. Der abschließende Schritt beinhaltet das Durchlaufen der Liste mit einem vordefinierten Fenster von festgelegter Größe. Wenn die Fenstergröße w beträgt, wird jeder neue Datensatz, der in das Fenster eintritt, mit den $w-1$ vorhergehenden Datensätzen verglichen, um potenzielle Übereinstimmungen zu identifizieren. Sobald der erste Datensatz im Fenster verarbeitet wurde, rutscht er aus dem Fenster heraus und der Prozess wiederholt sich für die verbleibenden Datensätze [36].

Neben der SNM existieren weitere Blocking-Methoden, die unterschiedliche Ansätze zur effizienten Gruppierung von Datensätzen nutzen. Dazu gehört das Token basierte Blocking, bei dem Datensätze anhand gemeinsamer Wörter zusammengeführt werden, die q-Gram-Indexierung, bei der Zeichenketten in Abschnitte fester Länge unterteilt werden und die Suffix-Array-Indexierung, die auf Endungen von Zeichenketten basiert. Zudem werden Verfahren wie Canopy Clustering, Locality-Sensitive Hashing (LSH) sowie Meta-Blocking und das Progressive-Blocking zur Optimierung der Blockstruktur eingesetzt [37, S.88-90].

2.1.3. Record Linkage Verfahren

Im Rahmen der Datensatzverknüpfung lassen sich das deterministische regelbasierte Record Linkage und das probabilistische, auf Wahrscheinlichkeiten beruhende, Record Linkage unterscheiden. Beide Methoden weisen in der praktischen Anwendung Überschneidungen auf und werden nach der Vorverarbeitung für den Vergleich der Daten verwendet (vgl. Abbildung 2.1) [21, S.1], [33, S.955], [7, S.31], [20, S.7]. Insbesondere bei großen und heterogenen Datensätzen kommt häufig eine hybride Strategie zum Einsatz, die Elemente beider Verfahren kombiniert, um die Verknüpfungsgenauigkeit zu optimieren [22, S.8]. Das übergeordnete Ziel jeder Verknüpfungsmethode besteht darin, Paare von Datensätzen auf Basis ihres tatsächlichen Übereinstimmungsstatus korrekt zu klassifizieren [9, S.2050]. Die zugrundeliegende Annahme dieser Verfahren besagt, dass Datensätze mit hoher Übereinstimmung in ihren gemeinsamen Variablen mit großer Wahrscheinlichkeit der gleichen Identität zuzuordnen sind [34].

Deterministisches Record Linkage

Deterministische Methoden zur Datenverknüpfung basieren auf vordefinierten Regeln und stellen den einfachsten Ansatz zur Identifikation zusammengehöriger Datensätze dar. Dabei wird ein festgelegter Satz an Kriterien verwendet, um Datensatzpaare eindeutig einer Person zuzuord-

nen oder als nicht übereinstimmend zu klassifizieren. So kann beispielsweise definiert werden, dass ein Datensatzpaar dann als zusammengehörig gilt, wenn in den Matching-Variablen ”Name”, ”Vorname” und ”Geburtsdatum” eine Übereinstimmung festgestellt wird. Eine verbreitete Methode der deterministischen Datensatzverknüpfung basiert auf der Nutzung einer gemeinsamen, eindeutigen Kennung [20, S.7]. Besonders in Kohorten-Längsschnittstudien findet dieses Verfahren Anwendung, um Daten aus mehreren Erhebungswellen miteinander zu verknüpfen [33, S.955]. Ein Beispiel hierfür ist die Verknüpfung über die Sozialversicherungsnummer, die in beiden Datensätzen vorhanden sein muss. Allerdings können Kodierungsfehler oder fehlerhafte Einträge in einer der Dateien dazu führen, dass tatsächliche Übereinstimmungen nicht erkannt werden [38, S.1246].

Im Rahmen des deterministischen Record Linkage haben die verwendeten Matching-Variablen in der Regel denselben Stellenwert [22, S.8]. Dabei kann sowohl ein strenger, auf exakter Übereinstimmung basierender Abgleich erfolgen, als auch ein weniger strikter Ansatz, bei dem bestimmte Abweichungen innerhalb definierter Toleranzgrenzen zugelassen werden. Bei einem exakten Abgleich wird ein Datensatzpaar nur dann als übereinstimmend betrachtet, wenn dieses in allen festgelegten Matching-Variablen identische Ausprägungen aufweist, ohne Abweichungen oder Toleranzen [7, 32].

In der praktischen Anwendung kann auch eine weniger strikte Vorgehensweise als die exakte deterministische Verknüpfung gewählt werden, indem die Übereinstimmungskriterien für bestimmte Matching-Variablen flexibler definiert werden. Dabei wird geprüft, ob Abweichungen in diesen Variablen innerhalb eines festgelegten Toleranzbereichs liegen, der durch einen Schwellenwert festgelegt wird. Auf diese Weise wird nicht nur eine exakte Übereinstimmung, sondern auch eine gewisse Ähnlichkeit zwischen zwei Datensätzen berücksichtigt. Anschließend erfolgt eine Bewertung anhand zuvor definierter Regeln, ob ein Datensatzpaar als übereinstimmend gilt, basierend darauf, ob die relevanten Matching-Variablen innerhalb der zulässigen Toleranzgrenzen liegen [22, S.8,16].

Ein wesentlicher Nachteil des exakten Abgleichs besteht darin, dass tatsächliche Übereinstimmungen unentdeckt bleiben können, wenn Fehler in den Verknüpfungsvariablen enthalten sind [22, S.17]. Solche deterministische Verfahren sind demnach grundsätzlich anfällig für fehlende Übereinstimmungen, da beispielsweise Aufzeichnungsfehler oder unvollständige Angaben dazu führen können, dass einzelne Identifikatoren nicht exakt übereinstimmen. Gleichzeitig sind die Falschübereinstimmungsraten bei dieser Methode in der Regel gering, da die Wahrscheinlichkeit, dass unterschiedliche Personen zufällig in allen relevanten Identifikatoren übereinstimmen vergleichsweise niedrig ist [39]. Der deterministische Ansatz berücksichtigt außerdem nicht die Tatsache, dass einige Matching-Variablen eine höhere Trennschärfe besitzen als andere. Zur Überwindung dieser Einschränkung wurden probabilistische Verfahren entwickelt [33, S.955,956], [22, S.17].

Probabilistisches Record Linkage

Ein probabilistisches Record Linkage berücksichtigt explizit, dass die Verknüpfungsvariablen sowohl hinsichtlich ihrer Trennschärfe als auch ihrer Qualität variieren können. Dabei wird eine Übereinstimmung einer Variable, die eine hohe Diskriminierungsfähigkeit aufweist, stärker gewichtet als bei Variablen mit geringerer Aussagekraft. Diese Eigenschaften machen probabilistische Methoden, trotz ihrer höheren rechnerischen Komplexität, attraktiver als deterministische Verfahren [22, S.8-10]. Im Vergleich zu deterministischen Methoden bieten probabilistische Ansätze demnach oft präzisere und flexiblere Lösungen, insbesondere in Szenarien, in denen Quasi-Identifikatoren existieren [40, S.739]. Ein probabilistische Record Linkage wurde 1969 von Fellegi und Sunter [35] auf den Grundlagen der Arbeiten von Newcombe aus dem Jahr 1959 [34] entwickelt und hat sich als eine zentrale Methode in der Datenverknüpfung etabliert. Dieses Verknüpfungsverfahren basiert auf Wahrscheinlichkeits- und statistischen Modellen und geht davon aus, dass für zwei Quellen alle möglichen Datensatzpaare in die zwei disjunkte Mengen M (Matched) und U (Unmatched) unterteilt werden können. Ein Datensatzpaar gehört zu M, wenn die beiden Datensätze tatsächlich derselben Person zugeordnet sind, andernfalls wird es der Menge U zugeordnet. Diese Einteilung erfolgt durch die Prüfung der Übereinstimmung von Verknüpfungsvariablen innerhalb jedes Paares [22, S.12]. Um das Konzept zu verdeutlichen, wird angenommen, dass zwei Datensätze A und B aus zwei unterschiedlichen Quellen vorliegen, dessen Ausprägungen für eine Matching-Variable als a und b dargestellt werden. Die Mengen M und U können demnach wie folgt dargestellt werden [35]:

$$M = \{(a, b) \mid a = b, a \in A, b \in B\}$$
$$U = \{(a, b) \mid a \neq b, a \in A, b \in B\}$$

Das probabilistische Record Linkage lässt sich insgesamt in fünf Schritte unterteilen. In einem ersten Schritt erfolgt die Schätzung der sogenannten m- und u-Wahrscheinlichkeiten für jede Matching-Variable. Da die genannten Mengen M (Übereinstimmung) und U (Nicht-Übereinstimmung) unbekannt sind, müssen diese Wahrscheinlichkeiten geschätzt werden. Die u-Wahrscheinlichkeit gibt die Wahrscheinlichkeit an, dass die Ausprägungen einer spezifischen Variable k zufällig bei einem Vergleichspaar aus zwei Datensätzen übereinstimmt, die nicht zur selben Identität gehören und somit eine Nicht-Übereinstimmung darstellen. Im Gegensatz dazu bezeichnet die m-Wahrscheinlichkeit die Wahrscheinlichkeit einer Übereinstimmung für eine gegebene Variable k, wenn das Vergleichspaar tatsächlich einer Übereinstimmung entspricht [38, S.1247], [22, S.12]. Die beiden Wahrscheinlichkeiten können durch den von Fellegi und Sunter beschriebenen Erwartungsmaximierung (EM)-Algorithmus berechnet werden. Dieser iterative Optimierungsalgorithmus besteht aus dem E-Schritt (Erwartungsschritt) und dem M-Schritt (Maximierungsschritt) [31, S.51]. Zu Beginn des E-Schritts werden initiale Werte für die Pa-

rameter *m* und *u* festgelegt. Auf Grundlage dieser Werte werden die erwarteten Werte der Variablen berechnet, die als Grundlage für die anschließende Aktualisierung der Parameter im nächsten Schritt dienen. Im M-Schritt werden die Parameter *m* und *u* optimiert, basierend auf den im E-Schritt berechneten erwarteten Werten. Die so angepassten Parameter werden anschließend wieder in den E-Schritt eingebracht, um die erwarteten Werte erneut zu berechnen. Dieser iterative Prozess von Erwartung und Maximierung wird solange wiederholt, bis eine Konvergenz erreicht ist, die dann eintritt, wenn die Parameterwerte nur noch minimale Änderungen aufweisen [31, S.51]. Eine alternative Methode ist der Einsatz des Fuzzy-Algorithmus. In diesem Ansatz werden zufällig ausgewählte Datensätze verglichen, wobei für jedes Paar die Anzahl der Übereinstimmungen, Nicht-Übereinstimmungen und fehlenden Werte in Bezug auf die Verknüpfungsvariablen ermittelt wird. Die mittleren Werte dieser Beobachtungen werden dann verwendet, um die Parameter *u* und *m* zu schätzen. Darüber hinaus können im Vorfeld bekannte Informationen über die Wahrscheinlichkeitsverteilung der genutzten Variablen berücksichtigt werden [35]. So lässt sich beispielsweise der Parameter *m* ableiten, indem man die bekannte Fehlerrate eines Identifikators von eins abzieht [22, S.17,18], [41]. Die *m*-Wahrscheinlichkeit und die *u*-Wahrscheinlichkeit können wie folgt dargestellt werden [22, S.12]:

$$m(k) = P(a_k = b_k \mid a \in A, b \in B, (a, b) \in M)$$
$$u(k) = P(a_k = b_k \mid a \in A, b \in B, (a, b) \in U)$$

Im zweiten Schritt erfolgt die Gewichtung der Matching-Variablen basierend auf den Wahrscheinlichkeiten *m* und *u* [22, S.10]. Ein wesentliches Element des Modells von Fellegi und Sunter sind die feldspezifisch geschätzten Gewichte für jede Matching-Variable. Diese Gewichtung dient dazu, verschiedene Identifikatoren innerhalb der Datensätze entsprechend ihrer Aussagekraft und Relevanz zu differenzieren. In der Regel wird beispielsweise dem Geburtsdatum eine höhere Gewichtung zugewiesen als dem Geschlecht, da es eine größere individuelle Unterscheidung ermöglicht [40, S.739]. Auch die Anzahl möglicher Ausprägungen einer Matching-Variable beeinflusst ihre Gewichtung [8, S.15]. Die Gewichtung der Matching-Variablen erfolgt unter Verwendung der geschätzten Wahrscheinlichkeiten *m* und *u*, dabei steht die *m*-Wahrscheinlichkeit im Zähler und im Nenner steht die *u*-Wahrscheinlichkeit [42, S.2], [22, S.8]. Für jede Matching-Variable wird das individuelle Übereinstimmungsgewicht *w* berechnet, das sich aus diesen beiden Wahrscheinlichkeiten ableitet. Die Berechnung dieses Gewichts für jede Matching-Variable folgt der nachstehenden Formel [22, S.13], [31, S.51], [35]:

$$w = \log\left(\frac{m}{u}\right)$$

Der dritte Schritt umfasst die Berechnung der Ähnlichkeit zwischen den Ausprägungen einer Matching-Variable aus zwei Datensätzen. Dazu kommen unterschiedliche Algorithmen zum

Einsatz, die in Kapitel 2.1.4 dargestellt werden. Diese Verfahren ermitteln für jede verglichene Ausprägung der Matching-Variablen einen Ähnlichkeitswert, der zwischen 0 und 1 liegt. Ein Wert von 1 kennzeichnet eine vollständige Übereinstimmung, während 0 eine vollständige Diskrepanz darstellt [33, S.957], [31, S.41,50], [43, S.1]. Der Ähnlichkeitswert quantifiziert somit, inwieweit die Ausprägungen einer Matching-Variable zweier Datensätze übereinstimmen. Diese Werte werden verwendet, um den Vektor y im Vergleichsraum Γ zu erstellen, der sich im Intervall von 0 bis 1 erstreckt [31, S.41,50], [43, S.1]. Für jede Matching-Variable kann ein spezifischer Schwellenwert festgelegt werden, der von dem Ähnlichkeitswert erreicht oder überschritten werden muss, damit die zu vergleichenden Ausprägungen einer Matching-Variable als übereinstimmend gelten [33, S.961], [31, S.52]. Es ergeben sich zwei bedingte Wahrscheinlichkeiten für y [35]:

$$m(y) = P(y[a,b] \mid (a,b) \in M)$$
$$u(y) = P(y[a,b] \mid (a,b) \in U)$$

Im anschließenden vierten Schritt wird das Gesamtverknüpfungsgewicht R für das untersuchte Datensatzpaar bestimmt [22, S.10]. Dies geschieht durch die Berechnung des Verhältnisses der Wahrscheinlichkeit P, dass die Felder a und b anhand eines Ähnlichkeitsmaßes entweder der Menge der Matches oder der No-Matches zugeordnet werden können. Die Berechnung des Gesamtgewichts R für ein verglichenes Datensatzpaar lässt sich wie folgt darstellen [31, S.50]:

$$R = \frac{P(y \in \Gamma | (a,b) \in M)}{P(y \in \Gamma | (a,b) \in U)}$$

Eine Möglichkeit R zu berechnen, ist zunächst das Gewicht jeder einzelnen Matching-Variablen mit ihrem entsprechenden Ähnlichkeitswert zu multipliziert. Anschließend werden die Produkte derjenigen Variablen summiert, dessen Ähnlichkeitswert den festgelegten Schwellenwert erreicht oder überschritten hat. Parallel dazu erfolgt eine weitere Berechnung, bei der die Produkte derjenigen Variablen aufsummiert werden, dessen Ähnlichkeitswert unterhalb des gesetzten Schwellenwertes liegt. Schließlich wird die Summe der Produkte der Variablen mit einem Ähnlichkeitswert über dem Schwellenwert als Zähler und die Summe der verbleibenden Produkte als Nenner in das Verhältnis eingesetzt [44, S.3]. Das Gesamtverknüpfungsgewicht, das einem verglichenen Datensatzpaar zugewiesen wird, ergibt sich aus dieser Berechnung und dient als Maß für die Wahrscheinlichkeit einer tatsächlichen Übereinstimmung [29, S.6], [38, S.1248]. Ein höheres Gewicht weist auf eine größere Übereinstimmungswahrscheinlichkeit hin, während ein niedrigeres Gewicht darauf hindeutet, dass es sich mit hoher Wahrscheinlichkeit um kein Match handelt [38, S.1248].

Im fünften und letzten Schritt wird das Gesamtgewicht eines Datensatzpaares anhand zweier Schwellenwerte (Thresholds) klassifiziert (siehe Abbildung 2.1) [22, S.10]. Auf Basis der gewichteten Bewertung *R* erfolgt die Klassifizierung der Paare in die drei Kategorien ”Verknüpfung” (Match), ”Nicht-Verknüpfung” (No-Match) und ”potenzielle Verknüpfung” (Possible-Match), die einer manuellen Überprüfung unterzogen werden sollten (siehe Abbildung 2.1) [22, S.14], [39, S.259]. Fellegi und Sunter empfehlen die Festlegung zweier Schwellenwerte, um eindeutige Verknüpfungen von potenziellen sowie von nicht übereinstimmenden Datensätzen abzugrenzen. Es wird demnach ein oberer Schwellenwert T_μ und ein unterer Schwellenwert T_λ definiert. Wenn das Gesamtgewicht *R* des Datensatzpaars den oberen Schwellenwert T_μ erreicht oder überschreitet, wird das Paar als übereinstimmend definiert. Wenn der Wert den unteren Schwellenwert T_λ erreicht oder unterschreitet, wird das Datensatzpaar als nicht-übereinstimmend klassifiziert. Befindet sich *R* zwischen dem oberen und unteren Schwellenwert, erfolgt eine manuelle Zuordnung durch einen Menschen, um zu entscheiden, ob die Datensätze zu derselben Person gehören. Es ergeben sich somit folgende Regeln [45, S.2], [43, S.1]:

Wenn $R \geq T_\mu$	dann liegt ein Match vor
Wenn $T_\lambda < R < T_\mu$	dann liegt ein mögliches Match vor
Wenn $R \leq T_\lambda$	dann liegt kein Match vor

Die Festlegung der Schwellenwerte sollte auf eine Art erfolgen, bei der ein optimales Gleichgewicht zwischen der Anzahl der falsch positiven und falsch negativen Verbindungen gewährleistet wird, während gleichzeitig die Anzahl der zu überprüfenden Verbindungen minimiert wird [22, S.18]. Eine Erhöhung der Schwellenwerte führt dazu, dass Nicht-Übereinstimmungen häufiger erkannt werden, da der obere Schwellenwert seltener überschritten wird, was die Wahrscheinlichkeit für das Auftreten von Nicht-Übereinstimmungen steigert. Im Gegensatz dazu führt eine Senkung der Schwellenwerte dazu, dass der obere Schwellenwert schneller erreicht wird, wodurch mehr Übereinstimmungen identifiziert werden und weniger Nicht-Übereinstimmungen erkannt werden [38, S.1248], [31, S.37], [22, S.27], [21, S.1]. Die Wahl einer Verknüpfungsstrategie sollte stets an der spezifischen Anwendung oder der zugrunde liegenden Forschungsfrage ausgerichtet sein [21, S.1].

2.1.4. Algorithmen

Es gibt verschiedene algorithmische Ansätze zur Bestimmung des Übereinstimmungsgrads eines Datensatzpaares [31, S.41,50], [8, S.15]. Diese lassen sich in unterschiedliche Kategorien einteilen. Eine dieser Kategorien umfasst sogenannte Zeichenketten-Algorithmen, die auf der Analyse und dem Vergleich von Zeichenfolgen basieren. Sie identifizieren gemeinsame Muster

innerhalb der Zeichenketten und berechnen deren Ähnlichkeit oder Distanz. Ein weitere Kategorie ist die phonetische Kodierung, bei der Wörter in standardisierte Codes umgewandelt werden. Ziel dieses Verfahrens ist es, ähnlich klingenden Wörtern denselben oder zumindest einen ähnlichen Code zuzuweisen, um phonetische Ähnlichkeiten zu erfassen [1, S.77,78], [46, S.1].

Zeichenketten-Algorithmen

Eine spezifische Untergruppe der Zeichenketten-Algorithmen sind die sogenannten Editierdistanz-Verfahren, zu denen unter anderem die Levenshtein-Distanz und die Hamming-Distanz zählen. Diese Methoden berechnen die Differenz zwischen zwei Zeichenketten, indem sie die Anzahl der notwendigen Bearbeitungsschritte (Einfügen, Löschen und Ersetzen) von Zeichen ermitteln, die erforderlich sind, um eine Zeichenfolge in eine andere zu transformieren [33, S.963], [7, S.31], [1, S.78].

Zunächst wird die *Levenshtein-Distanz* als eine der bekanntesten Editierdistanzen vorgestellt. Entwickelt wurde sie von dem russischen Mathematiker Vladimir Levenshtein [1, S.78], [47]. Wie bereits erläutert, basiert die Editierdistanz auf der Berechnung der minimalen Anzahl an Operationen (Einfügen, Löschen und Ersetzen) von Zeichen, die erforderlich sind, um eine Zeichenkette in eine andere zu überführen. Wie in Tabelle 2.1 dargestellt, bildet die minimale Anzahl an benötigten Operationen die Levenshtein-Distanz [1, S.78].

Tabelle 2.1.: Ein Beispiel für die Levenshtein-Distanz auf Basis der Publikation [1]

Tier - Tor	Test - Test
1. T**i**er = T**o**er	/
2. To**er** = Tor	/
Levenshtein-Distanz = 2	Levenshtein-Distanz = 0

Bei der Anwendung der Levenshtein-Distanz muss ein Schwellenwert für die maximal zulässige Distanz zwischen zwei Zeichenketten festgelegt werden, ab dem eine Übereinstimmung angenommen wird [1, S.79]. Ein wesentlicher Faktor in diesem Zusammenhang ist die Länge der Zeichenkette. Während bei längeren Zeichenketten ein höherer Schwellenwert gewählt werden kann, um Abweichungen, beispielsweise durch Tippfehler, zu tolerieren, ist dies bei kürzeren Zeichenfolgen problematischer. Da kürzere Zeichenketten weniger redundante Information enthalten, wirken sich selbst geringe Abweichungen stärker auf das Ergebnis aus. Zudem weisen längere Zeichenfolgen tendenziell einen höheren Anteil gemeinsamer Zeichen auf, wodurch Unterschiede relativ gesehen weniger ins Gewicht fallen [1, S.79,80]. Ein weiterer Vorteil der Levenshtein-Distanz liegt in ihrem einfachen Konzept, das eine unkomplizierte Implementierung ermöglicht.

Die *Hamming-Distanz* misst die Anzahl der Positionen, an denen zwei Zeichenketten unterschiedliche Zeichen aufweisen. Sie kann sowohl für Zeichenketten gleicher Länge als auch für unterschiedlich lange Zeichenfolgen angewendet werden. Bei Letzteren werden die fehlenden Zeichen in der kürzeren Zeichenkette als Abweichungen gewertet, da sie in der längeren Zeichenkette vorhanden sind, jedoch keine Entsprechung finden [31, S.42]. Um den Vergleich für unterschiedlich lange Zeichenfolgen zu standardisieren, wird in diesem Fall die Anzahl der nichtübereinstimmenden Zeichen durch die maximale Länge der Zeichenkette geteilt [31, S.42].

Neben den Editierdistanzen gibt es auch *token-basierte Zeichenketten-Algorithmen*. Im Gegensatz zu zeichenbasierten Verfahren wie der Levenshtein-Distanz erfolgt der Vergleich hier nicht auf der Ebene einzelner Zeichen, sondern auf Grundlage von Token, die beispielsweise Wörter, N-Gramme oder Mengen umfassen können. Zwei Algorithmen dieser Kategorie sind der N-Gramm- (auch Q-Gramm-) Algorithmus sowie der Jaccard-Index.

Bei der N-Gramm-Methode (auch als Q-Gramm-Verfahren bezeichnet) wird eine Zeichenkette in kleinere Teilsequenzen unterteilt. Ein N-Gramm (oder Q-Gramm) bezeichnet dabei eine Teilzeichenkette mit einer festen vordefinierten Länge von q Zeichen. In der Praxis werden häufig Bigramme verwendet, bei denen die Teilsequenzen aus zwei Zeichen bestehen. Der Vergleich zweier Zeichenketten erfolgt anhand der Übereinstimmung der N-Gramme in beiden Zeichenketten. Eine hohe Anzahl an gemeinsamen N-Grammen weist auf eine hohe Ähnlichkeit der Zeichenketten hin [1, S.81,82], [48, S.84,85], [33, S.957]. Ein wesentlicher Unterschied zwischen Editierdistanzen und token-basierten Algorithmen liegt darin, dass bei token-basierten Verfahren die Position der Abweichungen eine Rolle spielt, während dies bei Editierdistanzen in der Regel nicht der Fall ist. Bei token-basierten Algorithmen haben Abweichungen, die am Rand der Zeichenkette auftreten, einen geringeren Einfluss auf das Ergebnis als Abweichungen, die zentral in der Zeichenkette liegen. Die Levenshtein-Distanz zwischen ”Thomson” und ”Thompson” sowie zwischen ”Thomson” und ”Thomsons” ist in beiden Fällen identisch und beträgt 1. Im Gegensatz dazu variiert jedoch die Anzahl der übereinstimmenden Trigramme. Sie beträgt 3 bei der zentral positionierten Abweichung (tho, hom, son). Von den insgesamt acht Trigrammen sind drei in beiden Worten enthalten (siehe Tabelle 2.2) [1, S.83].

Tabelle 2.2.: Ein Beispiel für die N-Gramm Methode auf Basis der Publikation [1]

Name	**Trigramme**					
Thomson	*tho*	*hom*	oms	mso		*son*
Thompson	*tho*	*hom*	omp	mps	pso	*son*

Ein weiterer token-basierter Algorithmus ist der Jaccard-Index. Dieser wird ebenfallsmverwendet, um die Ähnlichkeit zwischen zwei Mengen von Token zu bestimmen. Die Jaccard-Ähnlichkeit wird berechnet, indem die Anzahl der gemeinsamen Begriffe (Token), die in beiden Zeichenfolgen vorkommen, ins Verhältnis zur Gesamtanzahl aller einzigartigen Begriffe gesetzt wird [49]. Die Anzahl der gemeinsamen Elemente in beiden Mengen wird durch die Anzahl aller

unterschiedlichen Elemente aus beiden Mengen geteilt. Damit misst der Algorithmus, wie viele Token sich in zwei Zeichenketten überschneiden [48, S.112]. Um das Vorgehen zu verdeutlichen, werden die beiden Wörter *Maus* und *Haus* miteinander verglichen.

Token	Haus	Maus
H	1	0
a	1	1
u	1	1
s	1	1
M	0	1

Die Schnittmenge und Vereinigung der beiden Wörter sind:

$$\text{Schnittmenge} = \{a, u, s\} \quad (3 \text{ Token})$$

$$\text{Vereinigung} = \{H, a, u, s, M\} \quad (5 \text{ Token})$$

Der Jaccard-Index wird demnach wie folgt berechnet:

$$Jaccard\ Index = \frac{3}{5} = 0.6$$

Der Jaccard-Index für die Wörter *Haus* und *Maus* beträgt demnach 0.6.

Die Jaro-Ähnlichkeit ist ebenfalls ein Zeichenkettenalgorithmus, gehört jedoch nicht den Kategorien der Editierdistanzen oder der token-basierten Verfahren an. Er kombiniert Aspekte der Levenshtein-Distanz mit der Verwendung von N-Grammen und berücksichtigt die Transposition, wenn übereinstimmende Zeichen in einer unterschiedlichen Reihenfolge auftreten. Der Algorithmus berechnet daher die Ähnlichkeit basierend auf der Anzahl der Übereinstimmungen, der Transpositionen und der Längen der beiden Zeichenketten [31, S.43,44], [48, S.109]. Eine Erweiterung der Jaro-Ähnlichkeit ist die Jaro-Winkler-Ähnlichkeit, die von Winkler 1990 entwickelt wurde, um den Übereinstimmungen am Anfang einer Zeichenketten eine höhere Gewichtung zu verleihen. Dies erweist sich als besonders vorteilhaft in Anwendungen wie dem Namensvergleich, bei dem Zeichenketten häufig gemeinsame Präfixe aufweisen, die stärker gewichtet werden sollten. Dieser Bonus steigt mit der Anzahl der Übereinstimmungen am Beginn der Zeichenketten, was zu einer erhöhten Genauigkeit bei der Ähnlichkeitsberechnung führt, insbesondere bei Daten, in denen Präfixe eine bedeutende Rolle spielen [1, S.98]. Darüber hinaus existieren zahlreiche weiter Variationen dieses Algorithmus, die in unterschiedlichen Anwendungen zur Verbesserung der Genauigkeit herangezogen werden [33, S.958].

Phonetische Kodierung

Neben den Algorithmen der Zeichenkettenverarbeitung existieren auch phonetische Algorithmen, die Zeichenketten vor dem Vergleich in phonetische Codes umwandeln. Dabei erhalten Wörter, die ähnlich klingen, einen ähnlichen Code [7, S.31]. Die grundlegende Idee dieser Algorithmen liegt darin, nicht die Wörter selbst zu vergleichen, sondern deren vereinfachte Repräsentationen, die geringfügige Unterschiede in der Aussprache berücksichtigen [1, S.83]. Zu den phonetischen Algorithmen zählen unter anderem Soundex, Metaphone, Double Metaphone, Matching-Rating-Approach, Kölner Phonetik, Caverphone 1 und 2 sowie New York State Identification and Intelligence System (NYSIIS) [50, S.11], [7, S.31], [33, S.957].

Der *Soundex-Algorithmus* ist ein weit verbreitetes Verfahren zur phonetischen Kodierung, das bereits 1918 patentiert wurde [1, S.78]. Die Stärken von Soundex liegen in seiner Einfachheit und der Effizienz seiner Berechnung. Der Soundex-Code setzt sich aus dem Anfangsbuchstaben eines Wortes sowie drei weiteren Ziffern zusammen. Hierbei bleibt der erste Buchstabe unverändert, während die nachfolgenden Buchstaben in Zahlen umgewandelt werden. Diese Umwandlung erfolgt anhand einer festgelegten Transformationstabelle, die jedem Buchstaben eine bestimmte Ziffer zuweist und in Tabelle 2.3 dargestellt ist [48, S.74], [20, S.21], [48, S.83,84].

Tabelle 2.3.: Transformationstabelle von Soundex (reproduziert von [1])

Buchstabe	**Ziffer**
b f p v	1
c g j k q s x z	2
d t	3
l	4
m n	5
r	6

Vokale wie ”a” , ”e” , ”i” , ”o” und ”u” sowie die Buchstaben ”w” , ”y” und ”h” werden ausgelassen. Wenn durch die Transformation zwei gleiche Zahlen nebeneinander stehen, wird der Code angepasst, sodass nur eine der beiden Zahlen in der Zeichenkette verbleibt. Dieser Code wird auch ”Similarity Key” genannt. Wenn innerhalb des Similarity Keys nach dem Anfangsbuchstabe keine drei weiteren Zahlen vorhanden sind, werden diese Stellen mit Nullen aufgefüllt. Es ist wichtig zu beachten, dass Soundex auf englischen Schreibgewohnheiten basiert [1, S.83,85].

Es gibt jedoch für andere Sprachen angepasste Verfahren, wie zum Beispiel die *Kölner Phonetik* für die Deutsche Sprache, die von Postel 1969 auf den Grundlagen von Soundex entwickelt wurde. Bei diesem Verfahren wurde die Transformation der Buchstaben an die deutsche Sprache angepasst [1, S.100].

Neben Soundex findet auch der *NYSIIS-Algorithmus* in der phonetischen Kodierung häufig Anwendung [48, S.76,77]. Damit wurde Soundex 1970 um kontextspezifische Regeln erweitert [1, S.99]. Der NYSIIS berücksichtigt eine Vielzahl von sprachlichen und phonetischen Nuancen, was ihn komplexer macht als Soundex, sowohl in Bezug auf die Implementierung als auch auf die Berechnung. Ein wesentliches Merkmal von NYSIIS ist die Anwendung spezifischer Regeln für bestimmte Buchstabenkombinationen, die in der englischen Sprache sowie in anderen Sprachen häufig vorkommen. So wird beispielsweise die Kombination ”kn” zu ”n” umgewandelt, um dem Umstand zu berücksichtigen, dass das ”k” in dieser Kombination stumm ist [1, S.99]. Darüber hinaus transformiert der Algorithmus verschiedene Endungen von Namenszeichenketten. Die Kombinationen ”ee” und ”ie” werden durch ”y” ersetzt, während ”dt”, ”rt”, ”rd”, ”nt” und ”nd” alle durch ”d” ersetzt werden [48, S.76].

Im Jahr 1990 entwickelte der Software-Ingenieur Lawrence Philips eine weitere Variante des Soundex, die als *Metaphone* bekannt ist. Dieser Algorithmus beruht auf spezifischen Regeln zur Vereinfachung und orientiert sich an den gängigen Prinzipien der englischen Aussprache [1, S.100]. Metaphone ignoriert Vokale, die nach dem ersten Buchstaben folgen und reduziert das verbleibende Alphabet auf sechzehn Konsonantenlaute. Vokale bleiben nur erhalten, wenn sie als erster Buchstabe auftreten und doppelte Buchstaben werden bei der Erstellung des Codes nicht berücksichtigt. Der Laut ”th” wird unter der Verwendung von Metaphone durch die Null dargestellt, da er dem griechischen Theta ähnelt, während ”sh” durch ”X” kodiert wird [32, S.5,6].

Zehn Jahre später führte Lawrence Philips den *Double Metaphone-Algorithmus* ein. Dieser Algorithmus deckt die Besonderheiten einiger europäischer und asiatischer Sprachen ab. Angesichts der zunehmend multikulturellen Gesellschaft ist es wichtig, dass ein phonetischer Kodierungsalgorithmus in der Lage ist, nicht-englische Namen zu verarbeiten [48, S.78]. Allgemein scheinen die von Double Metaphone generierten Kodierungen näher an der korrekten Aussprache von Namen zu sein als die von NYSIIS [48, S.78]. Dennoch ist zu beachten, dass beide Algorithmen nicht durchweg konsistent angewendet werden. Insbesondere zeigt Double Metaphone Schwächen bei Namensvariationen aus ostasiatischen sowie südostasiatischen Ländern und von indischen Subkontinent sowie dem Nahen Osten und Afrika [1, S.100].

Die phonetische Kodierung *Caverphone* wurde im Rahmen des Caversham-Projekts entwickelt und zielt darauf ab, die phonetische Zuordnung englischer Namen zu optimieren, insbesondere in Neuseeland und in der Region um Dunedin. Die Kodierung umfasst zum einen die Version Caverphone 1.0, die im Jahr 2002 eingeführt wurde, und zum anderen die verbesserte Version Caverphone 2.0, die 2004 veröffentlicht wurde. Letztere bietet eine angepasste Methodik, die insbesondere für die häufigsten phonetischen Übereinstimmungen optimiert ist und deswegen in der praktische Anwendung häufiger verwendet wird [50, S.11].

Ein wesentlicher Aspekt des phonetischen *Match-Rating-Ansatzes* ist die Unterscheidung von Ähnlichkeiten, die darauf abzielt, die Anzahl der Zeichen zu bestimmen, die in zwei zu ver-

gleichenden Namen nicht übereinstimmen. Bei diesem Verfahren erfolgt die Analyse in zwei Richtungen. Zunächst von links nach rechts und anschließend von rechts nach links. Anschließend werden identische Zeichen entfernt und die verbleibenden Differenzen von der Zahl 6 subtrahiert. Das Ergebnis wird mit einem schwellenwertabhängigen Grenzwert verglichen, der sich nach der Namenslänge richtet. Zudem wird der kodierte Name auf maximal sechs alphanumerische Zeichen begrenzt, was die Verarbeitung vereinfacht. Dieser kodierte Name wird auch als „persönlicher numerischer Bezeichner“ bezeichnet [51, S.2].

2.1.5. Herausforderungen

Verknüpfungsfehler

Im Verlauf des Record Linkage Prozess können sich vier verschiedene Fälle ergeben. Datensatzpaare, die als zusammengehörig identifiziert werden, werden auch als ”Matches” oder ”Positives” bezeichnet. Im Gegensatz dazu bezeichnet man Datensatzpaare, bei denen angenommen wird, dass diese nicht zusammen gehören, als ”No-Matches” oder ”Negatives”. Es kann zwischen den Fällen ”True Positive”, ”False Positive”, ”True Negative” und ”False Negative” unterschieden werden (siehe Tabelle 2.4) [1, S.12].

Tabelle 2.4.: Darstellung der verschiedenen Fälle innerhalb des Record Linkage Prozesses (entnommen aus: eigene Aufnahmen)

Fall	**Beschreibung**
True Positive	Datensatzpaare die als Match klassifiziert wurden und zur selben Identität gehören
False Positive	Datensatzpaare die als Match klassifiziert wurden und zu verschiedenen Identität gehören
True Negative	Datensatzpaare die als No-Match klassifiziert wurden und zu verschiedenen Identität gehören
False Negative	Datensatzpaare die als No-Match klassifiziert wurden und zur selben Identität gehören

Es können sich daher falsche und fehlende Übereinstimmungen innerhalb des Record Linkage ergeben [24, S.1700]. Das Ziel jeder Record Linkage Lösung liegt daher darin, ausschließlich ”True Positives” und ”True Negatives” zu finden. Die beiden übrigen Fälle, ”False Negative” und ”False Positive”, repräsentieren, wie in Abschnitt 1.1 erläutert, den Homonym- und Synonymfehler [1, S.12]. Ein Beispiel für einen Synonymfehler ist die Änderung eines Nachnamens infolge einer Heirat oder Scheidung, wodurch eine Person fälschlicherweise als

zwei unterschiedliche Individuen erfasst wird. Im Gegensatz dazu kann ein Homonymfehler auftreten, wenn beispielsweise Zwillingen, die einen hohen Anteil an denselben identifizierenden Daten aufweisen, fälschlicherweise eine gemeinsame Identität zugeschrieben wird [52, S.2] Trotz kontinuierlicher Verbesserung der Verknüpfungsmethoden bleibt die grundlegende Herausforderung bestehen, ein ausgewogenes Verhältnis zwischen der Anzahl falsch positiver und falsch negativer Matches zu gewährleisten. Diese Herausforderung erfordert eine sorgfältige Abwägung verschiedener Faktoren, insbesondere die Festlegung von Gewichten und Schwellenwerten.

Einflussfaktoren

”Gleiches mit Gleiches zu verbinden stellt überall dort eine besondere Herausforderung dar, wo keine eindeutigen Identifikationsmerkmale vorliegen” [53, S.1]. Eine zentrale Ursache für Verknüpfungsfehler ist die Qualität der zugrunde liegenden Daten, denn fehlerhafte oder unvollständige Ausprägungen der Matching-Variablen erschweren die korrekte Identifikation von Datensätzen und erhöhen somit die Wahrscheinlichkeit von fehlerhaften Klassifizierungen [54, S.647]. Studien zeigen, dass die Fehlerquote in Krankheitsregistern für Variablen wie ”Nachname”, ”Vorname”, ”Postleitzahl” und ”Geburtsdatum” zwischen 4 % und 15 % liegen kann, während die Rate fehlender Werte zwischen 0 % und 9 % variiert [55, S.80]. Der Einfluss dieser Fehler variiert dabei je nach verwendeter Matching-Variable [31, S.11].

Datenerfassung

Die Qualität der zu vergleichenden Datensätze wird unter Anderem durch die Methoden der Datenerfassung beeinflusst [56, S.1]. Ein zentrales Problem bei der Integration verschiedener Datenquellen und -systeme besteht in den voneinander abweichenden Methoden der Erfassung, Speicherung, und Aktualisierung von Daten. Diese Unterschiede führen dazu, dass identische Informationen in variierenden Formaten und Kodierungen vorliegen können [48, S.41], [22, S.10]. Ein Beispiel hierfür ist die Kodierung der kategorialen Variable „Geschlecht“, die je nach Quelle unterschiedlich dargestellt werden kann, etwa als ”w/m” in einer Datenbank und als ”f/m” in einer anderen [56, S.1], [33, S.958]. Die Art und Weise, wie Daten in verschiedenen Quellen eingegeben werden, kann ebenfalls zu variierenden Schreibweisen und inkonsistenten Einträgen führen. Solche Abweichungen hängen maßgeblich davon ab, ob die Informationen manuell abgeschrieben, telefonisch übermittelt oder mittels Spracherkennung erfasst werden [1, S.21]. Besonders problematisch ist dies bei Namen, die Diakritika enthalten. Wird beispielsweise ein französischer Name auf einer deutschen Tastatur eingegeben, kann es leicht zu Fehlern kommen, indem Akzente falsch gesetzt, visuell ähnliche Buchstaben verwendet oder Zeichen gänzlich weggelassen werden [1, S.72,73]. Ein weiteres

häufiges Problem tritt auf, wenn Namen aus fremden Sprach- und Kulturräumen diktiert werden. Fehlende Vertrautheit mit der Aussprache kann dazu führen, dass Namen inkorrekt verstanden und somit fehlerhaft erfasst werden [31, S.20].

Die nicht einheitliche Verwendung von Präfixen, etwa ”Frau”, ”Dr.” oder ”Prof.”, sowie Suffixen wie ”Senior” kann zu zusätzlichen Inkonsistenzen in den Daten führen. Diese entstehen insbesondere dann, wenn solche Namenszusätze in einer Datenquelle erfasst werden, in einer anderen jedoch fehlen [33, S.958].

Fehler basierend auf der Art der Variable

Neben den verschiedenen Methoden der Datenerfassung können auch die spezifischen Eigenschaften der erfassten Variablen die Entstehung von Verknüpfungsfehlern beeinflussen. Grundsätzlich lassen sich Verknüpfungsfehler in Bezug auf String-Variablen, numerische Variablen, kategoriale Variablen und dynamische Variablen differenzieren. Bei String-Variablen können Schreibfehler auf verschiedene Weisen entstehen, darunter das Einfügen zusätzlicher Buchstaben, das Vertauschen von Zeichen innerhalb eines Wortes (Transposition) sowie das versehentliche Entfernen oder Ersetzen einzelner Zeichen [22, S.10]. Solche Fehler können auf motorische Unachtsamkeiten, Probleme bei der optischen Zeichenerkennung, orthografische Unsicherheiten oder ergonomische Einschränkungen der Tastatur, wie beispielsweise eine ungünstige ”Typing Distance”, zurückgeführt werden [1, S.71-73]. Dabei treten versehentliche Tastenanschläge häufiger bei benachbarten Buchstaben auf, etwa ”n” statt ”m”, als bei Zeichen, die weiter voneinander entfernt liegen. Obwohl solche Fehler in einigen Fällen unmittelbar erkannt und korrigiert werden können, bleiben sie insbesondere unter Zeitdruck oder bei abgelenkter Dateneingabe oft unbemerkt [48, S.47]. Ein weiteres Problem für den Datenabgleich stellen Schreibvarianten aufgrund von Tippfehlern dar, die zwar die phonetische Struktur eines Namens nicht verändern, jedoch unterschiedliche Schreibweisen aufweisen, wie beispielsweise „Meier“ und „Meyer“. Diese unterschiedlichen Schreibweisen werden als Homophone bezeichnet [48, S.46], [1, S.67]. Zusätzlich können bei String-Variablen wie dem Vorname sogenannte ”Multiple-Value-Felder” vorliegen, in denen mehrere Zeichenketten enthalten sind, die in ihrer Reihenfolge variieren können, etwa „Anna Lena“ und „Lena Anna“ [48, S.46], [33, S.958]. Numerische Variablen sind neben Tippfehlern auch anfällig für Zahlendreher, die zu fehlerhaften Werten führen können [52, S.2]. Insbesondere bei Messwerten besteht zudem das Risiko von Rundungsfehlern, wenn keine exakten Angaben vorliegen oder gerundete Werte verwendet werden [22, S.10]. Die kategorialen Variablen sind aufgrund ihrer standardisierten Kodierung, beispielsweise durch kurze Zeichenfolgen wie ”w” und ”m” zur Angabe des Geschlechts, in der Regel weniger anfällig für Eingabefehler. Dennoch können Inkonsistenzen in der Zuordnung auftreten, insbesondere wenn unterschiedliche Kodierungssysteme oder uneinheitliche Klassifikationen verwendet werden [22, S.10].

Dynamische Variablen, die sich im Zeitverlauf ändern können, stellen eine zusätzliche Herausforderung im Kontext der Datenverknüpfung dar [24, S.1700]. Während Vornamen in der Regel relativ stabil bleiben, unterliegen Nachnamen häufigen Veränderungen, beispielsweise durch Heirats- oder Scheidungsprozesse [20, S.9]. Darüber hinaus können sich die Adressen von Personen aufgrund eines Wohnortwechsels ändern. Solche Veränderungen führen dazu, dass in Datensätzen unterschiedliche Werte für die Matching-Variablen erscheinen, was potenziell falsche Zuordnungen zur Folge haben kann [24, S.1700]. In der heutigen Zeit ist es zudem, wenn auch seltener, möglich, dass sich das Geschlecht einer Person ändert. Die einzigen Merkmale, die in der Regel konstant bleiben, sind das Geburtsdatum und der Geburtsort [48, S.45].

Die identifizierten Fehler, die die Datenqualität beeinträchtigen, lassen sich in zwei Hauptkategorien ”zufällige Fehler” und ”systematische Fehler”. Zufällige Fehler, die zu Verknüpfungsfehlern führen, können in der Regel leichter statistisch korrigiert werden als systematische Fehler, die nicht auf Zufälligkeit basieren [9, S.2053]. Zu den systematischen Fehlern zählen unter anderem falsche Schreibweisen von Nachnamen, insbesondere bei Personen aus anderen Kulturkreisen, inkonsistente Kodierungen, Namensänderungen durch Heirats- oder Scheidungsprozesse sowie Änderungen der Adressen aufgrund eines Wohnortwechsels [22, S.10], [20, S.9]. Zufällige Fehler hingegen treten häufig in Form von Tippfehlern auf, die durch Einfügungen, Ersetzungen oder Zahlendreher in Namen, Geburtsdaten oder Postleitzahlen verursacht werden. Zufällige Fehler, die nicht vom Identifikationswert ausgehen und somit in jedem anderen Datensatz auch enthalten sein können, weisen trotz ihrer zufälligen Natur häufig gewisse Regelmäßigkeiten in den Fehlern auf [22, S.10]. Solche Regelmäßigkeiten können bei der Entwicklung von Matching-Verfahren berücksichtigt werden, um die Genauigkeit und Zuverlässigkeit des Datenabgleichs zu verbessern.

Neben fehlerhaften Einträgen in den Matching-Variablen kann auch das Fehlen von Werten die Qualität der Daten erheblich beeinträchtigen [24, S.1700]. Solche Lücken können den Datenabgleich erschweren, da möglicherweise nicht ausreichend Informationen vorliegen, um präzise Vergleiche und Klassifizierungen von Datensatzpaaren vorzunehmen [48, S.40].

Konsequenzen

Verknüpfungsfehler sind in jedem Projekt, dass eine Datenverknüpfung beeinhaltet, unvermeidlich [20, S.5]. Eine erhebliche Anzahl solcher Fehler kann jedoch gravierende Auswirkungen auf die Ergebnisse und Schlussfolgerungen von Studien haben [24, S.1700]. Das zuvor erläuterte DFG-Projekt in Kapitel 1.1 verdeutlicht die Relevanz dieser Herausforderung durch die Unterschätzung des Krebsrisikos infolge einer Vielzahl an Verknüpfungsfehlern [8, S.31,89]. Das Ausmaß der Auswirkungen variiert in Abhängigkeit des spezifischen Forschungsvorhabens, der Art der Fehler, der Anzahl der Fehler, den relevanten Variablen sowie der Verteilung

der Verknüpfungsfehler innerhalb dieser Variablen [[9] zitiert in [25, S.219]]. Die Toleranz gegenüber Fehlern und deren Bewertung in den Schlussfolgerungen muss stets individuell unter Berücksichtigung der jeweiligen Forschungsfrage erfolgen [25, S.219].

Verknüpfungsfehler, die mit der interessierenden Variable in Verbindung stehen, können zu Fehlklassifizierungen oder Messfehlern führen und damit die Wahrscheinlichkeit von Verzerrungen erhöhen [9, S.2053]. In wissenschaftlichen Untersuchungen spielen falsch negative Ergebnisse (Nachnamensänderung) sowie falsch positive Ergebnisse (Zwillingspaar) eine entscheidende Rolle bei der Bildung einer Stichprobe. Solche Verknüpfungsfehler können dazu führen, dass bestimmte Individuen zu Unrecht aus der Stichprobe ausgeschlossen werden, was in der Folge zu Fehlklassifizierungen führt, bei denen Daten fälschlicherweise einer bestimmten Kategorie zugeordnet werden. Darüber hinaus können Messfehler auftreten, die wiederum zu falschen Schlussfolgerungen führen [9, S.2058]. Falsche Verknüpfungen führen jedoch nur dann zu Fehlklassifizierungen oder Messfehlern, wenn die aus diesen Verknüpfungen abgeleiteten Informationen von den Informationen abweichen, die durch korrekte Verknüpfungen erhalten worden wären [9, S.2051]. Fehlende Verknüpfungen können beispielsweise dazu führen, dass der Krankheitsstatus einer Person fälschlicherweise als negativ klassifiziert wird, was zu einer systematischen Unterschätzung der Krankheitsrate führen kann. Im Gegensatz dazu können falsch positive Verknüpfungen zu einer Überschätzung der Prävalenz führen [9, S.2058], [25, S.220].

Einer der Konsequenzen von Fehlklassifizierungen ist die Entstehung von Informationsverzerrungen, die dazu führen, dass die erhobenen Daten die tatsächlichen Gegebenheiten nicht angemessen repräsentieren. Darüber hinaus können Selektionsverzerrungen entstehen, bei denen die Stichprobe nicht repräsentativ für die zugrunde liegende Population ist. Solche Ungenauigkeiten führen zu fehlerhaften Einschätzungen der Beziehungen zwischen Exposition und Wirkung innerhalb einer bestimmten Population. Zudem kann eine Selektionsverzerrung entstehen, wenn fehlende Übereinstimmungen in einer oder mehreren relevanten Variablen systematisch auftreten, was die Validität der Analyse weiter gefährdet [9, S.2052,2053], [20, S.9,12]. Demnach sind Fehler, die systematisch in bestimmten Personengruppen auftreten, von großer Bedeutung, denn diese dazu führen, dass einige Personengruppen über oder unterrepräsentiert sind, was wiederum das Gesamtergebnis beeinflusst [25, S.220].

Fälschliche Einbeziehungen oder Ausschlüsse von Individuen in einer Studienpopulation können außerdem ebenfalls die statistische Aussagekraft erheblich beeinflussen. Dies ist besonders problematisch, wenn die Studienpopulation aufgrund des versehentlichen Ausschlusses relevanter Personen signifikant verkleinert wird [25, S.220].

2.2. Einführung in HL7 FHIR

Dieses Kapitel bietet eine Einführung in den HL7 FHIR Standard. Neben einem Überblick über dessen Ursprung werden das Grundkonzept und die Architektur beschrieben sowie Bezüge zu bestehenden Interoperabilitätsstandards hergestellt.

2.2.1. Ursprung von HL7 FHIR

Jedes System speichert und verarbeitet Daten in einem eigenen internen Format. Damit der Datenaustausch zwischen verschiedenen Computersystemen reibungslos funktioniert, ist eine Übersetzung dieser Formate erforderlich. Dies kann durch die Umwandlung der ursprünglichen Daten in ein standardisiertes Übertragungsformat, das von allen beteiligten Systemen verstanden wird, realisiert werden. Entscheidend dabei ist, dass die semantische Bedeutung der übertragenen Informationen erhalten bleibt, auch wenn sich die strukturelle Darstellung unterscheidet. Die bloße Festlegung einer gemeinsamen Austauschsyntax reicht jedoch nicht aus, um Interoperabilität sicherzustellen. Eine erfolgreiche Interoperabilität erfordert nicht nur eine gemeinsame Sprache, sondern auch eine klare Übereinkunft darüber, welche Informationen unter welchen Bedingungen und zu welchem Zweck ausgetauscht werden [57, S.26,27,82].

Diese Herausforderung wird besonders im Gesundheitswesen deutlich, in dem die Standards für den Datenaustausch mit der erheblichen Variabilität klinischer Versorgungsstrukturen konfrontiert sind. Sowohl die Abläufe in der Patientenversorgung als auch die Art und Weise, wie medizinische Probleme erfasst und beschrieben werden, unterscheiden sich weltweit erheblich. Dies stellt internationale Interoperabilitätsstandards vor die Herausforderung, flexibel genug zu sein, um sowohl unterschiedliche Abläufe als auch variierende Verständnisse und Beschreibungsweisen derselben Sachverhalte zu berücksichtigen. Damit diese Standards in verschiedenen Ländern und Projekten anwendbar sind, müssen sie als Grundlage dienen, auf die spezifische Regelungen und Anpassungen aufgesetzt werden können. Grundsätzlich gibt es zwei Möglichkeiten, einen Standard mit dieser Herausforderung zu gestalten. Erstens könnte jedes potenziell relevante Datenelement im Voraus definiert werden. Zweitens könnte ein Basisdatensatz festgelegt werden, der es einzelnen Implementierungen ermöglicht, nach Bedarf zusätzliche Elemente hinzuzufügen. Beide Ansätze sind jedoch mit Nachteilen verbunden. Während die erste Methode zu einer umfangreichen Spezifikation führt, die wahrscheinlich in der Praxis aufgrund ihres hohen Aufwands selten vollständig genutzt werden würde, birgt die zweite Methode das Risiko, dass verschiedene Implementierer inkompatible Erweiterungen einführen. Dies würde die Skalierbarkeit und Interoperabilität erheblich einschränken [57, S.91,92].

Eine Organisation, die sich dieser Herausforderung widmet, ist HL7. Diese wurde 1987 mit dem Ziel gegründet einen umfassenden Rahmen zu Standardisierung des Austauschs, der Integration, der gemeinsamen Nutzung und des Abrufs elektronischer Gesundheitsdaten bereitzustellen. Um

dieses Ziel zu erreichen stellt HL7 ein Set von Standards für verschiedene Bereiche des Gesundheitswesens bereit. Diese Standards sollen die klinische Praxis sowie die Verwaltung, Bereitstellung und Evaluation von Gesundheitsdienstleistungen optimieren und dadurch die Interoperabilität im Gesundheitswesen maßgeblich verbessern. HL7 verfolgt dabei die Vision einer Welt, in der alle Menschen jederzeit und überall auf sichere, aktuelle und präzise Gesundheitsinformationen zugreifen können, um fundierte Entscheidungen über ihre Gesundheit zu treffen [58]. Die HL7-Standards sind weit verbreitet und werden insbesondere zur Systemintegration in Krankenhäusern genutzt.

Die Entwicklung der Interoperabilitätsstandards begann 1987 mit HL7 Version 2. HL7 V2 ist derzeit der am weitesten verbreitete Standard im Gesundheitswesen. Besonders relevant ist HL7 V2 für den Austausch in Krankenhäusern. Darüber hinaus wird der Standard auch intensiv im Bereich der öffentlichen Gesundheitskommunikation genutzt [58]. Im Jahr 1997 wurde HL7 Version 3 veröffentlicht. HL7 V3 verfolgt ein konsistentes, strukturiertes Modell, das Inkonsistenzen früherer Versionen wie V2 vermeiden sollte [58]. Im Jahr 2011 wurde dann die Clinical Document Architecture (CDA) als Teil des HL7 V3-Standards eingeführt, um eine standardisierte Struktur für den Austausch von medizinischen Dokumenten wie Befunden, Entlassungsberichten und Patientenakten zu schaffen. Die CDA basiert auf der Architektur von V3, ist jedoch eine eigenständige Spezifikation, die speziell auf klinische Dokumente ausgerichtet ist [57, S.80], [58]. Obwohl HL7 V3 seine eigenen Ziele erreicht hatte, konnte diese Version das übergeordnete Ziel, eine kostengünstigere und einfachere Interoperabilität sowie die Entwicklung wettbewerbsfähiger Standards, nicht vollständig erfüllen. Angesichts dieser Herausforderungen gründete HL7 eine ”Fresh Look -Taskforce”. Aus dieser Arbeit entstand das FHIR-Projekt, das auf einer Analyse der bestehenden Standards und erfolgreicher Beispiele für Interoperabilität basierte. Der erste Entwurf von FHIR, zunächst unter der Bezeichnung Resources for Health (RFH) bekannt, orientierte sich an den Prinzipien einer RESTful Application Programming Interface (API). Im Mai 2012 wurde FHIR schließlich offiziell eingeführt und gilt heute als der modernste HL7-Standard. Die aktuellste Version, HL7 FHIR Release 5 (R5), wurde am 26. März 2023 veröffentlicht [57, S.25,81], [58], [4].

Die zuvor erläuterte Herausforderung, einen Interoperabilitätsstandard zu entwickeln, der sowohl die Vielfalt klinischer Prozesse als auch unterschiedliche Verständnisse und Beschreibungsweisen berücksichtigt, ohne dabei eine übermäßige Komplexität zu erzeugen, wurde in den Designprinzipien von FHIR aufgegriffen. Zwei zentrale Ziele von FHIR bestehen darin, eine flexible Erweiterbarkeit für Implementierende zu ermöglichen und gleichzeitig sicherzustellen, dass die Verwaltung dieser Erweiterungen überschaubar bleibt. Darüber hinaus verfolgt das FHIR-Entwicklungsteam das übergeordnete Ziel, Interoperabilität so kosteneffizient zu gestalten, dass sie zur Selbstverständlichkeit im Gesundheitswesen wird. Dies soll gewährleisten, dass sowohl klinische als auch administrative Anwender jederzeit und ortsunabhängig auf die benötigten Informationen zugreifen können [57, S.79,93,99]. In diesem Zusammenhang vereint

HL7 FHIR die Stärken der bestehenden HL7-Standards, Version 2, Version 3 und die CDA und nutzt Webstandards wie Extensible Markup Language (XML), JavaScript Object Notation (JSON), Hypertext Transfer Protocol (HTTP) und Open Authorization (OAuth). Ein besonderer Fokus liegt auf der einfachen Implementierung, viele Entwickler sind in der Lage, bereits innerhalb eines Tages erste Schnittstellen erfolgreich einzurichten [57, S.79]. Der Einsatzbereich von FHIR erstreckt sich über verschiedene Bereiche des Gesundheitswesens, einschließlich der Human- und Veterinärmedizin, der klinischen Versorgung, der öffentlichen Gesundheit, klinischer Studien sowie administrativer und finanzieller Prozesse. Innerhalb dieser Anwendungsfelder wird FHIR beispielsweise für mobile Gesundheitsanwendungen, Cloud-basierte Kommunikation, den Austausch von Daten aus elektronischen Patientenakten und die serverseitige Interoperabilität genutzt [57, S.79], [58].

HL7 FHIR besteht aus zwei zentralen Komponenten, der technischen Spezifikation und der aktiven Community, die die Entwicklung und Pflege des Standards vorantreibt. Während die Spezifikation als dokumentierte Grundlage dient, liegt ihr eigentlicher Wert in der engagierten Gemeinschaft, die kontinuierlich an der Verbesserung und Weiterentwicklung arbeitet. Diese offene und kollaborative Struktur ermöglicht es Interessierten, sich aktiv an der Weiterentwicklung von FHIR zu beteiligen. Obwohl HL7 die übergeordnete Kontrolle über die Spezifikation behält, verfolgt die Organisation das Ziel, eine möglichst breite Beteiligung zu fördern. Die Community bildet zudem eine wichtige Schnittstelle zu anderen Standardisierungs- und Implementierungsgemeinschaften, wodurch eine enge Vernetzung und die kontinuierliche Weiterentwicklung des Standards gewährleistet werden [57, S.96,97], [4].

2.2.2. Grundkonzept und Architektur

HL7 FHIR ist ein internationaler Standard für den strukturierten und sicheren Austausch von Gesundheitsdaten zwischen IT-Systemen im Gesundheitswesen. FHIR basiert auf einer modularen Architektur, deren zentrale Bestandteile der FHIR-Server, die FHIR-Clients und die RESTful API sind, über die standardisierte Datenformate, sogenannte Ressourcen, verwaltet und ausgetauscht werden (vgl. Abbildung 2.2).

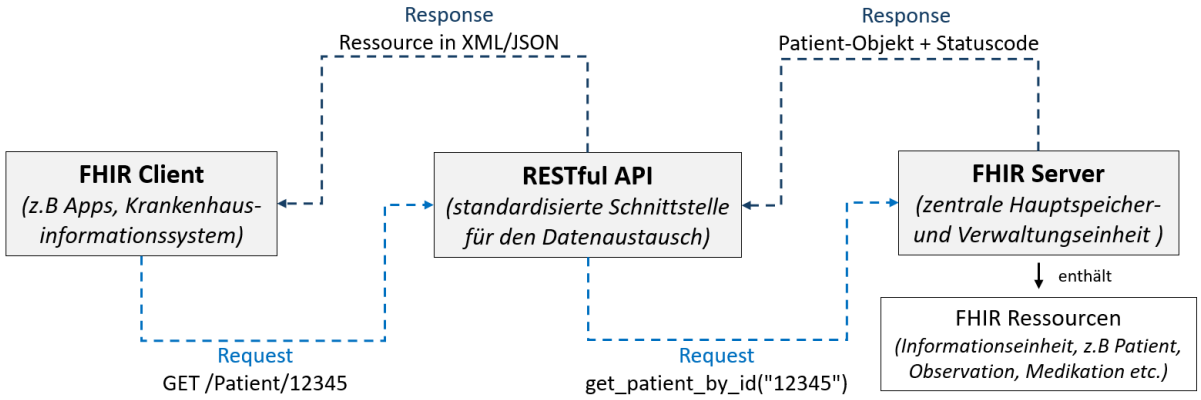


Abbildung 2.2.: Eine Übersicht über die grundlegende Architektur von HL7 FHIR (entnommen aus: eigene Aufnahmen)

FHIR Architektur

FHIR folgt einer modernen Client-Server-Architektur, bei der ein zentraler FHIR-Server als Hauptspeicher- und Verwaltungseinheit für Gesundheitsdaten fungiert. Externe Systeme, die auf diese Daten zugreifen oder sie aktualisieren möchten, werden als FHIR-Clients bezeichnet. Die Interaktion zwischen diesen Clients und dem Server erfolgt über eine RESTful API, einer standardisierte Schnittstelle, die den effizienten Austausch von Informationen ermöglicht [4].

Ein FHIR-Server ist vergleichbar mit einer digitalen Bibliothek für Gesundheitsdaten. Dieser speichert und verwaltet sämtliche FHIR-Ressourcen (z.B. Patientendaten, Diagnosen, Medikationspläne) und stellt diese bei Bedarf zur Verfügung. Der Server gewährleistet nicht nur die Datenvalidierung und die Zugriffskontrolle sondern auch die Datenverarbeitung. Transaktionen in diesem Framework werden direkt über HTTP-Anfragen auf der Serverressource durchgeführt. Die Entscheidung, welche Interaktionen und Ressourcentypen ein Server unterstützt, wird in einem ”Capability Statement” dokumentiert. Dieses Statement gibt den Clients die Möglichkeit, sich über die verfügbaren Funktionen des Servers zu informieren. Ressourcen werden dabei nach ihrem Typ organisiert. Der ”Type Manager” innerhalb eines FHIR-Servers übernimmt die Verantwortung für die Verwaltung und Verarbeitung dieser Ressourcentypen. Die Implementierung eines FHIR-Servers ist komplexer als die eines Clients, da der Server zahlreiche Aufgaben wie die Bereitstellung von Ressourcen, Validierung und Zugriffskontrollen übernimmt. Als Mindestanforderung muss der Server das Capability Statement bereitstellen [4]. Die Entwicklung eines FHIR-Servers erfolgt in der Regel nicht von Grund auf, stattdessen wird FHIR häufig als Schnittstelle zu bestehenden Systemen integriert. Je nach Ansatz kann die Speicherung der Ressourcen in relationalen Structured Query Language (SQL)-Datenbanken, dokumentenbasierten Systemen (z.B. MongoDB) oder hybriden Modellen erfolgen [57, S.181-183], [4].

Ein FHIR-Client stellt eine Anwendung oder ein System dar, das auf die von einem FHIR-Server gespeicherten Daten zugreift. Zu den typischen FHIR-Clients gehören Krankenhausinformati-

onssysteme, die Patientendaten verwalten, mobile Gesundheitsanwendungen, die Medikationspläne anzeigen, oder Laborinformationssysteme, die Testergebnisse speichern und versenden. Diese Clients nutzen die RESTful API von FHIR, um Daten zu lesen, zu speichern oder zu aktualisieren [4].

Die FHIR-API verfolgt einen datensatzzentrierten Ansatz für den Datenaustausch. Anstatt den Server direkt zu einer Operation aufzufordern, spezifiziert der Client, wie der Inhalt des Datensatzes aussehen soll [57, S.84]. Die RESTful API bildet demnach das zentrale Kommunikationsmittel zwischen einem FHIR-Client und einem FHIR-Server. Der Client sendet eine Anfrage (Request) an den Server, beispielsweise mit der Aufforderung, alle Patientendaten eines bestimmten Namens abzurufen. Der Server verarbeitet diese Anfrage und sendet eine Antwort (Response) zurück, die die angeforderten Daten in Formaten wie JSON oder XML enthält (siehe Abbildung 2.2). Um diesen Datenaustausch zu ermöglichen, verwendet die RESTful API standardisierte HTTP-Methoden, *GET* für den Abruf von Daten (z.B. zum Lesen von Patientendaten), *POST* für die Erstellung neuer Daten (z.B. das Hinzufügen eines neuen Patienten), *PUT* zur Aktualisierung bestehender Daten (z.B. eine Adressänderung) und *DELETE* für das Entfernen von Daten (z.B. das Löschen einer veralteten Diagnose). Neben den genannten *CRUD*-Operationen (Create, Read, Update, Delete) bietet FHIR auch spezialisierte Dienste, die über die Standardoperationen hinausgehen. Dazu gehören beispielsweise Anfragen zur Validierung von Ressourcen, das Verknüpfen von Patientendaten oder das Abrufen sämtlicher Datensätze eines bestimmten Patienten. Darüber hinaus können Server auch eigene zusätzliche Dienste definieren, die auf spezifische Anforderungen ihrer Implementierung zugeschnitten sind [57, S.84].

Die FHIR-Suche ist eine wesentliche Funktion zur Navigation und Verknüpfung von Ressourcen innerhalb eines Systems. Über *GET*-Anfragen an den Type Manager können Clients gezielt nach Ressourcen filtern, indem sie spezifische Kriterien angeben. Zum Beispiel liefert folgender Request alle Patienten des Geschlechts ”männlich”.

```
GET [base-address]/Patient?gender=male
```

Es lassen sich ebenfalls mehrere Parameter kombinieren:

```
GET [base-address]/Patient?name=peter&gender=male
```

So lassen sich alle männlichen Patienten mit dem Namen ”Peter” finden. Die FHIR-Spezifikation trennt den Inhalt einer Ressource von den durchsuchbaren Parametern, um die Suchleistung zu optimieren. Server nutzen vordefinierte, effizient indexierbare Suchparameter, die sowohl standardisiert als auch serverseitig erweitert werden können. Besonders nützlich ist die Verknüpfung von Ressourcen durch Suchanfragen, etwa mittels ”Parameter Chaining” oder ”Include”. Im Rahmen des Parameters Chaining werden Suchparameter über Referenzen verbunden, um alle Beobachtungen eines bestimmten Patienten zu erhalten:

```
GET [base-address]/Observation?subject=Patient/345
```

Die Include-Methode gibt verwandte Ressourcen direkt in der Suchergebnisliste zurück, was die Anzahl an separaten Anfragen reduziert und Netzwerklatenzen minimiert. Ein Beispiel hierfür ist:

```
GET [base-address]/Observation?code=1234-5&_include=Observation:subject
```

Es werden nicht nur alle Beobachtungen mit dem Code *1234-5* zurückgegeben, sondern auch die zugehörigen Patienten anzeigt. Zusätzlich zur Suche stellt die FHIR-API eine direkte Leseoperation bereit, die das Abrufen spezifischer Ressourcen anhand ihrer ID ermöglicht, um beispielsweise die Patientendaten mit der ID 345 abzurufen [57, S.105-111]:

```
GET [base-address]/Patient/345
```

Um eine neue Ressource zu erstellen, wird diese mittels einer *POST*-Anfrage an den entsprechenden Typ-Manager gesendet:

```
POST [Basisadresse]/[Typ]
```

Wenn die Ressource vom Server akzeptiert wird, weist der Server dieser Ressource eine neue Identität zu, speichert sie an einem definierten Ort und teilt dem Client den neuen Speicherort mit, damit dieser weiß, wo die Ressource abgelegt wurde. Der Umfang der Überprüfung der Inhalte und Geschäftslogik während dieses Prozesses wird vom Server bestimmt [57, S.109]. Zur Modifikation einer bestehenden Ressource sendet der Client die aktualisierten Daten über eine *PUT*-Anfrage an die spezifische Adresse der Ressource, zum Beispiel:

```
PUT [base-address]/Patient/345 <body: neuer Patientendatensatz>
```

Bevor eine Aktualisierung angenommen wird, prüft der Server die Gültigkeit der Ressource, stellt sicher, dass unveränderbare Elemente nicht verändert wurden und dass die referenzielle Integrität der Daten gewahrt bleibt. Die genauen Prüfregeln sind von der jeweiligen Implementierung abhängig [57, S.112].

Das FHIR-Team verfolgt das Ziel, den Austausch von FHIR-Daten nicht nur über die RESTful API, sondern auch über alternative Mechanismen wie Dokumente, Nachrichten und Dienste zu ermöglichen. Dies erweitert die Nutzung von FHIR über die REST-Architektur hinaus und schließt auch Anwendungen wie Workflows, Benachrichtigungen und strukturierte Berichte mit ein [57, S.100].

FHIR Ressourcen

Ein zentraler Baustein des FHIR-Standards ist die Ressource, die eine abgegrenzte Informationseinheit repräsentiert, beispielsweise ein Patient, eine Diagnose oder ein Rezept, damit verschiedene Systeme diese Informationen speichern und austauschen können. Ressourcen sind unabhängig von der jeweiligen Anwendung verwendbar, was an Bedeutung gewinnt, da Gesundheitsdaten zunehmend von Patienten selbst in ihrem jeweiligen Kontext geteilt werden [57, S.130]. FHIR definiert derzeit 157 Ressourcentypen (Stand R5), wobei die Anzahl mit jeder neuen Version moderat wächst, um neue Anwendungsfälle abzudecken [4]. Diese können in standardisierten Formaten wie JSON, XML und Resource Description Framework (RDF) übertragen werden, um Kompatibilität mit unterschiedlichen technischen Infrastrukturen sicherzustellen [57, S.126]. Die Ressourcen werden in verschiedene Kategorien unterteilt: ”Grundlagen”, ”Basis”, ”Klinisch”, ”Finanziell” und ”Spezialisiert”. Innerhalb dieser Kategorien erfolgt eine weitere thematische Differenzierung. So gehören beispielsweise die Ressourcen ”Patient” und ”Praktiker” zur Kategorie ”Basis” in der Gruppe ”Einzelpersonen”. Eine vollständige Auflistung der Ressourcentypen ist im Anhang A zu finden.

Alle Ressourcen folgen dabei einer einheitlichen Struktur. Sie basieren auf wiederverwendbaren Datentypen, die standardisierte Elemente definieren. Die FHIR-Spezifikation verwendet standardisierte Datentypen, um die strukturierte Repräsentation der Informationen zu ermöglichen. Diese umfassen sowohl grundlegende Basistypen als auch primitive, komplexe und spezialisierte Typen wie Referenzen oder Erweiterungen. Sie bilden die Grundlage für die konsistente Modellierung der Ressourcenelemente in FHIR [4]. Die Patientenressource stellt eine der zentralen und am häufigsten genutzten Ressourcen in FHIR dar, da sie grundlegende Anwendungsfälle wie den Zugriff auf Patientenakten und die Verwaltung von Patientenlisten unterstützt (siehe Abbildung 2.3).

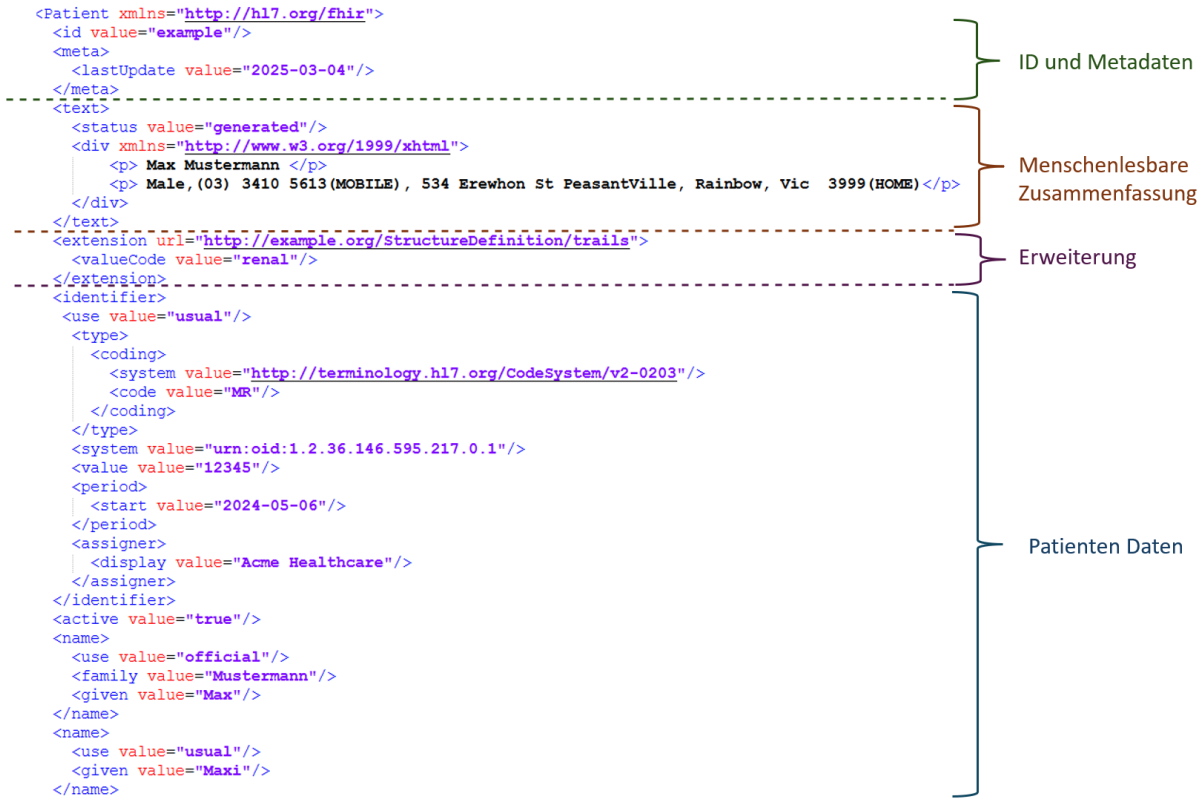


Abbildung 2.3.: Beispiel der Patient-Ressource im XML-Format [4] (entnommen aus: eigene Aufnahmen)

Die Patientenressource dient der Abbildung von Informationen sowohl über menschliche Patienten als auch über Tiere, die in verschiedenen gesundheitsbezogenen Kontexten eine Rolle spielen. Der Fokus liegt insbesondere auf demografischen Daten, die für administrative, finanzielle und logistische Prozesse von Bedeutung sind. Zu den zentralen Attributen gehören eine eindeutige Patientenkennung sowie weitere Identifikatoren, die eine zuverlässige Zuordnung innerhalb und zwischen Systemen ermöglichen. Ein wesentliches Merkmal dieser Ressource ist die Möglichkeit zur Verknüpfung von Patientenakten. Über das ”Link-Element” können Beziehungen zwischen verschiedenen Patientenressourcen hergestellt werden. Da Patientenakten in spezifischen Fällen zusammengeführt oder getrennt werden müssen, stellt die korrekte Verwaltung dieser Verknüpfungen eine zentrale Herausforderung für die Entwicklung von Gesundheitsinformationssystemen dar [57, S.144], [4].

Ressourcen enthalten, wie in Abbildung 2.3 dargestellt, einen Satz von Metadaten [4]. Das Metadaten-Element („meta“) einer Ressource umfasst verschiedene technische sowie workflowbezogene Informationen. Dazu zählen insbesondere die Versionierung, das Datum der letzten Modifikation, die Quelle der Ressource sowie spezifische Metadaten wie Profile, Sicherheitskennzeichnungen und Tags. Diese Metadaten spielen eine zentrale Rolle in der Kategorisierung, der Zugriffskontrolle und der strukturellen Definition von Ressourcen. Die Tags dienen der Verschlagwortung und ermöglichen eine gezielte Suche nach bestimmten Eigenschaften oder Themenbereichen, während Profile definieren, welche Elemente einer Ressource verpflichtend sind

und dessen Struktur an spezifische Anforderungen anpassen. Die Sicherheitskennzeichnungen des Metadaten-Elements legen die Datenschutzrichtlinien fest und bestimmen, in welchem Umfang auf sensible Informationen zugegriffen werden darf. Bei Änderungen an einer Ressource werden auch diese Kennzeichnungen angepasst, um den aktuellen Datenschutzvorgaben zu entsprechen.

Zur eindeutigen Identifikation von Ressourcen existieren in FHIR verschiedene Arten von Identifikatoren, die sich je nach Ressourcentyp und Anwendungsfall unterscheiden. Jede Ressource verfügt über eine ”logische ID”, die vom jeweiligen Server vergeben wird und innerhalb dieses Servers eindeutig ist. Diese ID dient der Identifikation einer spezifischen Instanz einer Ressource, ist jedoch nicht über verschiedene Server hinweg persistent, sodass eine solche ID nicht für eine langfristige Identifikation geeignet ist [57, S.129]. Neben der logischen ID existieren ”Business-Identifikatoren”, die reale Objekte, beispielsweise eine Patientennummer oder eine Medikamentenkennung, eindeutig kennzeichnen. Diese Identifikatoren gewährleisten eine systemübergreifende Konsistenz. Eine weitere Identifikator-Kategorie bilden ”Informations-Identifikatoren”, die für die eindeutige Kennzeichnung von Datensätzen verwendet werden, selbst wenn diese nicht direkt mit einem realen Objekt verknüpft sind. Sie finden beispielsweise Anwendung in internen Dokumentationsprozessen oder temporären Zwischenspeicherungen. Zusätzlich gibt es ”Instanz-Identifikatoren”, die jeder individuellen Kopie einer Ressource zugeordnet sind. Außerdem können sogenannte Universally Unique Identifier (UUID) vergeben werden. Eine UUID ist ein 128-Bit-Wert, der dazu dient, temporäre eindeutig erkennbare Referenzen zu erzeugen. Serverinterne UUIDs sind eindeutige IDs, die ein FHIR-Server einer Ressource zuweist und die nur innerhalb dieses Servers eindeutig sind. Logische Referenzen hingegen verwenden Identifiers, also externe oder geschäftliche Kennungen, um auf eine Ressource zu verweisen, ohne die konkrete Server-ID zu kennen oder zu benötigen. Während serverinterne UUIDs konkrete Ressourcen auf einem Server adressieren, ermöglichen systemübergreifende Verweise. Schließlich existieren noch kanonische URLs, die zur Identifikation standardisierter Inhalte wie Terminologie-Ressourcen oder Implementierungsleitfäden genutzt werden. Diese URLs sind global eindeutig und bleiben unverändert, sodass sie als feste Referenzpunkte innerhalb von FHIR dienen. Außerdem ist in einer Ressource auch der jeweilige Ressourcen Typ enthalten, wie beispielsweise ”Patient” [4].

Ein weiteres Merkmal der FHIR-Ressourcen ist die Möglichkeit, eine für Menschen lesbare Darstellung der enthaltenen Informationen bereitzustellen, das sogenannte Narrative. Diese Darstellung erfolgt in einem strukturierten Hypertext Markup Language (HTML)-Format und dient als ”Fallback-Anzeigeoption”, insbesondere in Umgebungen, in denen die direkte Verarbeitung strukturierter FHIR-Daten nicht gewährleistet ist [4]. Das Konzept des Narratives ist aus dem CDA-Standard abgeleitet. Die Verantwortung für die Generierung dieser Darstellung liegt bei dem System oder der Person, die die jeweilige Ressource erstellt. Zur Sicherstellung der Datensicherheit und Systemintegrität sind aktive Inhalte, darunter Formulare, Skripte, eingebettete

Objekte oder die Nutzung des lokalen Speichers, nicht zulässig [57, S.94].

Zusätzlich zu den Metadaten und der menschenlesbaren Zusammenfassung bestehen FHIR-Ressourcen aus verschiedenen Elementen, die strukturierte Datenfelder repräsentieren und die inhaltliche Definition einer Ressource prägen. Beispiele hierfür sind Angaben wie ”Name”, ”Geschlecht” oder ”Diagnose”, die in Abbildung 2.3 als ”Patienten Daten” zusammengefasst dargestellt sind. Jedes Element kann entweder einen primitiven Wert enthalten oder über untergeordnete Elemente weiter differenziert werden. Darüber hinaus sind jedem Element spezifische Attribute zugewiesen, die Informationen über Kardinalitätsregeln, zulässige Datentypen sowie weitere semantische und syntaktische Spezifikationen bereitstellen. Suchparameter sind ein zentrales Konzept in FHIR, mit denen Ressourcen über die RESTful API gezielt gesucht und gefiltert werden können. Jede Ressource definiert eigene Suchparameter, die der Client mit einem eindeutigen Namen anspricht und die der Server auf bestimmte Felder anwendet. Suchparameter sind standardmäßig aktiviert, können jedoch je nach Bedarf angepasst werden, um Speicherplatz und Leistung zu optimieren. Bei der Patient-Ressource sind beispielsweise Suchparameter wie name (Patientenname), identifier (Patienten-ID), birthdate (Geburtsdatum), gender (Geschlecht) und organization (verweisende Organisation) standardmäßig aktiviert. Damit bieten Suchparameter eine flexible und standardisierte Möglichkeit, medizinische Daten abzurufen [4].

Ein wesentliches Merkmal von FHIR-Ressourcen ist die Möglichkeit, Referenzen auf andere Ressourcen zu enthalten. Diese Referenzen werden genutzt, um Beziehungen zwischen verschiedenen Ressourcen herzustellen, ohne dass deren gesamte Inhalte innerhalb einer einzelnen Ressource dupliziert werden müssen. So kann beispielsweise eine Patientenressource eine Verknüpfung zu einer Practitioner-Ressource aufweisen, die den behandelnden Arzt repräsentiert. Eine solche Referenz besteht aus zwei zentralen Bestandteilen. Zum einen besteht die Referenz aus dem Typ der referenzierten Ressource und zum anderen aus der zugehörigen ID, die die spezifische Instanz dieser Ressource eindeutig identifiziert. Grundsätzlich wird dabei zwischen internen Referenzen, die auf Ressourcen innerhalb desselben Systems oder Servers verweisen und externe Referenzen, die den Zugriff auf Ressourcen ermöglichen, die in einem anderen System verwaltet werden, unterschieden. Durch diese Referenzierungsmöglichkeiten lassen sich komplexe Informationsnetzwerke aufbauen, in denen verschiedene Akteure des Gesundheitswesens miteinander verknüpft sind [57, S.90], [4].

FHIR verwendet eine Reihe spezialisierter Ressourcen, um den strukturierten Datenaustausch und die Interoperabilität zwischen Systemen zu gewährleisten. Hierzu gehört beispielsweise das ”CapabilityStatement”, die ”StructureDefinition-Ressource”, die ”Composition-Ressource”, die ”OperationOutcome-Ressource”, die ”Bundle-Ressource”, die ”List-Ressource” und die ”Konformitätsressource”. Eine zentrale Ressource in diesem Kontext ist das ”CapabilityStatement”, das die Funktionalitäten einer Implementierung beschreibt. Es legt fest, welche Schnittstellen verfügbar sind und welche Operationen sowie Ressourcen ein System unterstützt. Die ”StructureDefinition-Ressource” dient hingegen der Definition von Einschränkungen hinsicht-

lich der Kardinalität, zulässiger Datentypen, Terminologiebindungen und möglicher Erweiterungen, wodurch eine flexible Anpassung an spezifische Anforderungen ermöglicht wird [4]. Für die strukturierte Übertragung mehrerer Ressourcen innerhalb eines Transfers dient die ”Bundle-Ressource”. Diese wird beispielsweise für Suchergebnisse, Transaktionen oder Nachrichten genutzt und enthält eine eindeutige Kennung, einen Typ sowie optionale Verweise zur Navigation oder Kontextbereitstellung. Bundles besitzen keine eigenständige inhaltliche Bedeutung, sondern fungieren als organisatorisches Konstrukt zur Gruppierung mehrerer Ressourcen [57, S.144,145], [4]. Im Gegensatz dazu stellt die ”List-Ressource” eine inhaltliche Gruppierung dar, die beispielsweise zur Verwaltung von Medikamentenlisten oder stationären Patienten genutzt wird [57, S.145,146].

Da sich Anforderungen, regulatorische Vorgaben und praktische Gegebenheiten im Gesundheitswesen je nach Land, Institution und Anbieter erheblich unterscheiden, ist es häufig erforderlich FHIR an spezifische Rahmenbedingungen anzupassen. Um das zu ermöglichen, wurden in der Entwicklung von FHIR zwei zentrale Designziele verfolgt. Einerseits sollen Implementierende die Möglichkeit haben, Ressourcen flexibel zu erweitern, andererseits muss der Einsatz solcher Erweiterungen kontrollierbar bleiben. FHIR ermöglicht eine solche Anpassungen auf zwei Ebenen. Zum einen durch das ”Profiling” und zum anderen durch ”Extensions”. Während Profiling eine Modifikation der Struktur und der Anwendungsregeln einer Ressource erlaubt, ohne deren zugrunde liegende Definition zu verändern, bieten Extensions die Möglichkeit, zusätzliche Informationen zu speichern, die in der Standardressource nicht vorgesehen sind. In Bezug auf die Erweiterungen folgt FHIR der sogenannten 80/20-Regel. Rund 80% der typischen Anwendungsfälle werden durch die standardisierten Ressourcen abgedeckt. Die verbleibenden 20% können über Erweiterungen (Extensions) abgebildet werden, sodass auch projektspezifische oder lokale Anforderungen ohne Beeinträchtigung der Interoperabilität berücksichtigt werden können. Grundsätzlich kann jedes Element einer FHIR-Ressource neben seinem regulären Inhalt ein oder mehrere Extensions enthalten. Eine Extension besteht dabei aus zwei Komponenten. Einer eindeutigen URL zur Identifikation sowie einem Wert, der einem der grundlegenden FHIR-Datentypen entsprechen muss. Die angegebene URL verweist auf die offizielle Definition dieser Erweiterung innerhalb des FHIR-Standards. Diese URL dient nicht nur als eindeutige Kennung der Erweiterung, sondern ermöglicht es auch, eine formale Definition dieser Erweiterung abzurufen. Obwohl Entwickler eigene Erweiterungen definieren und veröffentlichen können, empfiehlt sich die Nutzung eines zentralen Registers, um bereits bestehende Erweiterungen zu identifizieren und deren Mehrfachentwicklung zu vermeiden. Die FHIR-Spezifikation fördert einen verantwortungsvollen Umgang mit Extensions durch mehrere Mechanismen. Erstens erleichtert die FHIR-API das Auffinden und Registrieren bestehender Erweiterungen, um deren Wiederverwendung zu fördern. Zweitens unterstützt die Community durch soziale Netzwerke die Verbreitung von Best Practices für die Entwicklung und Nutzung von Erweiterungen. Drittens wird der Austausch über Plattformen wie Stack Overflow oder spezialisierte Foren empfohlen, um Implementierende zu ermutigen, sich mit der Community abzustimmen [57, S.94,157,158], [4].

In FHIR bezeichnet Profiling den Prozess der Anpassung von Ressourcen und Datentypen an spezifische Anforderungen. Jedes Element innerhalb einer FHIR-Ressource oder eines Datentyps ist durch eine ”ElementDefinition” beschrieben. Diese enthält zentrale Informationen wie den Pfad des Elements, eine Beschreibung, Nutzungshinweise, die Kardinalität, den Datentyp sowie spezifische Regeln, etwa Terminologie-Bindungen. Alle Definitionen, die eine Ressource oder einen Datentyp beschreiben, werden in der ”StructureDefinition” zusammengefasst. Die StructureDefinition” beschreibt Einschränkungen sowie Erweiterungen mit dem ”Differential” und dem ”Snapshot”. Während das Differential ausschließlich die Abweichungen gegenüber der zugrunde liegenden Basisressource dokumentiert, stellt der Snapshot die vollständige Ansicht dar, die alle Anpassungen berücksichtigt. Falls eine Implementierung besondere Anpassungen erfordert, können diese durch zusätzliche Definitionen innerhalb einer ”StructureDefinition” vorgenommen werden. Diese Erweiterungen legen spezifische Regeln fest, die die ursprüngliche Definition verfeinern, ohne ihr zu widersprechen. Dieser Prozess wird als Profiling bezeichnet. Profile dienen demnach dazu, die Nutzung und Semantik von Ressourcen oder Datentypen weiter einzuschränken und zu präzisieren [57, S.164,165,168]. Profile werden durch die kanonische URL identifiziert, dadurch ist es möglich, die Konformität einer Ressource mit einem bestimmten Profil zu deklarieren [57, S.167,172].

Die Qualität der FHIR-Spezifikationen wird bei HL7 durch ein mehrstufiges Ballot-Verfahren sichergestellt. Entwürfe einzelner Ressourcen, Profile oder technischer Konzepte werden zunächst als ”Standard for Trial Use” veröffentlicht, von einer internationalen Community praktisch erprobt, kommentiert und überarbeitet, bevor sie nach mehreren Abstimmungsrunden als normative Standards verabschiedet werden. Auch von HL7 definierte Extensions können Teil dieses Ballot-Verfahrens sein, sofern sie Bestandteil offizieller Profile oder Implementierungsleitfäden sind. Lokale Erweiterungen hingegen unterliegen diesem Prozess nicht.

Codesysteme und Value Sets bilden in FHIR die Grundlage für die semantische Interoperabilität. Sie ermöglichen medizinische Informationen eindeutig, standardisiert und für verschiedene Systeme verständlich zu kodieren, was für die Interoperabilität zwischen unterschiedlichen Gesundheitssystemen von entscheidender Bedeutung ist. Ein Codesystem umfasst eine Sammlung von Codes, die jeweils eine spezifische Bedeutung repräsentieren. Dabei können sowohl externe, weit verbreitete Standards wie ”Systematized Nomenclature of Medicine (SNOMED)”, ”Logical Observation Identifiers Names and Codes (LOINC)” oder ”International Classification of Diseases (ICD)-10” als auch von HL7 selbst definierte Codesysteme verwendet werden. Eine detaillierte Liste empfohlener Codesysteme ist auf der offiziellen Website von HL7 zugänglich [4]. Jeder Code, der innerhalb einer FHIR-Ressource verwendet wird, setzt sich mindestens aus dem System, das die Quelle des Codes angibt (z.B. *http://loinc.org*), und dem Code selbst, der die spezifische Kennung darstellt (z.B. *718-7* für Hämoglobin) zusammen. Optional können zusätzlich Informationen wie die Version des Codesystems sowie ein Display-Wert, der eine menschenlesbare Bezeichnung (z.B. *Hämoglobin [Masse/Vol.] im Blut*) enthält, aufgeführt wer-

den. Neben einzelnen Codes verwendet FHIR auch Value Sets, um eine vordefinierte Auswahl gültiger Codes für spezifische Anwendungsfälle zu bieten. Ein Value Set könnte zum Beispiel eine Liste von Codes für Diagnosen enthalten, die für Abrechnungszwecke zugelassen sind. Wenn es erforderlich ist, Begriffe aus verschiedenen Codesystemen miteinander zu verknüpfen oder zu übersetzen, kommen Concept Maps zum Einsatz. Diese ermöglichen beispielsweise die Umwandlung eines SNOMED-Codes in einen ICD-10-Code, wodurch die Kommunikation zwischen Systemen mit unterschiedlichen Kodierungen ermöglicht wird [57, S.159-160], [4].

2.2.3. FHIR und bestehende Interoperabilitätsstandards

Neben dem weit verbreiteten Interoperabilitätsstandard HL7 FHIR existieren zahlreiche weitere Standards, die in unterschiedlichen Kontexten Anwendung finden. Dazu zählen unter anderem HL7 V2, HL7 V3, CDA, IHE, OpenEHR und Digital Imaging and Communications in Medicine (DICOM).

HL7 FHIR ist ein Interoperabilitätsstandard im Gesundheitswesen, der sowohl Gemeinsamkeiten als auch Unterschiede zu den weiteren HL7 Standards aufweist. Der Standard HL7 V2 dient seit über 30 Jahren dem elektronischen Austausch klinischer und administrativer Informationen zwischen IT-Systemen in medizinischen Einrichtungen wie Krankenhäusern und Laboren und zählt zu den weltweit am weitesten verbreiteten Standards (aktuell: Version 2.9) [57, S.213]. Die Nachrichtenstruktur ist hierarchisch organisiert und der Nachrichtenaustausch erfolgt ereignisgesteuert durch sogenannte Trigger Events, die sich aus einem Nachrichtentyp und einem zugehörigen Ereignis zusammensetzen. Zur Anpassung an systemspezifische Anforderungen können sogenannte Z-Segmente verwendet werden, die funktional den Extensions im FHIR-Standard entsprechen [59]. HL7 V3 basiert auf dem hierarchischen Reference Information Model (RIM), das zentrale Klassen wie *Act*, *Entity*, *Role* und *Participation* definiert, um medizinische Konzepte standardisiert abzubilden [59]. Im Gegensatz zum flexiblen HL7 V2 verfolgt V3 einen modellbasierten Ansatz zur Sicherstellung semantischer Interoperabilität [57, S.443–444]. Die Nachrichten werden mittels Refined Message Information Model (RMIM)-Diagrammen modelliert und in XML umgesetzt. Trotz begrenzter Verbreitung bildet das RIM die Basis langlebiger Standards wie der Clinical Document Architecture von HL7 [57, S.466]. Die CDA ist die am weitesten verbreitete Anwendung von HL7 V3 und dient der standardisierten Erstellung, Archivierung und dem Austausch medizinischer Dokumente im XML-Format, die langfristig sowohl menschen- als auch maschinenlesbar bleiben [57, S.234]. CDA-Dokumente können drei Ebenen enthalten: Level 1 mit Header und menschenlesbarem Body, Level 2 mit strukturiertem oder unstrukturiertem Body und Level 3 mit zusätzlichen maschinenlesbaren Daten [57, S.236-238], [58].

Während HL7 V2 auf einem nachrichtenbasierten Modell zur Übermittlung klinischer Informationen beruht, verfolgt HL7 FHIR einen ressourcenorientierten Ansatz, der auf modernen

Web-Technologien, insbesondere RESTful APIs, aufbaut. HL7 V3 hingegen basiert auf einem streng modellgetriebenen Konzept unter Verwendung des RIM, wodurch eine höhere semantische Präzision angestrebt wird. Im Gegensatz dazu zeichnet sich FHIR durch eine modulare und flexible Architektur aus, die eine anwendungsfallorientierte Anpassung erleichtert. Trotz konzeptioneller Unterschiede bestehen enge Beziehungen zwischen den HL7-Standards. FHIR integriert wesentliche Konzepte und Strukturelemente aus HL7 V2 und HL7 V3 in eine moderne und leicht implementierbare Systemarchitektur. CDA wurde speziell für den strukturierten Austausch medizinischer Dokumente entwickelt. FHIR adressiert vergleichbare Anwendungsfälle durch die Bereitstellung der Ressourcen ”Composition” und ”DocumentReference”, die die Verwaltung und Integration medizinischer Dokumente innerhalb FHIR-basierter Systeme ermöglichen. Darüber hinaus können bestehende CDA-Dokumente über FHIR-Mechanismen übertragen werden, wodurch Interoperabilität zwischen den Standards CDA und FHIR entsteht.

Die Organisation IHE wurde 1999 gegründet und fördert die Interoperabilität zwischen IT-Systemen im Gesundheitswesen, indem sie Vorgaben und Empfehlungen zur Anwendung bestehender Standards im klinischen Kontext bereitstellt. IHE definiert standard basierte Implementierungsprofile, die die praktische Anwendung von Kommunikationsstandards wie HL7 und DICOM in klinischen Bereichen regeln [57, S.437], [60]. Ein zentrales Profil ist Cross-Enterprise Document Sharing (XDS), das den standortübergreifenden Austausch medizinischer Dokumente mittels standardisierter Metadaten ermöglicht und fünf Akteure umfasst: *Document Source*, *Document Repository*, *Document Registry*, *Document Consumer* sowie *Patient Identity Source*. Die Registry verwaltet Metadaten, während Dokumente in Repositories gespeichert sind, was die Suche und den Zugriff optimiert [57, S.255–260]. XDS-Metadaten enthalten Informationen zu Dokumenttyp, Patient, Autor und Ereignissen, um strukturierte Verwaltung und Recherche zu gewährleisten [57, S.260–267]. Erweiterungen wie XDS-I, XDS-MS und XDS-Lab integrieren spezielle Formate wie DICOM-Bilder und CDA-Dokumente [60]. Weitere relevante Profile sind Patient Identifier Cross-reference Service (PIX) und Patient Demographics Query (PDQ) für Patientenidentitätsmanagement sowie Cross-Enterprise Document Media (XDM), Cross-Enterprise Document Reliable Interchange (XDR), Document Subscription (DSUB) und Modality Performed Procedure Step Query (MPQ) für Datenübertragung, Informationsabruf und sichere Kommunikation [57, S.267–268], [60].

FHIR bietet mit dem Profil Mobile Health Documents (MHD) eine Möglichkeit, die Funktionalitäten von XDS über RESTful APIs bereitzustellen, wodurch die Integration in unterschiedliche Systemarchitekturen erleichtert wird. In Zusammenarbeit mit IHE entwickelt die FHIR-Community die Ressourcen ”AuditEvent”, ”DocumentReference” und ”DocumentManifest”, welche eine RESTful-Anbindung von XDS-Repositories ermöglichen und so den interoperablen Dokumentenaustausch unterstützen [57, S.255,256], [60], [4].

OpenEHR stellt einen modellgetriebenen Ansatz zur strukturierten, herstellerunabhängigen und langfristig nutzbaren Speicherung von Gesundheitsdaten dar. Als internationaler, gemeinnüt-

ziger Standard verfolgt OpenEHR das Ziel, Interoperabilität durch eine offene Plattform mit wiederverwendbaren klinischen Informationsmodellen zu fördern [61]. Zentrale Elemente der Architektur sind ”Archetypen”, die medizinische Konzepte formal beschreiben und ”Templates”, die deren Kombination für spezifische Anwendungsfälle ermöglichen. Die Daten werden versionierbar in einem Electronic Health Record (EHR)-Repository gespeichert und können über standardisierte APIs in andere Systeme integriert werden [57, S.440–442], [61]. Ein zugrunde liegendes Referenzmodell gewährleistet die strukturierte, einheitliche Ablage verschiedenster Datentypen [61], [62].

Während OpenEHR Archetypen als strukturelle Vorlagen für Gesundheitsinformationen verwendet, konzentriert sich FHIR auf einen flexiblen, transaktionsbasierten Datenaustausch. Beide Ansätze können komplementär genutzt werden, wobei sich OpenEHR auf die langfristige Speicherung und Modellierung von Gesundheitsdaten mit einer datenmodellgetriebenen Architektur konzentriert, während FHIR einen flexiblen, ressourcenorientierten Ansatz für den schnellen, transaktionsbasierten Austausch von Daten über moderne Web-Technologien verfolgt.

DICOM ist ein internationaler Standard zur Speicherung, Übertragung und Darstellung medizinischer Bilddaten. Der Standard definiert einheitliche Dateiformate und Kommunikationsprotokolle, um Bilder in hoher Qualität und mit relevanten Zusatzinformationen zu übermitteln. DICOM wurde 1985 eingeführt und wird von der Medical Imaging & Technology Alliance (MITA) verwaltet. Er besteht aus 22 Hauptteilen und über 200 Ergänzungen, die kontinuierlich weiterentwickelt werden, um den technischen Fortschritt zu berücksichtigen [57, S.437], [63]. Eingesetzt wird DICOM in nahezu allen bildgebenden Verfahren wie Röntgen, MRT, CT oder Ultraschall und bildet die Grundlage moderner Picture Archiving and Communication Systems (PACS)-Systeme, die eine effiziente Speicherung, Verwaltung und Übertragung medizinischer Bilddaten ermöglichen. Neben Bilddaten werden umfangreiche Metadaten, etwa zu Patient, Untersuchung, Gerät und technischen Bildparametern gespeichert, die eine korrekte Einordnung, Analyse und Archivierung der Bilder sowie eine hohe diagnostische Qualität ermöglichen [63].

Zur Verknüpfung von FHIR und DICOM dient die FHIR-Ressource ”ImagingStudy”, die strukturierte Metadaten zu bildgebenden Untersuchungen bereitstellt und dabei auf DICOM-Objekte referenziert. Dabei abstrahiert die Ressource komplexe DICOM-Strukturen in eine verständliche Form für klinische Systeme, Patientenportale oder mobile Anwendungen. Die Ressource ”ImagingStudy” erlaubt etwa die Angabe von Modalität, Zeitstempeln, untersuchtem Körperbereich und Zugriffspfade auf die DICOM-Bilder über ”DICOMweb-URLs”. Diese semantische Brücke ermöglicht es, DICOM-Bilddaten in FHIR-konformen Workflows einzubetten, etwa zur Anzeige relevanter Bildgebung im Rahmen eines ”DiagnosticReport” oder zur Übergabe von Bildinformationen an Patienten. Trotz der Integration bleiben die Standards komplementär, DICOM ist im technischen Betrieb bildgebender Systeme verankert, während FHIR die interoperable Kommunikation strukturierter medizinischer Inhalte zwischen Systemen erleichtert [4, 63].

Im Bereich der semantischen Interoperabilität bestehen ebenfalls Überschneidungen. So kooperiert HL7 beispielsweise mit SNOMED International, um die Integration von SNOMED in HL7-Spezifikationen sicherzustellen. Diese Zusammenarbeit wird auch in FHIR fortgeführt, wobei HL7 und International Health Terminology Standards Development Organisation (IHTSDO) gemeinsam für die Spezifikation der Nutzung von SNOMED verantwortlich sind, einschließlich der Zuordnung von Ressourcenelementen zu den entsprechenden SNOMED-Definitionen. Ergänzend dazu wird der LOINC-Standard über den LOINC FHIR-Terminologieserver eingebunden, der den Zugriff auf LOINC-Daten mithilfe von FHIR-Mechanismen ermöglicht [57, S.98]. Darüber hinaus bestehen weitere Kooperationen zwischen HL7 und Organisationen, einschließlich SMART Health, die in Verbindung mit FHIR ein Framework für die Integration von Anwendungen in EHR bereitstellen [57, S.98].

3. Vorstellung der Record Linkage Lösungen

Im Folgenden werden die beiden im Rahmen dieser Arbeit zu vergleichenden Record Linkage Lösungen, HAPI FHIR MDM und E-PIX vorgestellt. Dabei erfolgt die Betrachtung aus verschiedenen Perspektiven, um einen Überblick über die jeweiligen Funktionsweisen der Lösungen zu geben.

3.1. E-PIX

Dieses Kapitel widmet sich dem Konzept der Record Linkage Lösung E-PIX. Dazu werden zunächst grundlegende Informationen vermittelt, bevor zentrale Funktionen wie die Verwaltung von Identitäten, der Matching-Prozess sowie die verschiedenen Konfigurationsmöglichkeiten erläutert werden.

3.1.1. Grundlegende Informationen

Die Record Linkage Lösung E-PIX wird seit 2009 durch die Universitätsmedizin Greifswald im Rahmen des GANI-MED-Projekts entwickelt und wurde 2014 erstmals als Teil des MOSAIC-Projekts unter Open-Source-Lizenz (AGPLv3) veröffentlicht [12, 13]. Der E-PIX ist kostenfrei verfügbar und kann für kommerzielle und nicht-kommerzielle Zwecke verwendet werden [2, S.12]. Die Lösung basiert auf einem probabilistischen Record Linkage Verfahren und implementiert das Konzept eines Master Patient Index (MPI)s [5]. Der E-PIX bietet verschiedene Schnittstellen zur Interaktion und Integration. So kann zum einen die Web-Oberfläche direkt im Browser genutzt werden, oder eine Simple Object Access Protocol (SOAP)-Schnittstelle die maschinenlesbare Kommunikation über das SOAP-Protokoll verwendet werden [2, S.81]. Darüber hinaus unterstützt der E-PIX die IHE-Profile PIX und PDQ sowie HL7 FHIR [2, S.15], [13]. Tabelle 3.1 bietet eine Übersicht der wichtigsten Funktionen des E-PIX in Abhängigkeit von der jeweiligen Zugriffsmethode. Es wird deutlich, dass sowohl die SOAP-Schnittstelle als auch die Web-Oberfläche den größten Funktionsumfang abdecken, einschließlich der Anlage, Suche und Aktualisierung von Personen, der Duplikaterkennung sowie der manuellen Zusammenführung und Trennung von Datensätzen. Die FHIR-basierte REST-Schnittstelle unterstützt bereits grundlegende Operationen vollständig. Funktionen zur Dublettenauflösung und zur manuellen Verwaltung von Personen sind für die FHIR-Schnittstelle zwar bereits entwickelt, wurden jedoch im aktuellsten Release (Version 2024.3.1, März 2025) noch nicht implementiert.

Tabelle 3.1.: Funktionale Übersicht des E-PIX nach Zugriffsmethode (entnommen aus: eigene Aufnahmen)

Funktion	**SOAP**	**Web-Oberfläche**	**FHIR (REST)**
Domäne, Datenquelle und Identifier-Domäne anlegen	✓	✓	–
Personen anlegen	✓	✓	✓
Personen suchen	✓	✓	✓
Personen aktualisieren	✓	✓	✓
Personen löschen	✓	✓	–
Entfernen von Matches aus eine Domäne	✓	✓	*
Manuelles Trennen	✓	✓	*
Manuelles Zusammenführen	✓	✓	*
Export einer Liste von Possible Matches	✓	✓	*
Konfiguration der Matching-Regeln	✓	✓	–
Abrufen von Protokollen	✓	✓	–
Abrufen von Statistiken	✓	–	–

✓= Vollständig verfügbar – = Nicht verfügbar
* Funktionen via FHIR sind bereits spezifiziert, jedoch in der Version 2024.3.1 (März 2025) noch nicht implementiert.

Die folgende Abbildung 3.1 zeigt die Web-Oberfläche des E-PIX, die die Nutzung der zuvor genannten Funktionen in Tabelle 3.1 ermöglicht.

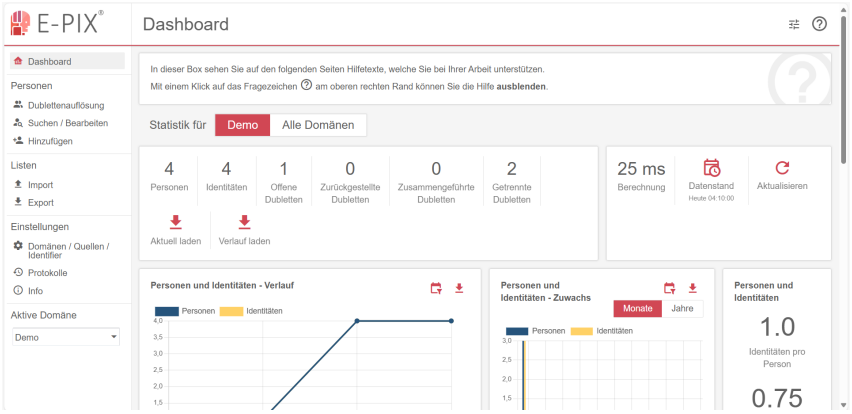


Abbildung 3.1.: Darstellung der Web-Oberfläche des E-PIX mit Ansicht des Dashboards (entnommen aus: [5])

Die Bereitstellung des E-PIX erfolgt regulär als Docker-Container, wobei die Nutzung von Docker und Docker-Compose empfohlen wird. Alternativ kann der Dienst innerhalb eines WildFly-Applikationsservers als Servlet betrieben werden. Für den Betrieb sind ein aktuelles Java Development Kit (JDK) ab Version 17, WildFly 26 sowie MySQL 8 erforderlich [2, S.16,17,22].

Um bestehende Anwenderprojekte sowie zukünftige Nutzer bei der Implementierung von FHIR-basierten Infrastrukturen und Prozessen zu unterstützen, wird das Treuhandstellen-FHIR-Gateway (Trusted Third Party (TTP)-FHIR-Gateway) bereitgestellt [6]. Dieses Gateway dient als Schnittstelle zwischen den FHIR-spezifischen Komponenten und dem E-PIX. Die Kommunikation mit dem Gateway erfolgt über eine RESTful API, die den FHIR-Standards folgt. Für das Record Linkage wird der folgende FHIR-Endpunkt bereitgestellt: ”`http[s]://<host>:<port>/ttp-fhir/fhir/epix`” [6]. Innerhalb des TTP-FHIR-Gateway kommen spezifische FHIR-Profile zum Einsatz, um die Verwaltung von Patienten- und Personenidentitäten im E-PIX zu standardisieren. Die beiden zentralen Profile sind das ”Person-Profil” und das ”Patient-Profil”.

```
<Person xmlns="http://hl7.org/fhir">
    <id value="Person-example-1" />
    <meta>
        <lastUpdated value="2021-06-17T08:28:03.200+02:00" />
        <profile value="https://ths-greifswald.de/fhir/StructureDefinition/
            epix/Person" />
    </meta>
    <identifier>
        <system value="https://ths-greifswald.de/fhir/identifier/
            BeispielMpiId" />
        <value value="1001000000066" />
    </identifier>
    <managingOrganization>
        <identifier>
            <value value="MIRACUM" />
        </identifier>
    </managingOrganization>
    <active value="true" />
    <link>
        <target>
            <reference value="Patient/52" />
        </target>
    </link>
    <link>
        <target>
            <reference value="Patient/53" />
        </target>
        <assurance value="level4" />
    </link>
</Person>
```

MPI-ID

Domäne

Verweis auf weitere Identitäten

Verweis auf Patient 53 mit dem höchsten Vertrauensniveau

Abbildung 3.2.: Beispiel des Person-Profils in E-PIX (entnommen aus: [6])

Das in Abbildung 3.2 dargestellte ”Person-Profil” repräsentiert eine reale Person und ermöglicht die Abbildung verschiedener Identitäten oder Schreibweisen dieser Person. Dabei wird die aktuell als korrekt anerkannte Variante einer Person durch die Angabe von *link.assurance='level4'* eindeutig gekennzeichnet. Innerhalb des FHIR-Modells existiert stets genau ein Link mit diesem höchsten Assurance-Level, um eine eindeutige Zuordnung sicherzustellen. Das ”Patient-Profil” beschreibt Patientenidentitäten, die einer realen Person zugeordnet werden und unterschiedliche Schreibweisen oder administrative Varianten enthalten können (siehe Abbildung 3.3).


```
1  <Patient xmlns="http://hl7.org/fhir">
2      <id value="Patient-example-2" />
3      <meta>
4          <lastUpdated value="2021-05-19T17:50:23.000+02:00" />
5          <profile value="https://ths-greifswald.de/fhir/
             StructureDefinition/epix/Patient" />
6      </meta>
7      <extension url="https://ths-greifswald.de/fhir/
         StructureDefinition/epix/CustomIdatValues">
8          <extension url="value1">
9      <valueString value="A928374650" />
10   </extension>
11   <extension url="value2">
12     <valueString value="2000-06-05" />
13   </extension>
14         <extension url="value3">
15             <valueBoolean value="true" />
16         </extension>
17     </extension>
18     <identifier>
19         <system value="https://ths-greifswald.de/fhir/epix/
             identifier/SystemXY" />
20         <value value="ABC_12345" />
21     </identifier>
22     <name>
23         <family value="Mustermann" />
24         <given value="Manfred" />
25     </name>
26     <telecom>
27         <system value="email" />
28         <value value="manfred.mustermann@example.org" />
29     </telecom>
30     <telecom>
31         <system value="phone" />
32         <value value="012345/5555567" />
33     </telecom>
34     <gender value="male" />
35     <birthDate value="1953-12-11" />
36     <address>
37         <line value="Musterweg 22" />
38         <city value="Musterstadt" />
39         <postalCode value="12345" />
40     </address>
41 </Patient>
```

Abbildung 3.3.: Beispiel einer einzelnen Identität im Patient-Profil in E-PIX (entnommen aus: [6])

Die in Abbildung 3.3 (Zeile 7) eingesetzte Extension stellt eine strukturierte Erweiterung zur Abbildung zusätzlicher identifizierender Daten innerhalb einer FHIR-Ressource dar. Sie wurde im Rahmen vom E-PIX definiert, um personenbezogene Merkmale zu erfassen, die nicht im FHIR-Standardmodell vorgesehen sind, etwa ein Diagnosedatum oder andere kontextspezifische Angaben. Es handelt sich dabei um eine benutzerdefinierte, frei strukturierte Erweiterung, die bis zu zehn Sub-Extensions (z.B value1, value2 usw.) aufnehmen kann. Die konkrete Ausgestaltung dieser Felder ist projektspezifisch und ermöglicht eine flexible Erweiterung des Ressourceninhalts im Rahmen definierter FHIR-Strukturen. Durch die Trennung von Person- und Patienten-Profil wird sichergestellt, dass sowohl die eindeutige Identifikation einer Person als

auch die flexible Verwaltung mehrerer Patientenidentitäten innerhalb des E-PIX gewährleistet ist. Für die Verarbeitung von Personendaten im FHIR-Format wurden spezifische Funktionalitäten implementiert, die nach der erfolgreichen Einrichtung des TTP-FHIR-Gateways über die REST-Schnittstellen zur Verfügung stehen. Die ”addPatient-Operation” ermöglicht das Anlegen neuer Patient-Ressourcen im E-PIX und deren Abgleich mit bereits bestehenden Identitäten. Im Zuge des Record Linkage Prozess werden eine oder mehrere Patientenidentitäten erstellt. Nach dem Record Linkage Prozess werden für jede übermittelte Patient-Ressource die MPI-ID, die MPI-Zuordnung (Person-Ressource) sowie der Match-Status und vorhandene Identitäten zurückgegeben. Voraussetzung für die Nutzung diese Operation ist die vorherige Konfiguration der relevanten Parameter für Matching-Domäne und Datenquelle innerhalb des E-PIX. Der Aufruf erfolgt per POST-Request an `/$addPatient` [6].

Die ”updatePatient-Operation” dient der Aktualisierung bestehender Patienteninformationen im E-PIX unter Angabe einer zuvor vergebenen MPI-ID. Hierbei werden die Patienten-Identitäten eines bestehenden MPI-Eintrags aktualisiert. Damit die Aktualisierung erfolgreich durchgeführt werden kann, müssen wie bei der Nutzung der ”addPatient-Operation” die zugehörige Matching-Domäne, die Datenquelle und die MPI-ID bereits im E-PIX hinterlegt sein. Der Aufruf dieser Operation erfolgt ebenfalls per POST-Request an `/$updatePatient` [6]. Erweiterungen und zusätzliche Funktionalitäten werden kontinuierlich entwickelt (z.B Auflösen von Matches) [2, S.111, 112].

3.1.2. Verwaltung von Identitäten

Der E-PIX implementiert ein System aus Haupt- und Nebenidentitäten. Einer Person kann demnach mehrere Identitäten zugewiesen werden, wobei lediglich eine als Hauptidentität festgelegt wird, die als korrekte Variante der Ausprägungen gilt. Alle weiteren Ausprägungen werden als Nebenidentitäten gespeichert [2, S.13,14]. Für jede registrierte Person wird eine eindeutige Kennung erzeugt, die als MPI-ID bezeichnet wird und innerhalb einer bestimmten Domäne eindeutig bleibt [2, S.30]. Bevor der Record Linkage Prozess des E-PIX konfiguriert wird, beinhaltet der erste Schritt das Anlegen einer Domäne, einer Identifier-Domäne und einer Datenquelle (siehe Abbildung 3.4). Die Herkunft der Personendaten wird durch die jeweilige Datenquelle bestimmt, dies kann beispielsweise ein Krankenhausinformationssystem, eine klinische Studie oder ein Forschungsnetzwerk sein. Jede registrierte Person erhält eine Hauptidentität, die mit einer sicheren Datenquelle verknüpft ist. Diese Quelle gilt als primärer Referenzpunkt für die identifizierende Daten (IDAT) und legt fest, welche Daten als verbindlich angesehen werden [2, S.25].

Darüber hinaus verwendet der E-PIX ”Identifier-Domänen” zur Verwaltung von Identifikatoren innerhalb spezifischer Kontexte. Dazu gehören sowohl die automatisch generierte MPIs als auch externe Identifikatoren wie Fallnummern. Jede Identifier-Domäne verfügt über eine eindeutige

Bezeichnung sowie ein Object Identifier (OID). Je nach Anwendungsfall kann eine Identifier-Domäne von mehreren Domänen gemeinsam genutzt werden [2, S.25,26,28,30].

Die Domäne bildet den kontextuellen Rahmen für das Record Linkage innerhalb von E-PIX. Sie kann beispielsweise ein Forschungsprojekt oder eine standortübergreifende Patientenverwaltung repräsentieren. Innerhalb einer Domäne ist jede Person eindeutig registriert, kann jedoch in mehreren Domänen parallel existieren. Jede Domäne ist einer spezifischen Identifier-Domäne sowie einer sicheren Datenquelle zugeordnet [2, S.26].

3.1.3. Matching-Prozess

Der E-PIX stellt zwei verschiedene Betriebsmodi zur Verfügung. Im ersten Modus führt die Lösung ein Record Linkage durch, um Personendaten abzugleichen und MPIs zu vergeben. Alternativ lässt sich die Lösung so konfigurieren, dass die Datensätze lediglich gespeichert, jedoch nicht miteinander abgeglichen werden [2, S.35]. Im Rahmen der Klassifizierung eines Datensatzpaares kann zwischen vier verschiedenen Matching-Typen unterschieden werden (siehe Tabelle 3.2) [2, S.35,62].

Tabelle 3.2.: Matching-Typen mit zugehörigen Ereignissen und Handlungen im E-PIX auf Basis von [2, S.62,63,83,84]

Matching-Typ	**Ereignis**	**Handlung**
Perfect-Match	Exakte Übereinstimmung	Automatische Zusammenführung, es wird keine neue Person oder Identität angelegt
Automatic-Match	”threshold-automatic-match” wurde erreicht/überschritten	Kennzeichnung Haupt- und Nebenidentität
Possible-Match	”threshold-possible-match” wurde erreicht, aber ”threshold-automatic-match” nicht überschritten	Manuelle Prüfung & Generierung einer vorläufigen MPI-ID
No-Match	”threshold-possible-match” nicht erreicht	Falls Person unbekannt, wird eine neue Identität als Hauptidentität angelegt
Multiple-Match	Mehrere potenzielle Matches	Generierung einer Liste möglicher Matches

Für die manuelle Prüfung der potenziellen Dubletten steht eine detaillierte Übersicht über Possible-Matches zur Verfügung. In dieser Übersicht werden die relevanten Personendatensätze in tabellarischer Form präsentiert, wobei Unterschiede in den einzelnen Variablen farblich hervorgehoben werden. Wenn beide Datensätze zu derselben Person gehören, wird der als korrekt

erachtete Datensatz mit dem anderen als Nebenidentität verknüpft. Handelt es sich hingegen um zwei unterschiedliche Personen, werden die Datensätze getrennt und aus der Dublettenprüfung entfernt. Jede Dublettenauflösung kann mit einem Kommentar ergänzt werden, um die getroffene Wahl zu dokumentieren. Sollte eine sofortige Klärung nicht möglich sein, beispielsweise weil zusätzliche Informationen benötigt werden, kann der Vorgang vorübergehend zurückgestellt werden. In diesem Fall wird der Eintrag aus der Liste der offenen Dubletten entfernt, bleibt jedoch für eine spätere Überprüfung verfügbar [2, S.54,88,89].

3.1.4. Konfigurationsoptionen

Die Konfiguration einer Domäne legt fest, unter welchen Bedingungen zwei Datensätze als Identitäten derselben Person betrachtet und entsprechend mit der gleichen MPI versehen werden. Die Konfiguration einer Domäne spielt somit eine zentrale Rolle bei der Klassifizierung der Datensatzpaare und erfolgt, wie in Abbildung 3.4 dargestellt, nachdem die Domänen und die Datenquelle angelegt wurden.

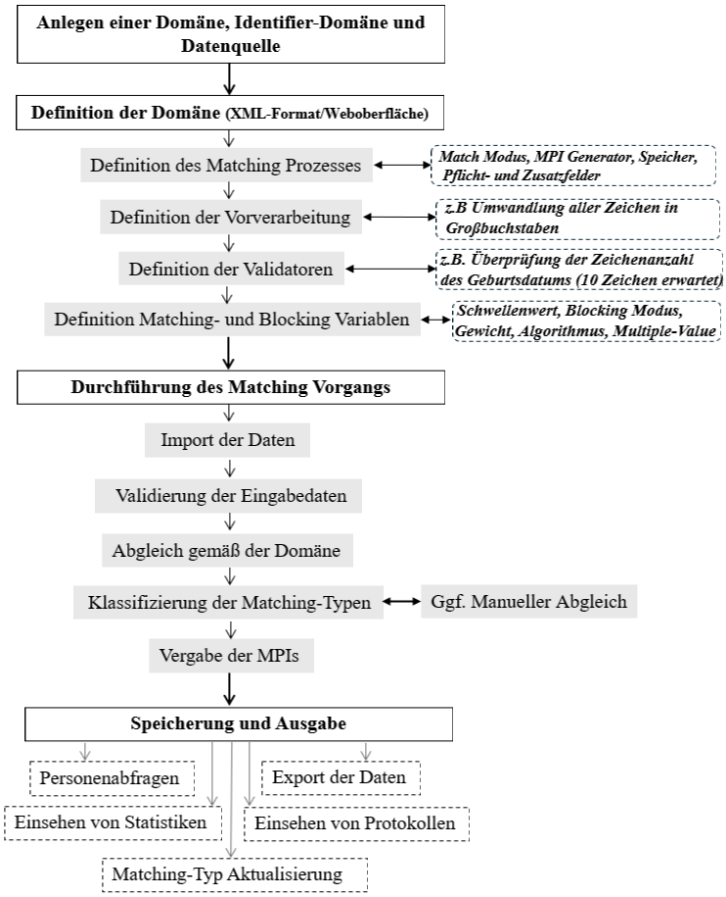


Abbildung 3.4.: Darstellung des Matching-Prozesses des E-PIX [5] (entnommen aus: eigen Aufnahmen)

Im Rahmen der Konfiguration einer Domäne, stehen verschiedene Elemente zur Verfügung, die spezifisch angepasst werden können (siehe Abbildung 3.5).

Abbildung 3.5.: Darstellung aller Elemente, die innerhalb einer Domäne konfiguriert werden können. (entnommen aus: [2])

Das Element ”MatchingConfiguration” stellt das übergeordnete Element dar, dem alle anderen Elemente untergeordnet sind. Die Struktur dieser Konfiguration legt fest, welche Elemente hierarchisch voneinander abhängen [2, S.46]. Die Anpassung der Domäne kann vollständig über die Weboberfläche vorgenommen werden. Zusätzlich besteht die Möglichkeit, eine bereits vordefinierte XML-Datei über die SOAP-Schnittstelle zu nutzen (gemäß Listing Anhang B) [2, S.45,46]. Als erstes können gemäß Abbildung 3.4 grundsätzliche Einstellungen des Matching Prozesses angepasst werden, die im Anhang B im zugehörigen Listing in den Zeilen vier bis 18 definiert sind. Diese beinhalten unter anderem die Konfiguration der MPIs über das Element ”mpi-generator”. Dieses Element ermöglicht die Implementierung zusätzlicher Generatoren für die Erstellung von MPIs. Außerdem können Benachrichtigungen über Änderungen im E-PIX aktiviert oder deaktiviert werden, die an andere Systeme, wie dem Treuhandstelle (THS)-Dispatcher, übermittelt werden. Dies kann über das Element ”use-notifications” gesteuert werden [2, S.47,48]. Auch der maximale Speicherbedarf lässt sich durch entsprechende Einstellungen vergeben. Das Element ”limit-search-to-reduce-memory-consumption” ermöglicht eine Reduzierung des Arbeitsspeichers, indem die Suchattribute einschränkt werden können. Wenn dieses Element auf ”true” gesetzt wird, erfolgt die Suche nur anhand der Felder, die tatsächlich für das Matching verwendet werden. Wenn dieses Element auf ”false” gesetzt wird, können demnach Personen anhand aller verfügbaren Attribute gesucht werden. Das Element ”persist-mode” legt dabei fest, wie die Daten im E-PIX gespeichert werden. In diesem Fall gibt es zwei mögliche Modi. Der Modus ”IDENTIFYING” sorgt dafür, dass alle übermittelten Daten gespeichert werden, während im Modus ”PRIVACY_PRESERVING” nur Daten, die einem ”Bloomfilter-

Ziel-Feld" entsprechen, beibehalten werden. Dabei handelt es sich um eine codierte Form von Daten, die im Rahmen des Privacy-Preserving Record Linkage (PPRL) genutzt wird, um Datensätze abzugleichen, ohne personenbezogene Informationen direkt offenzulegen. Denn der E-PIX unterstützt neben dem Abgleich von Personendaten im Klartext auch ein PPRL. Dabei werden Personendaten verschlüsselt, sodass sich diese nicht mehr auf die tatsächliche Identität der Person zurückgeführen lassen. Trotzdem bleibt es möglich, anhand dieser verschlüsselten Daten Vergleiche durchzuführen [2, S.13].

Es ist ebenfalls erforderlich, die Pflichtfelder für die Registrierung festzulegen, was über das Element "required-fields" gesteuert wird (siehe Listing im Anhang B Zeile: 9-14). Hierbei können bestimmte Felder wie "Vorname", "Nachname", "Geburtsdatum" und "Geschlecht" als verpflichtend angegeben werden. Standardmäßig sind diese Felder bereits als Pflichtfelder definiert. Darüber hinaus besteht die Möglichkeit, die Felder "value1" bis "value10" für benutzerdefinierte Werte zu nutzen (vgl. Listing im Anhang B Zeile: 15-18) [2, S.31,50].

Die Konfiguration des PPRL Verfahrens erfolgt über das Element "privacy", das mehrere "bloomfilter-config"-Elemente enthalten kann [2, S.54-58]. Standardmäßig wird kein Bloomfilter aktiviert, sodass seine Nutzung eine projektspezifische Anpassung erfordert [2, S.37-39].

Nach der Definition der grundlegenden Einstellungen des Matching-Prozess kann gemäß Abbildung 3.4 die Vorverarbeitung der Daten konfiguriert werden, die im Listing in Anhang B in den Zeilen 19 bis 144 beispielhaft definiert ist. Die Record Linkage Lösung unterscheidet dabei zwischen Ersetzungen, Umwandlungen und Filtern. Bei einer Ersetzung wird eine festgelegte Zeichenkette durch eine andere ersetzt, falls die neue Zeichenkette leer ist, wird die ursprüngliche Zeichenkette entfernt. Für gängige Anpassungen, wie die Umwandlung von Umlauten, stehen vordefinierte Umwandlungen zur Verfügung. Der E-PIX bietet vier Standardumwandlungen an, die in diesem Kontext verwendet werden können (siehe Tabelle 3.3).

Tabelle 3.3.: Standardumwandlungen der Vorverarbeitung im E-PIX auf Basis von [2]

Standardumwandlung	**Ereignis**
ToUpperCaseTransformation	Wandelt alle Zeichen in Großbuchstaben um, sodass Unterschiede in der Groß- und Kleinschreibung beim Record Linkage keine Rolle spielen.
CharsMutationTransformation	Ersetzt Umlaute durch ihre entsprechenden Umschreibungen (z. B. „ä“ → „ae“, „ß“ → „SS“)

CharNormalizationTransformation	Konvertiert Zeichen in das ASCII-Format, wodurch z. B. Akzente entfernt werden. Dabei werden Umlaute nicht in Umschreibungen wie „ae“, sondern in ihre Grundform überführt (z. B. „ä“ → „a“). Diese Umwandlung kann mit der CharsMutationTransformation kombiniert werden.
TrimTransformation	Entfernt führende und nachfolgende Leerzeichen (z. B. „ Müller “ → „Müller“).

Im Rahmen des Filterns von Daten wird definiert, welche Zeichen in den Eingaben zulässig sind. Zeichen, die nicht den festgelegten Kriterien entsprechen, werden entweder durch ein vordefiniertes Ersatzzeichen ersetzt oder, wenn kein Ersatzzeichen angegeben ist, entfernt [2, S.34]. Für die Datenvorverarbeitung wird das Element ”preprocessing-config” gemäß des Listings in Anhang B genutzt, das die Auswahl der verschiedenen Vorverarbeitungsoptionen für die Variablen ermöglicht. Innerhalb dieses Elements werden im ”preprocessing-field” sowohl die zu bearbeitende Variable als auch die anzuwendenden Transformationen spezifiziert. Dabei wird zwischen einfachen Transformationen (siehe Listing in Anhang B Zeile: 20-25), wie etwa dem Ersetzen bestimmter Zeichenfolgen und komplexeren Transformationen (siehe Listing in Anhang B Zeile: 74-76), wie der Normalisierung von Zeichen oder der Umwandlung in Großbuchstaben, unterschieden. Das Element ”simple-filter-type” ermöglicht außerdem das Filtern von Zeichen, wobei ein Alphabet definiert werden kann, um unerwünschte Zeichen zu entfernen [2, S.59-62].

Um Verknüpfungsfehler zu vermeiden, die aus unterschiedlichen Datenformaten resultieren können, kann innerhalb der Konfiguration einer Domäne in einem nächsten Schritt auch eine Validierung der eingegebenen Personendaten konfiguriert werden (vgl. Abbildung 3.4). Dieser Validierungsprozess findet nur statt, wenn für die entsprechenden Variablen mindestens ein Validator definiert wurde. So muss beispielsweise das Geburtsdatum im Format ”TT-MM-JJJJ” vorliegen. Entspricht das eingegebene Datum nicht diesem Format, wird der Registrierungsvorgang abgebrochen, da mindestens eine Ausprägung einer Variable nicht valide ist [2, S.32,96,97]. Standardmäßig sind für die Variablen keine Validatoren hinterlegt, doch für das Geburtsdatum und das Geschlecht werden immer Validierungsprüfungen durchgeführt, da diese Daten in einem bestimmten Format gespeichert werden [2, S.52,53].

Im E-PIX kommt das probabilistische Record Linkage nach Fellegi und Sunter zum Einsatz, um Wahrscheinlichkeiten für Übereinstimmungen zu bestimmen. Das Element ”Matching” (siehe Listing in Anhang B Zeile: 145) enthält zahlreiche weitere Unterelemente, die eine detaillierte Anpassung des Record Linkage und der Matching- sowie Blocking Variablen ermöglichen. Die Konfiguration des Abgleichs und der Variablen bildet daher den nächsten Schritt (siehe Abbildung 3.4). Zunächst können die Schwellenwerte für potenzielle und automatische Übereinstimmungen über die Elemente ”threshold-possible-match” und ”threshold-automatic-match” fest-

gelegt werden (siehe Listing in Anhang B Zeile 146-148). Das nächste Element ”use-cemfim” definiert, wie das System reagiert, wenn ein übermittelter Identifier mit einer Identität übereinstimmt, aber auch eine Übereinstimmung mit einer anderen Person vorliegt. Je nach Konfiguration dieses Elements (true oder false) wird die Identität entweder als potenzieller Treffer gespeichert oder als Fehler behandelt. Weiter können die Elemente ”parallel-matching-after” und ”number-of-threads-for-matching” verwendet werden, um den Einsatz von Multithreading beim Record Linkage zu steuern. Das Element ”parallel-matching-after” legt fest, ab welcher Anzahl registrierter Identitäten der Abgleich parallel auf mehreren Threads erfolgt, um die Performance zu optimieren. Die Anzahl der verwendeten Threads wird dabei durch das Element ”number-of-threads-for-matching” bestimmt, wobei standardmäßig vier Threads verwendet werden [2, S.65].

Unter Verwendung des Elements ”field” werden die Variablen ausgewählt, die für das Matching und/oder Blocking verwendet werden sollen (vgl. Listing in Anhang B Zeile: 151-184). Wenn innerhalb des entsprechenden ”field-Elements” neben einem matching-threshold auch ein blocking-threshold sowie ein blocking-mode definiert sind, wird die Variable sowohl für das Matching als auch für das Blocking verwendet (siehe Listing im Anhang B Zeile: 172-173). Wird hingegen nur ein matching-threshold angegeben und auf die Definition eines blocking-threshold und blocking-mode verzichtet, dient die Variable ausschließlich dem Matching. Mit dem ”blocking-mode” lässt sich definieren, ob die Blocking-Variable Zeichenketten oder numerische Werte enthält [2, S.66,67]. Für jede ausgewählte Matching- oder Blocking Variable lassen sich spezifische Parameter wie die Gewichtung, der Vergleichsalgorithmus und folglich auch Schwellenwerte festlegen [2, S.56,36]. Innerhalb des ”field” Elements wird der Name der jeweiligen Variable über das Unterelement ”Feldname” eingetragen [2, S.66]. Der konfigurierbare Schwellenwert reicht von 0.0 (0% Übereinstimmung) bis 1.0 (100% Übereinstimmung) und wird über das Element ”matching-threshold” festgelegt. Die Gewichtung einer Matching-Variable erfolgt über das Element ”weight” [2, S.67].

Der Abgleich der Ausprägungen einer Matching-Variable kann mit verschiedenen Algorithmen erfolgen, die im Element ”algorithm” angegeben werden können [2, S.67]. Der E-PIX unterstützt dabei die Kölner Phonetik, einen deterministischen Algorithmus, die Levenshtein-Distanz, den Sørensen-Dice-Koeffizient sowie den Jaccard-Koeffizient. Eine detaillierte Beschreibung der Algorithmen findet sich in Kapitel 2.1.4. Außerdem kann im E-PIX angegeben werden, dass eine Variable als ”Multiple-Value-Feld” behandelt werden soll. In solchen Fällen werden die Inhalte eines Feldes mithilfe eines Trennzeichens aufgespaltet und anschließend einzeln abgeglichen. Dies ist besonders nützlich, wenn erwartet wird, dass eine Variable wie ”Vorname” mehrere Vornamen enthält, die dann zwischen verschiedenen Personendatensätzen abgeglichen werden können [2, S.36,69]. Im Element ”multiple-values” können verschiedene Anpassungen vorgenommen werden, um diesen Prozess zu steuern (siehe Listing in Anhang B Zeile: 157-162). So kann beispielsweise das Trennzeichen zwischen den einzelnen Teilen einer Zeichen-

kette in einer Variablen über das Element ”separator” festgelegt werden. Wenn beispielsweise in der Variable ”Vorname” der Name ”Anna Lena” gespeichert ist, würde ein Leerzeichen als Trennzeichen verwendet werden. Zudem bietet das Element ”penalty-not-a-perfect-match” die Möglichkeit, einen Abzug vom Gesamtgewicht vorzunehmen, wenn die Teil-Zeichenketten in einem Multiple-Value-Feld zwar ähnlich sind, jedoch nicht exakt übereinstimmen, wie zum Beispiel bei den Namen ”Anna Lena” und ”Ana Lena”. Das Element ”penalty-one-short” zieht vom Gesamtgewicht eines Datenpaares einen konfigurierbaren Wert ab, wenn nicht alle Teilzeichenketten übereinstimmen, wie im Beispiel ”Anna Lena” und ”Anna”. Schließlich sorgt das Element ”penalty-both-short” dafür, dass das Gesamtgewicht verringert wird, wenn in beiden Feldern nicht alle Teile ähnlich sind, wie es bei ”Anna Lena” und ”Lena Müller” der Fall wäre [2, S.70,71].

Der E-PIX stellt eine Vielzahl weiterer Funktionen zur Verfügung, die über die Domänenkonfiguration hinausgehen und nach der Durchführung des Matching Vorgangs genutzt werden können (siehe Abbildung 3.4). Eine dieser Funktionen ist die Personensuche, mit der Personen anhand von MPIs, Identifikatoren oder projektspezifischen Zusatzfeldern abgerufen werden können [2, S.84]. Darüber hinaus können sowohl domänenspezifische als auch domänenübergreifende Statistiken eingesehen werden. Dazu gehören etwa die Anzahl der möglichen Übereinstimmungen, der registrierten Personen und der existierenden Identitäten [2, S.94]. In der Ergebnisübersicht werden detaillierte Informationen zu den gesuchten Personen angezeigt, was die Bearbeitung von Identitäten und die Verwaltung neuer Adressen ermöglicht [2, S.85,86]. Zusätzlich ist außerdem ein Protokoll verfügbar, das eine lückenlose Nachverfolgung von Ereignissen ermöglicht [2, S.93]. Des Weiteren ermöglicht der E-PIX den Import und Export von Personendaten im CSV-Format [2, S.91,92].

3.2. HAPI FHIR MDM

Im Folgenden wird die Record Linkage Lösung HAPI FHIR MDM vorgestellt. Analog zur Darstellung des E-PIX erfolgt eine Betrachtung der grundlegenden Merkmale, der Verwaltung von Identitäten, des Matching-Prozesses sowie der Konfigurationsmöglichkeiten.

3.2.1. Grundlegende Informationen

HAPI FHIR ist ein Produkt von Smile Digital Health, stellt eine vollständige Implementierung des HL7 FHIR-Standards dar und wird seit 2014 aktiv von einer offenen Community weiterentwickelt [14]. Die Java-basierte FHIR-Implementierung ist mit der Apache Software Lizenz 2.0 lizenziert, wobei die aktuelle stabile Version 8.0.0 am 17. Februar 2025 veröffentlicht wurde. HAPI FHIR erleichtert die Entwicklung von FHIR-konformen Anwendungen und APIs, indem es ein Framework für FHIR-Server und -Clients bietet und wird weltweit in einer Viel-

zahl von Ländern und Städten eingesetzt, darunter in Nordamerika, Südamerika, Europa, Asien, Australien und Neuseeland. Es existieren verschiedene Module, unter anderem eine Kernbibliothek, Client-Module, die die Kommunikation mit FHIR-Servern ermöglichen, Validierungsmodule, die FHIR-Ressourcen anhand von Profilen überprüfen und Server-Module, die es ermöglichen, FHIR-konforme Server mit verschiedenen Implementierungen wie einem Java Persistence API (JPA)-Server zu erstellen. Das MDM-Modul von HAPI FHIR dient speziell dazu, doppelte oder ähnliche Patientendaten zu erkennen und miteinander zu verknüpfen, um die Datenqualität und -integrität zu verbessern [3].

Das MDM-Modul ist ein Bestandteil des HAPI FHIR-Systems und erfordert keine umfassenden Anpassungen der bestehenden Infrastruktur. Dieses Modul nutzt die Standardfunktionen von HAPI FHIR, wie beispielsweise die RESTful API und arbeitet mit gängigen FHIR-Ressourcen wie ”Patient”, ”Observation” und weiteren. Das MDM-Modul ermöglicht das Erstellen und Verwalten von Verknüpfungen zwischen FHIR-Ressourcen, die darauf hinweisen, dass unterschiedliche Ressourcen entweder dieselbe tatsächliche Entität repräsentieren oder möglicherweise auf diese verweisen. Das Modul arbeitet mit dem JPA-Server, um FHIR-Daten in einer Datenbank zu verwalten und nutzt die Suchfunktionen von HAPI FHIR, um ähnliche Patienten zu identifizieren [3]. Wie in Tabelle 3.4 ersichtlich, verfügt HAPI FHIR MDM nicht über eine eigenständige Web-Oberfläche zur Verwaltung oder Konfiguration. Allerdings steht eine Swagger-UI als technisches Frontend zur Verfügung, die die Interaktion mit der zugrunde liegenden FHIR REST API ermöglicht. SOAP-basierte Schnittstellen werden von HAPI FHIR MDM nicht unterstützt, die Nutzung erfolgt ausschließlich über die REST-API.

Tabelle 3.4.: Funktionale Übersicht von HAPI FHIR MDM nach Zugriffsmethode (entnommen aus: eigene Aufnahmen)

Funktion	**SOAP***	**Swagger-UI**	**FHIR (REST)**
Personen anlegen	–	Es existiert	✓
Personen suchen	–	keine eigenständige	✓
Personen aktualisieren	–	Weboberfläche, lediglich	✓
Personen löschen	–	eine Swagger-UI	✓
Manuell ein Match hinzufügen	–	als technisches	✓
Manuelle Prüfung	–	Frontend zur	✓
Link-Historie anzeigen	–	Interaktion mit der	✓
Listenexport von Possible Matches	–	FHIR REST API	✓
Konfiguration der Matching-Regeln	–		✓

✓= Vollständig verfügbar, – = Nicht verfügbar

3.2.2. Verwaltung von Identitäten

Wenn die Daten eines neuen Patienten in eine FHIR-Ressource aufgenommen werden, erfolgt deren Speicherung in einer Datenbank. Im Anschluss daran prüft das MDM-System, ob der neu erstellte Datensatz mit bereits bestehenden Patientendaten übereinstimmt. Bei einer Übereinstimmung werden Verknüpfungen zwischen den entsprechenden Ressourcen erstellt und der Benutzer kann über die API unter Verwendung von FHIR-Operationen auf alle relevanten Datensätze zugreifen oder Korrekturen vornehmen. Ein Abgleich ist dabei ausschließlich zwischen Ressourcen des gleichen Typs möglich [3].

Wird innerhalb des Record Linkage Prozess festgestellt, dass mehrere Datensätze dieselbe Person repräsentieren, wird eine sogenannte ”Goldene Ressource” erzeugt. Bei der Goldenen Ressource wird davon ausgegangen, dass sie die korrekten Ausprägungen der Variablen enthält. Die als Duplikat identifizierte Ressource wird als ”Quellressource” bezeichnet. So könnten beispielsweise zwei Patienten mit identischen Ausprägungen in den Matching-Variablen als dieselbe Person angesehen und mit derselben Goldenen Ressource verknüpft werden. Wenn keine Übereinstimmung mit bestehenden Ressourcen vorliegt, wird eine neue Goldene Ressource erstellt [3].

Wenn das MDM-Modul in einem HAPI-FHIR-Server aktiviert ist, werden die Goldenen Ressourcen mit dem Tag ”happy-mdm” versehen und als schreibgeschützt behandelt. Diese Ressourcen werden ausschließlich durch das MDM-Modul verwaltet und lassen sich nur über spezielle MDM-Operationen ändern. Außerdem kann jede Quellressource höchstens mit einer Goldenen Ressource über einen ”MATCH-Link” verbunden sein. Ausnahmen bilden lediglich Datensätze, bei denen kein Abgleich der Daten stattfindet und die demnach mit ”NO-MDM” gekennzeichnet sind oder bei denen ein Possible-Match zur manuellen Überprüfung aussteht [3]. Enthält eine Quellressource eine Enterprise Identifier (EID), sucht das MDM-System nach einer bereits vorhandenen Goldenen Ressource mit derselben ID. Die EID dient als übergreifende Identifikationsnummer und ermöglicht die eindeutige Zuordnung einer Person über verschiedene Systeme hinweg, selbst wenn sie in unterschiedlichen Datenbanken oder Einrichtungen unter verschiedenen lokalen IDs erfasst wurde. Da EIDs nur selten vorhanden sind, erfolgt der Abgleich in den meisten Fällen über definierte Matching-Regeln, die verschiedene Matching-Variablen nutzen.

HAPI FHIR MDM nutzt verschiedene interne IDs zur Verwaltung und Verknüpfung von Ressourcen, so erhält jede Goldene Ressource und jede Quellressource eine eindeutige Identifikation, die bei einer Klassifizierung als Match entsprechend verknüpft werden. Die Zuordnung erfolgt über interne ”MDM-Link-IDs”, die nicht direkt als eigenständige FHIR-Ressourcen sichtbar sind. Jede Verbindung zwischen den Ressourcen wird mit einem spezifischen ”MatchResult-Status” versehen, der die Art der Beziehung definiert.

3.2.3. Matching-Prozess

Innerhalb des Matching-Prozess von HAPI FHIR MDM können den verglichene Datensatzpaaren vier verschiedene Matching Typen zugeordnet werden (siehe Tabelle 3.5).

Tabelle 3.5.: Matching-Typen mit zugehörigen Ereignissen und Handlungen in HAPI FHIR MDM auf Basis von [3]

Matching-Typ	**Ereignis**	**Handlung**
No-Match	Keine Übereinstimmung anhand der Regeln	Eine neue Goldene Ressource wird erstellt
Match	Die aufgestellten Regeln für eine Übereinstimmung treffen zu	Es wird eine MATCH-Verknüpfung zwischen der neuen Quellressource und der Goldenen Ressource erstellt
Possible-Match	Die Regeln für ein Possible-Match wurden erfüllt, zwei Quellressourcen zeigen eine teilweise Übereinstimmung	Manuelle Prüfung
Possible Duplicate	Zwei Goldene Ressourcen zeigen eine teilweise Übereinstimmung	Manuelle Prüfung

Der Status ”Possible-Match” kommt demnach zum Einsatz, wenn eine neue Quellressource mit einer Goldenen Ressource verglichen wird und eine unsichere Übereinstimmung vorliegt. ”Possible-Duplicate” hingegen tritt ausschließlich zwischen Goldenen Ressourcen auf, wenn das System vermutet, dass sie dieselbe Identität repräsentieren könnten. Falls eine Quellressource mit mehreren Goldenen Ressourcen übereinstimmt, werden diese als ”Possible-Duplicate” markiert, um eine spätere manuelle Überprüfung und mögliche Zusammenführung zu ermöglichen. Die in HAPI MDM verwalteten ”mdm-link-Datensätze”, die eine Quellressource mit einer Goldenen Ressource verbinden, werden, solange eine Verknüpfung durch vordefinierte Regeln erstellt oder aktualisiert wurde, als ”AUTO” gekennzeichnet. Sobald eine manuelle Änderung durchgeführt wurde, wird die Verknüpfung als ”MANUELL” markiert [3].

3.2.4. Konfigurationsoptionen

Die Verknüpfung von Datensätzen in HAPI FHIR MDM erfolgt anhand konfigurierbarer Matching-Regeln. Der Ablauf des Record Linkage lässt sich über verschiedene Aspekte anpassen (siehe Abbildung 3.6).

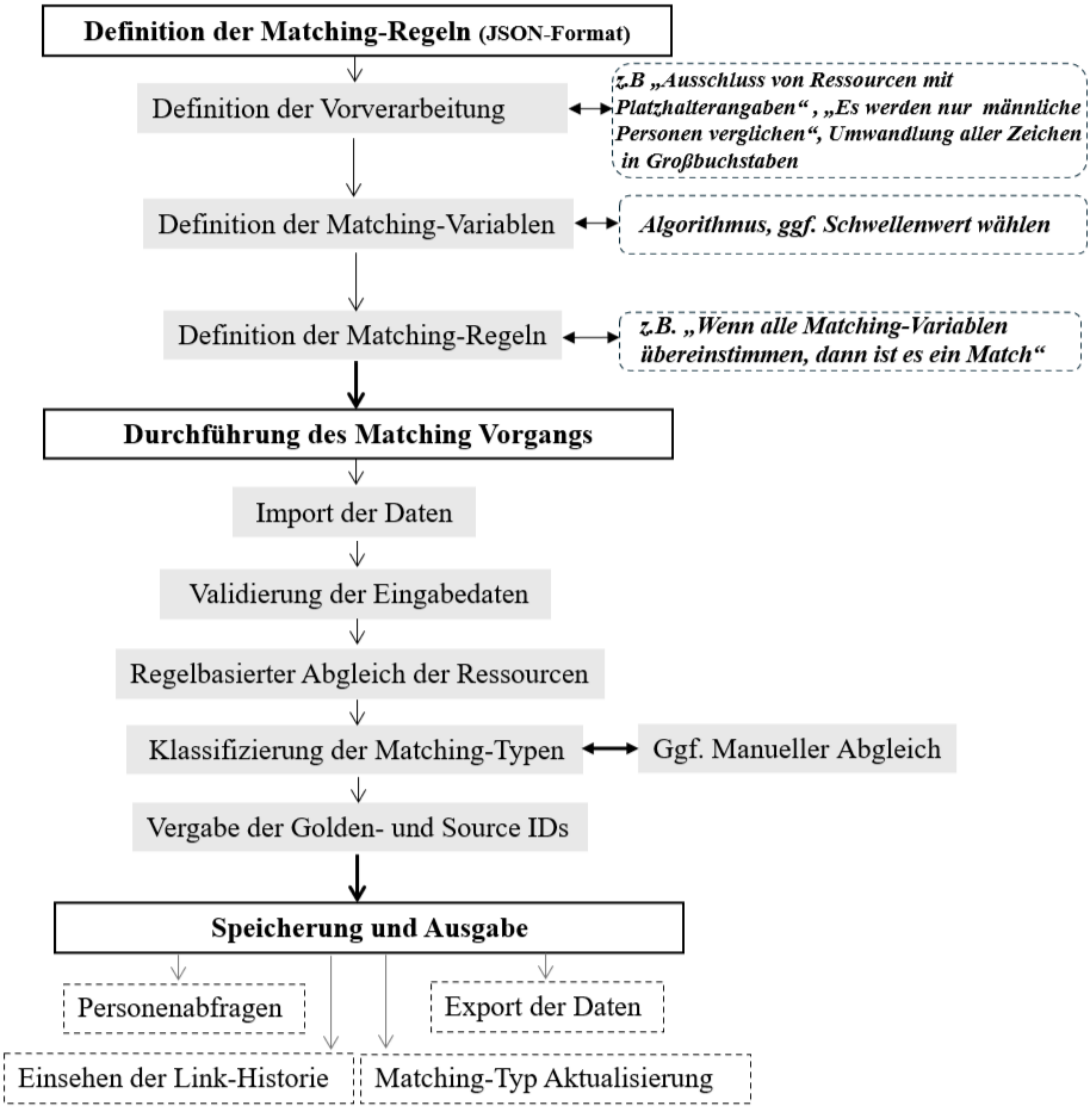


Abbildung 3.6.: Darstellung des Matching-Prozesses von HAPI FHIR MDM [3] (entnommen aus: eigene Aufnahmen)

Zunächst erfolgt gemäß Abbildung 3.6 die Konfiguration der Matching-Regeln in einer JSON-Datei, die beispielhaft im Anhang C abgebildet wird. Gemäß Abbildung 3.6 wird in einem ersten Schritt die Vorverarbeitung der Daten innerhalb der JSON-Datei konfiguriert. Standardmäßig werden bei allen Ausprägungen der Matching-Variablen diakritische Zeichen wie beispielsweise ”é”, ”ä” oder ”ç” entfernt und sämtliche Buchstaben vor dem Abgleich in Großbuchstaben umgewandelt. Falls eine exakte Übereinstimmung unter Berücksichtigung der ursprünglichen Groß- und Kleinschreibung sowie Akzentuierung erforderlich ist, kann die Eigenschaft ”exact=true” in den Matching-Regeln gesetzt werden. Ist diese Option deaktiviert (exact=false), werden Zeichenfolgen unabhängig von ihrer Formatierung verglichen [3].

In HAPI FHIR MDM besteht außerdem die Möglichkeit, bestimmte Ressourcen mithilfe von JSON-Regeln vollständig vom MDM-Abgleich auszuschließen. Das kann über die Elemente ”candidateSearchParams” und ”IBlockListRuleProvider” realisiert werden (siehe Listing in An-

hang C Zeile: 4-24). ”CandidateSearchParams” legt fest, welche Suchparameter zur Identifizierung potenzieller Übereinstimmungen verwendet werden, während ”IBlockListRuleProvider” spezifische Blocking-Regeln definiert, um sicherzustellen, dass nur geeignete Kandidaten in den Matching-Prozess aufgenommen werden. Die ”candidateSearchParams” definieren Matching-Variablen, die vorab exakt übereinstimmen müssen, damit zwei Ressourcen in den Matching-Prozess einbezogen werden. Dabei handelt es sich um eine vorgeschaltete ”Blocking”-Suche, die potenzielle Kandidaten identifiziert und so die aufwändige Berechnung von Match-Scores auf alle Ressourcen vermeidet. Innerhalb eines Blocks wirken mehrere Suchparameter als AND-Bedingungen, während verschiedene Blöcke in einer logischen OR-Verknüpfung zusammengeführt werden. Schwellenwerte oder Algorithmen für einen Abgleich können an dieser Stelle nicht definiert werden. Unter der Verwendung der Elemente ”BlockListRules” oder ”Interceptors” können Datensätze, die beispielsweise Platzhalternamen enthalten, von der Verknüpfung ausgeschlossen werden, da deren Berücksichtigung im Matching-Prozess überflüssig wäre. Eine exemplarische Konfiguration zeigt Listing 3.1. In diesem Beispiel werden Patienten mit den Namen ”John Doe” und ”Jane Doe” anhand ihrer Namensfelder vom Matching ausgeschlossen. Hierfür wird über sogenannte `fhirPath`-Angaben auf spezifische Felder innerhalb der Ressource zugegriffen, deren Werte mit den definierten Kriterien abgeglichen werden [3].

```
{
   "blocklist": [{
      "resourceType": "Patient",
      "fields": [{
        "fhirPath": "name.first().family",
        "value": "doe"
      }, {
         "fhirPath": "name.first().given.first()",
         "value": "john"
      }]
   }, {
      "resourceType": "Patient",
      "fields": [{
         "fhirPath": "name.first().family",
         "value": "doe"
      }, {
         "fhirPath": "name.first().given.first()",
         "value": "jane"
      }]
   }]
}
```

Listing 3.1: Beispiel Konfiguration für das Blocking in HAPI FHIR MDM [3]

Von der ausgeschlossenen Ressource kann zwar eine Goldene Ressource erstellt werden, jedoch erfolgt nach einem Ausschluss einer Ressource kein Abgleich mit vorhandenen Datensätzen. Für einen MDM-Abgleich, der auf vordefinierten Blocking-Regeln basiert, ist die Implementie-

rung eines ”IBlockListRuleProvider” erforderlich. Dieser enthält eine Liste spezifischer Regeln, die für unterschiedliche Ressourcentypen gelten und die relevanten Felder für die Blockierung definieren. Dabei wird keine Unterscheidung zwischen Groß- und Kleinschreibung gemacht [3]. Im Rahmen von ”candidateSearchParams” wird in der FHIR-Datenbank nach potenziellen Ressourcen gesucht, indem verschiedene Suchparameter wie ”Geburtsdatum” oder ”Telefonnummer” herangezogen werden. Wenn mehrere ”searchParams” in einem Suchblock definiert sind, werden diese als separate Suchkriterien interpretiert. Der HAPI FHIR-Server führt die Suchen in diesem Fall unabhängig voneinander durch und kombiniert die Ergebnisse der einzelnen Suchkriterien anschließend, um potenzielle Übereinstimmungen zu ermitteln. Fehlt jedoch ein spezifischer Wert innerhalb des eingehenden Datensatzes, wird die entsprechende Suche ignoriert und es wird lediglich anhand der verbleibenden Parameter nach Übereinstimmungen gesucht [3]. Über eine weitere Konfigurationsoption lässt sich die Menge der zu vergleichenden Datensätze gezielt einschränken. Unter Verwendung des Elements ”candidateFilterSearchParams” können die Ressourcen gefiltert werden, sodass nur diejenigen in den Abgleich einbezogen werden, die bestimmte Kriterien erfüllen. So ist es beispielsweise möglich, ausschließlich aktive Datensätze zu berücksichtigen, indem der Filter auf ”active = true” gesetzt wird [3].

Nach der Anpassung der Einstellungen für die Vorverarbeitung erfolgt die Konfiguration der Matching-Variablen (siehe Abbildung 3.6. Ein zentraler Bestandteil der Konfiguration in der JSON-Datei ist das Element ”matchFields”. In diesem Element werden die Variablen für den Abgleich definiert (siehe Listing im Anhang C Zeile: 25-84). Dabei kann für jede Variable der Name, der Ressourcentyp, der Pfad zur Ressource, der verwendete Algorithmus sowie ein optionaler Schwellenwert angegeben werden. Der Name innerhalb des Elements ”matchFields” bezeichnet die zu vergleichende Variable, beispielsweise ”birthday” für das Geburtsdatum. Der ”resourceType” gibt an, auf welchen Ressourcentyp sich das Feld bezieht, etwa ”Patient” für Patientendaten. Über den ”resourcePath” wird der Speicherort des Attributs innerhalb der Ressource bestimmt. Der angegebene Algorithmus legt fest, wie die Werte miteinander verglichen werden. Entweder werden die Ausprägungen einer Matching Variable durch eine binäre Entscheidung (Übereinstimmung oder keine Übereinstimmung) oder durch einen numerischen Ähnlichkeitswert zwischen 0.0 (keine Übereinstimmung) und 1.0 (exakte Übereinstimmung) verglichen. Falls der Algorithmus mit Ähnlichkeitswerten arbeitet, kann zusätzlich ein Schwellenwert für die jeweilige Matching-Variable definiert werden. Das MDM-Modul ermöglicht eine Anwendung mehrerer unterschiedlicher Algorithmen auf dieselbe Matching-Variable (siehe Listing im Anhang C Zeile: 59-72). FHIR HAPI MDM unterstützt für den Abgleich verschiedene Algorithmen (siehe Tabelle 3.6) [3].

Tabelle 3.6.: Unterstützte Algorithmen im HAPI FHIR MDM Record Linkage Modul auf Basis von [3]

Kategorie	**Algorithmus**	**Erläuterung**
Phonetische Kodierung	Caverphone 1 Caverphone 2 Kölner Phonetik Double Metaphone Match Rating Approach Metaphone Nysiis Soundex Verfeinerter Soundex	Vergleich von ähnlich klingenden Wörtern durch phonetische Kodierung
Zeichenketten Algorithmen	String-Matching Teilzeichenfolge Jaro-Winkler Levenshtein Kosinus Jaccard Sørensen-Dice-Koeffizient	Analyse und Vergleich von Zeichenfolgen
Numerisch	Numerisch Jaro-Winkler Numerisch Levenshtein Numerisch Kosinus Numerisch Jaccard Numerisch Sørensen-Dice	Entfernen aller nicht-numerischen Zeichen, vor Anwendung des Algorithmus
Feldspezifisch	Vor- und Nachname Spitzname Namen beliebiger Reihenfolge Erweiterungen beliebiger Reihenfolge Identifikator Leeres Feld	Abgleich in fester Reihenfolge True, wenn ein Name ein Spitzname des anderen ist Vor- und Nachname in beliebiger Reihenfolge vergleichen Vergleicht Erweiterungen unabhängig von der Reihenfolge, Übereinstimmend bei gleicher URL und gleichem Wert Übereinstimmungen, wenn das System und der Identifikator identisch sind. Entspricht einem leeren Feld

	Datum	Reduziert die Genauigkeit der Datumsangaben auf die geringere der beiden
	Numerisch	Entfernt alle nicht-numerischen Zeichen

Bei Variablen, die mehrere Werte enthalten (z.B. mehrere Vornamen), kann es sinnvoll sein, den Vergleich der einzelnen Werte gezielt zu steuern. Der Abgleich solcher Multiple-Value-Felder kann über die Verwendung des ”fhirPath-Elements” innerhalb der ”matchFields-Konfiguration” gesteuert werden. Mithilfe von ”FHIRPath-Ausdrücken” lässt sich das Verhalten anpassen, sodass eine direkte, positionsabhängige Kontrolle möglich ist. Ein Beispiel für die gezielte Steuerung des Abgleichs bei Feldern mit mehreren Werten liefert Listing 3.2. In dieser Konfiguration wird mithilfe des `fhirPath`-Ausdrucks ”name.given[0]” festgelegt, dass ausschließlich der erste Vorname eines Patienten beim Vergleich berücksichtigt wird. Dadurch wird verhindert, dass unterschiedliche Reihenfolgen der Vornamen, etwa ”Anna Lena” und ”Lena Anna”, fälschlicherweise als identisch erkannt werden, da es sich um zwei verschiedene Personen handeln kann. Als Matching-Algorithmus kommt in dem folgenden Beispiel Metaphone zum Einsatz, der phonetische Ähnlichkeiten zwischen Zeichenketten erkennt.

```
{
    "name": "firstname-meta-fhirpath",
    "resourceType": "Patient",
    "fhirPath": "name.given[0]",
    "matcher": {
        "algorithm": "METAPHONE"
    }
}
```

Listing 3.2: Beispiel Konfiguration für den Vergleich eines Multiple-Value-Feldes in HAPI FHIR MDM [3]

Die Konfiguration der Matching-Regeln bildet den letzten Schritt der Konfigurationsmöglichkeiten innerhalb der JSON-Datei, denn HAPI FHIR MDM basiert auf einem deterministischen Record Linkage Verfahren. Mithilfe des Elements ”matchResultMap” lassen sich diese Regeln definieren, so wird präzise festgelegt, unter welchen Bedingungen ein Datensatzpaar als Übereinstimmung oder als potenzielle Übereinstimmung klassifiziert wird. Ein Beispiel für die Matching-Regeln ist in der Beispiel-Konfiguration in Anhang C in Zeile 85 bis 93 abgebildet. Die erste Regel des JSON-Codes besagt beispielsweise, dass wenn der Vorname, der Nachname und das Geburtsdatum in beiden Datensätzen exakt übereinstimmen, das Datensatzpaar als ”MATCH” klassifiziert wird [3].

Externe EID-Systeme können für jede eingehende Ressource in dem Element ”eidSystems” individuell festgelegt werden. Es besteht auch die Möglichkeit, das Platzhalterzeichen ”*” zu verwenden, um eine EID für alle Ressourcentypen zu definieren.

Nachdem die JSON-Datei gemäß Abbildung 3.6 konfiguriert wurde, kann der Record Linkage Prozess durchgeführt werden. Bevor eine Ressource durch das MDM-Modul verarbeitet wird, prüft HAPI FHIR MDM zunächst, ob sie mindestens ein relevantes Attribut enthält, das in den Matching-Regeln definiert ist. Falls die Ressource nicht mindestens ein relevantes Attribut aufweist, erfolgt keine MDM-Verarbeitung. Wird die Quellressource jedoch später aktualisiert und erhält dadurch Attribute, die für das Matching relevant sind, wird sie zu diesem Zeitpunkt in den Abgleich mit einbezogen. Nachdem der Abgleich der Ressourcen und die Vergabe der Golden- und Source IDs erfolgt ist, bietet HAPI FHIR MDM eine Reihe von Funktionen zur Verwaltung der MDM-Verknüpfungen, die über den ”MdmProvider” zugänglich sind (siehe Abbildung 3.6). Diese Funktionen ermöglichen das Erstellen, Bearbeiten und Abfragen von MDM-Links. Um mit großen Datenmengen effizient umzugehen, nutzt HAPI FHIR MDM die sogenannte Paginierung, die es erlaubt, die Ergebnisse in kleinere und handhabbare Seiten zu unterteilen. Die Paginierung kann über die Parameter ”_offset” und ”_count” gesteuert werden.

Die Abfrage von MDM-Links kann mit verschiedenen optionalen Parametern wie ”goldenResourceId”, ”resourceId” und ”matchResult” erfolgen. Diese Parameter bieten eine Steuerung darüber, welche Links abgerufen werden. In der Antwort sind die relevanten MDM-Links enthalten, zusammen mit Informationen zu Übereinstimmungsstatus, Quelle der Ressource und Goldener Ressource. Zudem können die Ergebnisse nach verschiedenen Kriterien wie dem Erstellungsdatum sortiert werden. Eine weitere Funktion von HAPI FHIR MDM ist die ”Link-Historie”, die es ermöglicht, historische Einträge für Ressourcen oder deren Verknüpfungen abzurufen. Die Ergebnisse werden nach der goldenen Ressourcen-ID und dem Revisionszeitstempel sortiert, wobei nur ein Eintrag pro Kombination von goldener und Quellressourcen-ID für Duplikate zurückgegeben wird. Die Abfrage doppelter Goldener Ressourcen ermöglicht das Abrufen einer Liste von Duplikaten. Zudem gibt es die Funktion, doppelte Goldene Ressourcen zu ”entduplicieren”, indem die betroffenen Ressourcen als keine Duplikate markiert werden. Mit der Funktion Links zu aktualisieren, können MDM-Verknüpfungen zwischen Goldenen Ressourcen und Zielressourcen nachträglich manuell angepasst werden. Schließlich ermöglicht die Funktion ”Goldene Ressourcen zusammenführen” das Zusammenführen von zwei Goldenen Ressourcen, wobei Daten aus der Quellressource in die Zielressource übertragen und die Quelle als ”REDIRECTED” markiert wird.

Neben den MDM-Operationen bietet HAPI FHIR MDM auch die Möglichkeit, Abfragen von den Ressourcen durchzuführen. Diese Abfragefunktionen erlauben es, Ressourcen anhand festgelegter Übereinstimmungsregeln zu suchen. Der ”$match-Vorgang” ist speziell dafür zuständig, Ressourcen zu finden, die den angegebenen Kriterien entsprechen. Durch diese Abfrage wird der entsprechende Übereinstimmungsgrad zurück gegeben. Ein weiterer nützlicher Vorgang ist

der ”$mdm-clear-Vorgang”, der es ermöglicht, MDM-Verknüpfungen sowie die damit verbundenen Ressourcen in einem Batch zu löschen. Dies ist insbesondere hilfreich, wenn Anpassungen an den Regeln vorgenommen werden müssen. Schließlich gibt es noch den ”$mdm-submit-Vorgang”, mit dem Ressourcen zur MDM-Verarbeitung übermittelt werden. Dieser Vorgang erfolgt asynchron und der Benutzer erhält eine Rückmeldung bezüglich des Verarbeitungsstatus der gesendeten Ressourcen.

4. Methodik

Im Folgenden wird das methodische Vorgehen dieser Arbeit erläutert. Die Methodik gliedert sich in eine theoretische Analyse sowie einer daran anschließenden experimentellen Analyse. Beide Teile dienen dem Ziel, den E-PIX und FHIR HAPI MDM im Hinblick auf ihre Konzepte und ihre Verknüpfungsqualität miteinander zu vergleichen. Dafür wurden die beiden Record Linkage Lösungen zunächst anhand ihrer technischen Merkmale und Funktionen analysiert. Grundlage hierfür bildete eine Durchsicht der verfügbaren Dokumentationen. Darauf aufbauend wurde ein experimenteller Ansatz gewählt, um die Verknüpfungsqualität der Lösungen zu evaluieren (siehe Abbildung 4.1). Hierfür dient ein Echtdatensatz aus dem Krebsregister Mecklenburg-Vorpommern sowie erstelle Variation von Konfigurationszenarien die in Ahnhang D und E zu finden sind.

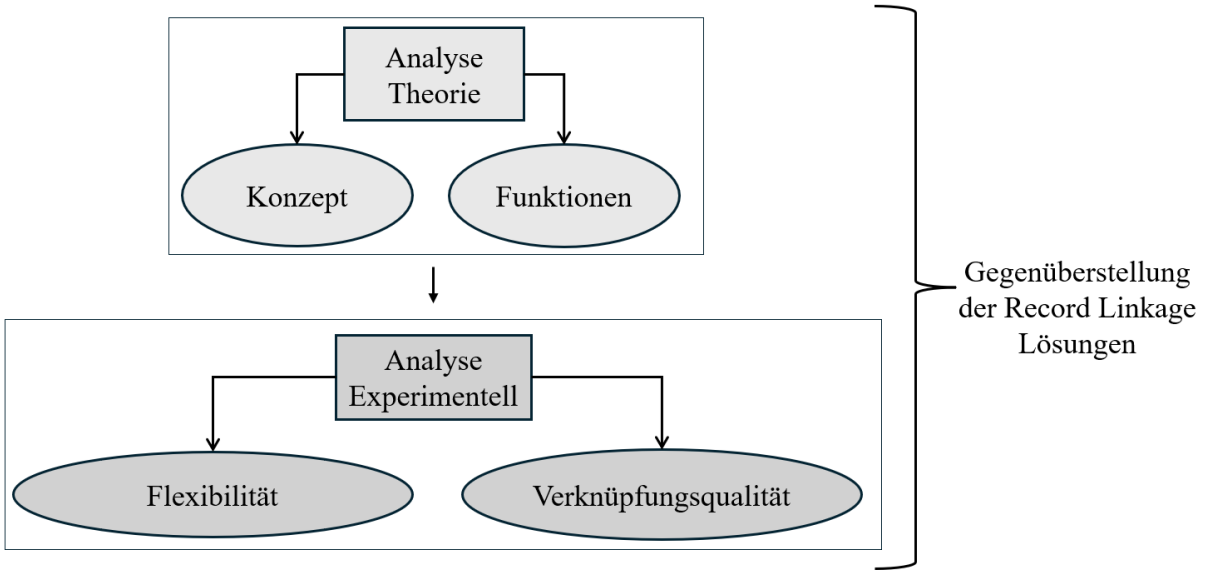


Abbildung 4.1.: Überblick über das methodische Vorgehen in dieser Arbeit (entnommen aus: eigene Aufnahmen)

4.1. Literaturrecherche

Die Literaturrecherche umfasste sowohl die Dokumentationen der Record Linkage Lösungen als auch theoretische Aspekte des Record Linkage und des HL7 FHIR Standards. Die Recherche erfolgte unter Nutzung verschiedener wissenschaftlicher Datenbanken, darunter ”PubMed”, ”SpringerLink” und ”IEEEXplore”. Ergänzend wurde eine gezielte Suche auf den Webseiten der Record Linkage Lösungen sowie auf Plattformen zum HL7 FHIR Standard durchgeführt und Anwenderhandbücher genutzt, um relevante Informationen zu den Konzepten der Record

Linkage Lösungen zu erhalten. Die Recherche erstreckte sich über den Zeitraum von Anfang März 2025 bis Anfang April 2025 und konzentrierte sich auf deutsch- und englischsprachige Literatur, insbesondere auf wissenschaftliche Veröffentlichungen aus den Jahren 2000 bis 2025. Darüber hinaus wurden auch grundlegende Werke berücksichtigt, die außerhalb dieses Zeitrahmens liegen, da sie in der Fachliteratur als theoretische Basis genutzt werden und einen entscheidenden Beitrag zum Verständnis des Record Linkage leisten [32,34–36,47,49]. Zu Beginn der Recherche wurde in den genannten Datenbanken mit spezifischen Suchstrings gearbeitet, wobei zunächst eine Vorauswahl anhand der Abstracts getroffen wurde. Die ausgewählte Literatur wurde daraufhin in der Literaturdatenbank ”JabRef” unter Verwendung der entsprechenden DOIs abgelegt. In einem zweiten Schritt erfolgte eine detailliertere Sichtung der Literatur, bei der thematisch nicht passende Arbeiten aussortiert wurden.

Tabelle 4.1.: Darstellung der Suchstrings, Rechercheziele und Trefferzahlen

Suchstring	**Suchziel**	**IEEEXplore**	**SpringerLink**	**PubMed**
Record Linkage AND Methods	Grundlegende Verfahren des Record Linkage	435	42.305	22
Record Linkage AND Probabilistic	Eingrenzung auf probabilistische Methoden	–	4.148	–
Record Linkage AND Deterministic	Eingrenzung auf deterministische Methoden	–	2.714	–
Record Linkage AND Bias	Untersuchung von Fehlerquellen und Herausforderungen	15	15.278	6
Record Linkage AND homonym error	Eingrenzung Literatur zu Homonymfehlern	–	46	–
Record Linkage AND synonym error	Eingrenzung Literatur zu Synonymfehlern	–	380	–
Record Linkage AND similarity measures	Algorithmen und Ähnlichkeitsmaße im Record Linkage	56	5.476	3
Record Linkage AND Phonetic	Eingrenzung Recherche zu phonetischen Verfahren	–	479	–
Record Linkage AND Edit Distance	Eingrenzung Recherche zu Edit-Distance-Algorithmen	–	1.083	–

Um FHIR in den Kontext weiterer Interoperabilitätsstandards einzuordnen und eine breitere Perspektive zu erhalten, wurde zusätzlich eine gezielte Suche nach Büchern durchgeführt, die verschiedene Interoperabilitätsstandards behandeln. Hierfür wurde der Suchstring ”FHIR AND Interoperability standards” verwendet und die Suche auf Bücher aus dem Zeitraum von 2000 bis 2025 sowie auf deutsch- und englischsprachige Literatur eingeschränkt. In IEEEXplore konnte mit diesem Suchstring keine relevanten Treffer erzielt werden, jedoch konnten in SpringerLink drei passende Bücher identifiziert werden, von denen eines in die Recherche einbezogen wurde. In PubMed ergab die Suche ebenfalls drei Bücher, die jedoch nicht verwendet wurden. Zusätzlich wurden die Webseiten der beschriebenen Standards herangezogen, um weiterführende Informationen zu sammeln.

4.2. Experimentelle Analyse

In diesem Kapitel wird die Analyse der Verknüpfungsqualität der beiden Record Linkage Lösungen beschrieben.

Zunächst wurde für jede Lösung eine eigene Testumgebung entwickelt, die die Durchführung mehrerer Konfigurationsszenarien ermöglichte. Diese Konfigurationen unterschieden sich unter anderem hinsichtlich der verwendeten Matching-Variablen, eingesetzten Algorithmen, Schwellenwerte sowie Gewichtungen (vgl. Anhang D und E). Durch die Variation der Konfigurationen entstanden mehrere Testdurchläufe, in denen die jeweiligen Konfigurationen in die Testumgebung integriert und auf den Testdatensatz angewendet wurden (siehe Abbildung 4.2). Nach jedem Durchlauf wurden die Verknüpfungsergebnisse mit dem Referenzdatensatz, bei dem die korrekten Verknüpfungen bereits bekannt waren, automatisiert abgeglichen. Auf diese Weise ließ sich analysieren, in welchem Umfang fehlerhafte oder fehlende Verknüpfungen auftraten.

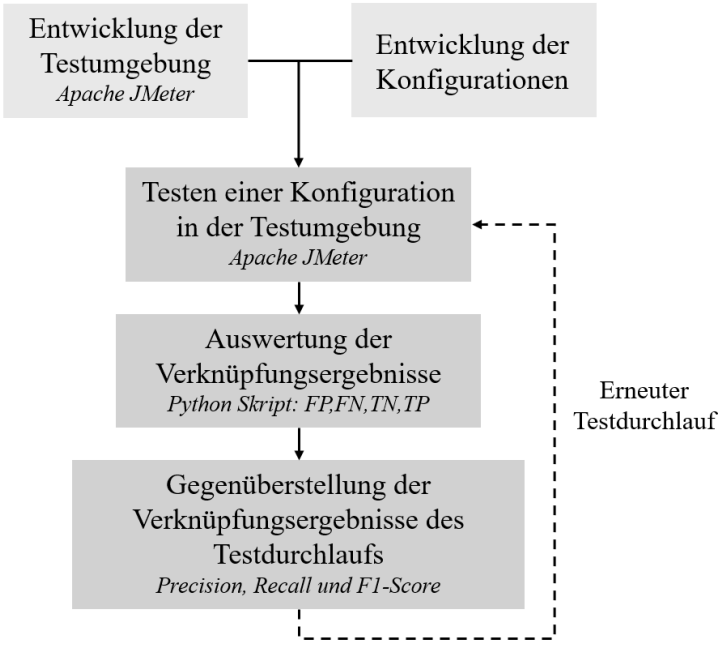


Abbildung 4.2.: Ablauf der experimentellen Analyse, Verwendete Abkürzungen: TP – True Positives, FP – False Positives, TN – True Negatives, FN – False Negatives (entnommen aus: eigene Aufnahmen)

4.2.1. Datensatz

Als Testdatensatz dienen insgesamt 2.729.204 Patientendatensätze in Form von FHIR-Patientressourcen aus dem Krebsregister Mecklenburg-Vorpommern. Die Eignung dieses Datensatzes als Referenz ergibt sich aus der Tatsache, dass ein manuelles Review im Vorfeld durchgeführt wurde. Insgesamt wurden 40.762 potenzielle Dubletten überprüft und aufgelöst. Davon wurden 21.057 Datensätze manuell als identisch erkannt und somit zusammengeführt. Weitere 8.187 Zusammenführungen erfolgten automatisch im Rahmen einer sogenannter indirekter Zusammenführungen. Diese treten auf, wenn ein Datensatz mit mehreren Datensätzen übereinstimmt und durch eine bereits erfolgte direkte Zusammenführung zusätzliche Verknüpfungen logisch ableitbar werden. Sobald eine Verbindung zwischen zwei Datensätzen festgestellt wird, werden alle weiteren, die mit einem der beiden verknüpft sind, automatisch gemeinsam zusammengeführt. Neben den zusammengeführten Datensätzen wurden in 11.518 Fällen potenzielle Dubletten getrennt, da diese als unterschiedliche Personen identifiziert wurden. Daher enthält der Datensatz keine offenen möglichen Duplikate und es befinden sich keine Datensätze in einem laufenden Dublettenprüfprozess. Es wird davon ausgegangen, dass die vorliegenden Verknüpfungen innerhalb des Testdatensatzes korrekt und vollständig sind. Somit eignet sich dieser Datensatz als verlässliche Referenz zur Validierung der Ergebnisse der eingesetzten Record Linkage Lösungen.

Der Datensatz enthält unter anderem die Pflicht-Variablen ”Vorname”, ”Nachname”, ”Geschlecht”, ”Geburtsdatum” sowie die ”Postleitzahl” des Hauptwohnsitzes. Werden ausschließlich diese Pflichtvariablen herangezogen, lassen sich innerhalb des Datensatzes 2.624.525 tatsächliche Übereinstimmungen (True Positives) sowie 104.679 tatsächliche Nicht-Übereinstimmungen (True Negatives) identifizieren. Darüber hinaus enthält der Datensatz weitere optionale Variablen, die zur Verfeinerung des Linkage Prozesses herangezogen werden können. Hierzu zählt die ”Krankenversichertennummer”. Die Krankenversichertennummer wurde im Vorfeld aufbereitet, sodass ausschließlich formal gültige Einträge verwendet wurden. Da diese optionalen Variablen nicht in allen vorliegenden Datensätzen vorhanden sind, reduziert sich die für Analysen nutzbare Fallzahl abhängig von den gewählten Matching-Variablen. Eine Übersicht zu den resultierenden Fallzahlen sowie der Verteilung von True Positives und True Negatives in Abhängigkeit von den verwendeten Matching-Variablen findet sich in Tabelle 4.2.

Tabelle 4.2.: Darstellung der Fallzahlen sowie Verteilung von True Positives und True Negatives in Abhängigkeit von den verwendeten Attributen

Matching-Variablen	**Anzahl der Testdatensätze**	**True Positives & True Negatives**
Pflichtvariablen	2.729.204	True Positives: 2.624.525 True Negatives: 104.679
Pflichtvariablen + Krankenversichertennummer	257.393	True Positives: 131.191 True Negatives: 126.202

Außerdem wurde jedem Datensatz eine eindeutige ”identity_person_id” zugeordnet. Diese Kennung markiert Datensätze, die eine tatsächliche Übereinstimmung im Testdatensatz darstellen, indem tatsächlichen Übereinstimmungen die identische ”identity_person_id” tragen. Zusätzlich besitzt jeder Datensatz eine individuelle ”uuid”, wodurch eine eindeutige Identifikation jedes Datensatzes möglich ist.

4.2.2. Testumgebung

Um einen systematischen Vergleich der beiden Record Linkage Lösungen zu ermöglichen, wurde zunächst, wie in Abbildung 4.2 dargestellt, jeweils eine Testumgebung für E-PIX und HAPI FHIR MDM eingerichtet. Zur Durchführung und Automatisierung der Tests wurde das Open-Source-Tool Apache JMeter genutzt, das sich für Last-, Performance- und Funktionstests von Webanwendungen und APIs eignet [64]. Die JMeter-Tests unterscheiden sich teilweise aufgrund der unterschiedlichen Architekturen der Lösungen, folgen jedoch in ihren grundlegenden Abläufen einem identischen Prinzip (siehe Abbildung 4.3)

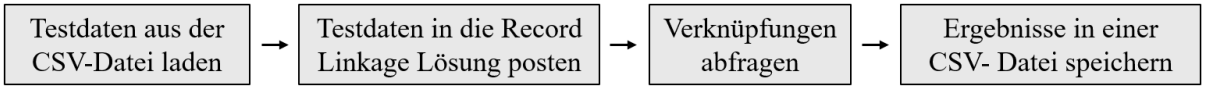


Abbildung 4.3.: Darstellung der grundlegenden Schritte innerhalb der JMeter-Tests beider Record Linkage Lösungen (entnommen aus: eigene Aufnahmen)

Für die Testumgebung des E-PIX wurde die aktuelle Version (2024.3.1, März 2025) über Docker Compose verwendet. JMeter wurde in einen eigenen Docker-Container eingebunden und in die bestehende ”docker-compose.yml-Konfiguration” des E-PIX integriert. Dadurch wurde beim Starten der Umgebung über Docker Compose der JMeter-Container automatisch mitgestartet. Innerhalb des JMeter Tests des E-PIX wurde zunächst die SOAP-Schnittstelle verwendet, um eine neue Domäne anzulegen. Dazu wurde ein ”HTTP-POST-Request” im SOAP-Format an den entsprechenden Webservice gesendet. Der POST-Requeste übermittelte grundlegende Parameter wie den Namen und die Beschreibung der Domäne, deren Zuordnung zu einer MPI-Domäne sowie die Konfiguration der verwendeten Sicherheitsquelle.

Im Anschluss wurden die patientenidentifizierenden Daten aus dem Testdatensatz an den E-PIX übermittelt. Die Daten wurden aus einer externen CSV-Datei mithilfe des Elements ”CSV Data Set Config” in JMeter geladen und als Variablen verfügbar gemacht. In einem weiteren Schritt wurden die Patientendaten über einen ”HTTP-Request” an das TTP FHIR Gateway und schließlich an den E-PIX übermittelt. Grundlage hierfür war die in Kapitel 3.1.1 beschriebene FHIR-Schnittstelle. Die Patientendaten wurden dabei in Form einer XML-Vorlage übermittelt, die zur Laufzeit dynamisch mit den entsprechenden Variablen aus der CSV-Datei mit den Testdaten befüllt wurde. Ergänzend wurden Metadaten wie die Ziel-Domäne, die verwendete Datenquelle sowie die gewünschte Speicheroperation übertragen. Für den Abgleich der Daten griff E-PIX auf die zuvor erstellte Matching-Konfigurationsdatei im XML-Format zurück. Diese wurde automatisch geladen und zur Klassifizierung der Datensätze verwendet.

Nach Absenden der Patientendaten extrahierte ein ”XPath Extractor” aus der Serverantwort die vom E-PIX generierten MPI-IDs. Diese wurden zusammen mit der ursprünglichen ”identity_person_id” und der ”uuid” aus der CSV-Datei der Testdaten mithilfe eines ”BeanShell-Skripts” in eine neue CSV-Datei geschrieben. Anhand dieser IDs ließ sich die Verknüpfungsqualität bewerten. Da Datensätze, die in dem Testdatensatz als tatsächliche Übereinstimmung klassifiziert wurden, dieselbe ”dentity_person_id” tragen, kann überprüft werden, ob der E-PIX denselben Datensätzen auch dieselbe MPI-ID zugeordnet hat. Die ”uuid” ermöglicht eine Nachverfolgung fehlerhaft klassifizierter Einträge. Dadurch lassen sich Ursachen für fehlerhafte Verknüpfungen nachvollziehen.

Für die Durchführung der Tests mit Hapi FHIR MDM wurde das JPA Server Starter-Projekt von HAPI FHIR eingesetzt und über Docker Compose ausgeführt. Im Rahmen der Konfiguration wurde das MDM-Modul aktiviert, um eine automatisierte Verlinkung und Verwaltung von Patientenidentitäten zu ermöglichen. Die erforderlichen Einstellungen wurden über pro-

jektspezifische Property-Dateien und Modulanpassungen vorgenommen. Um die Levenshtein-Distanz als Algorithmus im HAPI FHIR MDM-Modul nutzen zu können, musste diese zunächst manuell integriert werden. HAPI FHIR verweist auf ihrer Website in diesem Zusammenhang auf die Bibliothek unter [65] hin. Die empfohlene Logik zur Berechnung der Levenshtein-Distanz wurde übernommen, indem die Java-Bibliothek ”java-string-similarity” als Maven-Abhängigkeit in die ”pom.xml” des Projekts aufgenommen. Darauf aufbauend wurde eine eigene Java-Klasse implementiert, die die Logik zur Berechnung der Levenshtein-Distanz enthält und als benutzerdefinierte Matching-Regel dient. Diese neue Regel wurde anschließend in die zentrale MDM-Konfiguration integriert, indem in der Klasse ”MdmConfig” eine entsprechende ”Bean-Definition” hinterlegt wurde. Auf diese Weise konnte die Levenshtein-Distanz in den Matching-Konfigurationen des FHIR-Servers als eigenständige Vergleichslogik genutzt werden.

Auch für HAPI FHIR MDM wurde eine automatisierte Testumgebung auf Basis von Apache JMeter realisiert. Der Test beginnt ebenfalls mit dem Laden der Testdaten aus einer externen CSV-Datei über das Element ”CSV Data Set Config”, wodurch die Daten als Variablen im Test verfügbar gemacht werden. Die Patient Ressourcen werden anschließend über einen ”HTTP POST-Request” an den lokalen FHIR-Server gesendet. Die Konfigurationsregeln für das Matching, die als JSON-Datei vorliegen, wurden in einem Ressourcenverzeichnis des Servers abgelegt. Sie werden automatisch geladen und beim Eingang neuer Ressourcen für den Abgleich verwendet. Für die Erprobung unterschiedlicher Konfigurationen wurde diese Datei zwischen den einzelnen Testdurchläufen jeweils angepasst. Im Anschluss an die Übermittlung der Patientendaten erfolgt ein ”HTTP GET-Request”, der die FHIR-spezifische Operation ”$mdm-query-links” verwendet. Damit werden die vom MDM-Modul erkannten und gespeicherten Verlinkungen zwischen Patientendatensätzen abgefragt. Mithilfe eines ”JSR223 PostProcessors” wurde in einem nächsten Schritt ein Groovy-Skript implementiert, das über einen zusätzlichen Server-Request die vollständige JSON-Antwort der FHIR-Ressourcen abruft. Aus dieser Antwort wurden gezielt die ”identity_person_id”, die ”uuid” sowie die vom MDM-Modul vergebene ”goldenID” extrahiert und in eine neue CSV-Datei geschrieben. Die ”goldenID” erfüllt bei HAPI FHIR MDM eine vergleichbare Funktion wie die MPI-ID im E-PIX. Durch einen Abgleich mit der vorab definierten ”identity_person_id” lässt sich prüfen, ob die Verknüpfung korrekt erfolgte.

Nach Abschluss der JMeter-Tests liegen somit zwei CSV-Dateien vor, die jeweils die Variablen ”identity_person_id”, ”uuid” sowie, abhängig von der eingesetzten Lösung, entweder die ”MPI-ID” (beim E-PIX) oder die ”goldenID” (bei HAPI FHIR MDM) enthalten. Zur Ermittlung der True Positives, True Negatives, False Positives und False Negatives wurde ein Python-Skript entwickelt, das nach Abschluss der Tests in Visual Studio Code ausgeführt wurde (siehe Abbildung 4.2). Ziel dieses Skripts ist es, die vergebenen Identifikatoren (MPI-ID bzw. goldenID) mit der zuvor definierten ”identity_person_id” abzugleichen, um die Qualität der Zuordnungen systematisch zu bewerten. Die Logik basiert auf der Annahme, dass Datensätze, die im Vorfeld

mit derselben ”identity_person_id” versehen wurden, tatsächlich dieselbe Person repräsentieren. Stimmen in diesen Fällen auch die durch die Record Linkage Lösung vergebenen Identifikatoren überein, so handelt es sich um korrekte Verknüpfungen. Weichen sie hingegen voneinander ab, wird dies als Fehlklassifikation gewertet. Dem Abgleich zufolge können demnach vier verschiedene Ereignisse entstehen (siehe Tabelle 4.3).

Tabelle 4.3.: Ereignisse des Abgleichs mit den zugehörigen Ergebnissen

Ereignis	**Ergebnis**
Zwei Datensätze haben die gleiche identity_person_id und auch die gleiche MPI-ID/goldenID	True Positive
Zwei Datensätze haben die gleiche identity_person_id und **nicht** die gleiche MPI-ID/goldenID	False Negative
Zwei Datensätze haben **nicht** die gleiche identity_person_id und haben die gleiche MPI-ID/goldenID	False Positive
Zwei Datensätze haben **nicht** die gleiche identity_person_id und auch **nicht** die gleiche MPI-ID/goldenID	True Negative

Darüber hinaus ermöglicht das Python-Skript nach der Berechnung der Kennzahlen eine gezielte Ausgabe der uuid-Werte der jeweils betroffenen Datensätze. Auf diese Weise kann für jede Kategorie (True Positive, False Positive, False Negative, True Negative) nachvollzogen werden, welche konkreten Datensätze zugeordnet wurden.

4.2.3. Evaluation

Zur Bewertung der Verknüpfungsqualität wurden, wie in Abbildung 4.1 dargestellt, gängige Metriken wie Precision, Recall und der F1-Score verwendet. Für die Berechnung dieser Metriken wurden die mit dem Python-Skript automatisiert ermittelten Werte für False Positives, False Negatives, True Positives und True Negatives verwendet. Die Precision gibt den Anteil der Datensätze an, die als zusammengehörig identifiziert wurden und tatsächlich zusammengehören. Falsch-positive Zuordnungen wirken sich daher negativ auf die Precision aus. Diese Metrik wird nach folgender Formel berechnet:

$$P = \frac{|\text{true positives}|}{|\text{true positives}| + |\text{false positives}|}$$

Der Recall gibt an, wie viele der tatsächlich zusammengehörigen Datensätze korrekt erkannt wurden. Falsch-negative Ergebnisse können den Recall entsprechend verringern. Dieser wird nach folgender Formel berechnet:

$$R = \frac{|\text{true positives}|}{|\text{true positives}| + |\text{false negatives}|}$$

Der F1-Score kombiniert den Recall und die Precision und stellt somit ein Gleichgewicht zwischen beiden Metriken dar. Diese Kennzahl wird daher als harmonisches Mittel beschrieben. Der F1-Score kann mit folgender Formel berechnet werden:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Ein F1-Score von 1 weist auf ein perfektes Gleichgewicht zwischen Präzision und Recall hin, ohne falsch-positive oder falsch-negative Ergebnisse [66, S.262].

Abschließend wurde die Verknüpfungsqualität anhand dieser Metriken für jeden Testdurchlauf ausgewertet und für beide Lösungen gegenübergestellt.

4.2.4. Konfigurationen

Auf Basis des konzeptionellen und funktionalen Vergleichs wurden verschiedene Konfigurationen für den E-PIX und HAPI FHIR MDM entwickelt. Dabei wurde insbesondere darauf geachtet, die Konfigurationsparameter beider Lösungen, trotz ihrer unterschiedlichen Ansätze, so weit wie möglich anzugleichen, um eine vergleichbare Bewertung der Verknüpfungsqualität zu ermöglichen (siehe Ahnahng D und E).

Zunächst wurden zwei ”Basiskonfigurationen” getestet, die bewusst auf den Einsatz systemspezifischer Funktionalitäten verzichten (siehe Tabelle **??**. Ziel war es, einen ersten Vergleich der Verknüpfungsqualität zwischen den unterschiedlichen Record Linkage Verfahren von E-PIX und HAPI FHIR MDM zu ermöglichen und zugleich zu untersuchen, ob unter Verwendung verschiedener Algorithmen eine höhere Verknüpfungsqualität erzielt werden kann. In diesem ersten Testdurchlauf kamen ausschließlich Pflichtvariablen als Matching-Variablen zum Einsatz. In den Testdurchläufen 3 bis 5 wurden spezifische Funktionalitäten der beiden Lösungen in die jeweiligen Konfigurationen integriert, um deren Einfluss auf die Verknüpfungsqualität zu analysieren (vgl. Tabelle 4.4). Dabei standen insbesondere die Unterschiede in der Vorverarbeitung, die Implementierungsmöglichkeiten zur Identifikation von Multiple-Value-Feldern sowie die Fähigkeit zur Erkennung und Berücksichtigung von Spitznamen als Namenformen im Fokus. In dem nachfolgenden Testdurchlauf 6 wurden die Konfigurationen um eine zusätzliche Matching-Variable erweitert, um evaluieren zu können, inwieweit sich diese Erweiterungen auf die Verknüpfungsqualität auswirken.

Hervorzuheben ist, dass die definierten Schwellenwerte, Gewichtungen, Matching-Algorithmen sowie Matching-Regeln einen maßgeblichen Einfluss auf die erzielte Verknüpfungsqualität haben.

Tabelle 4.4.: Darstellung der Testdurchläufe mit den jeweiliegn Konfigurationen

Test	**Ziel**	**Matching-Variablen**	**Algorithmus**
1	Vergleich der Standardkonfigurationen beider Lösungen	Vorname, Nachname, Geburtsdatum, Geschlecht, Postleitzahl	Levenshtein-Distanz
2	Ergänzung der Standardkonfigurationen durch Kölner Phonetik für Vor- und Nachname	Vorname, Nachname, Geburtsdatum, Geschlecht, Postleitzahl	Kölner Phonetik, Levenshtein-Distanz
3	Vergleich der Vorverarbeitungsoptionen	Vorname, Nachname, Geburtsdatum, Geschlecht, Postleitzahl	Levenshtein-Distanz
4	Vergleich der Multiple-Value-Felder Optionen	Vorname, Nachname, Geburtsdatum, Geschlecht, Postleitzahl	Levenshtein-Distanz
5	Testen der Spitznamen-Funktion von HAPI FHIR MDM	Vorname, Nachname, Geburtsdatum, Geschlecht, Postleitzahl	Levenshtein-Distanz
6	Zusätzliche Matching-Variable testen	Pflichtvariablen + Krankenversichertennummer	Levenshtein-Distanz

5. Ergebnisse

In diesem Kapitel werden die Ergebnisse der durchgeführten Analyse dargestellt. Dabei erfolgt zunächst eine Gegenüberstellung der Konzepte des E-PIX und HAPI FHIR MDM, um die Anwendbarkeit sowie die Flexibilität der Lösungen zu bewerten, bevor im Anschluss die Verknüpfungsqualität in verschiedenen Szenarien gegenübergestellt und ebenfalls bewertet wird.

5.1. Gegenüberstellung der Konzepte

Die nachfolgende Gegenüberstellung von E-PIX und HAPI FHIR MDM erfolgt unter Berücksichtigung zentraler Aspekte wie der Systemarchitektur, der Identitätsverwaltung, der Mechanismen zur Datenvalidierung und -aufbereitung, des grundlegenden Matching-Verfahrens, der vorhandenen Konfigurationsmöglichkeiten sowie der Strategien zur Verwaltung von Verknüpfungen. Eine zusammenfassende tabellarische Darstellung stellt am Ende dieses Kapitels die wesentlichen Unterschiede und Gemeinsamkeiten der beiden Lösungen gegenüber (siehe Tabelle 5.1).

Beide betrachteten Record Linkage Lösungen sind als nicht-kommerzielle, frei verfügbare Open-Source-Ansätze konzipiert, die sich durch eine kontinuierliche Weiterentwicklung auszeichnen. Ziel beider Lösungen ist die zuverlässige Verwaltung und Zusammenführung von Identitäten, wobei sie unterschiedliche architektonische Ansätze verfolgen. Während E-PIX sowohl über eine Web-Oberfläche als auch über eine SOAP-basierte Schnittstelle sowie eine FHIR REST-Schnittstelle zugänglich ist, erfolgt der Zugriff auf HAPI FHIR MDM ausschließlich über eine RESTful API. Zwar stellt HAPI FHIR MDM eine Swagger-UI bereit, diese dient jedoch lediglich als technisches Frontend zur Nutzung der FHIR REST-API und stellt keine eigenständige Web-Oberfläche dar, was den E-PIX hinsichtlich der Nutzung flexibler gestaltet und die praktische Anwendbarkeit erleichtert. Der E-PIX ist primär als eigenständiges Identity Management Lösung ausgelegt, das durch ein FHIR-Gateway ergänzt werden kann. Im Gegensatz dazu ist HAPI FHIR MDM in die FHIR-Infrastruktur eingebettet und verwendet FHIR-Ressourcen durchgängig als zentrales Datenmodell. Diese konsequente Nutzung des Standards verleiht HAPI MDM eine Interoperabilität mit anderen FHIR-basierten Systemen. Der E-PIX hingegen ermöglicht durch die Unterstützung zusätzlicher Standards wie IHE PIX / PDQ eine breitere Abdeckung verschiedener Integrationsszenarien [2, 3, 6].

Im Hinblick auf die *Identitätsverwaltung* weisen E-PIX und HAPI FHIR MDM sowohl strukturelle Gemeinsamkeiten als auch terminologische und konzeptionelle Unterschiede auf. Beide Lösungen basieren auf dem Prinzip einer zentralen Referenzidentität, die mit weiteren, potenziell abweichenden Datensätzen verknüpft werden kann. Diese Datensätze können in beiden

Lösungen als Varianten oder Duplikate einer Person verstanden werden und dienen der Abbildung multipler Identitäten. Unterschiede zeigen sich in diesem Fall in der Begriffsverwendung. E-PIX spricht in Bezug auf die zentrale Referenzidentität von einer ”Hauptidentität”, während HAPI FHIR MDM die Bezeichnung ”Goldene Ressource” für den als korrekt anerkannten Datensatz verwendet. Die zugehörigen, als Duplikate betrachteten, Identitäten werden im E-PIX als Nebenidentitäten, in HAPI FHIR MDM hingegen als Quellressourcen geführt [2,3]. Sowohl der E-PIX als auch Hapi FHIR MDM verfolgt im Rahmen der Zusammenführung von Identitäten einen ID-basierten Ansatz, bei dem Datensätzen, die als übereinstimmend erkannt werden, im E-PIX eine gemeinsame MPI-ID und in Hapi MDM eine gemeinseme Golden-ID zugewiesen wird [2, 3].

Im Hinblick auf die verwendeten *Matching-Verfahren*, lassen sich grundzätzliche Unterschiede feststellen. Während der E-PIX ein probabilistisches Matching-Verfahren verwendet, basiert der Abgleich in HAPI FHIR MDM auf einer deterministischen Methode. Zwar unterstützt E-PIX auch deterministische Vergleiche, etwa durch die Möglichkeit exakter Abgleiche, jedoch erfolgt die Entscheidungsfindung über ein Bewertungssystem, das keine regelbasierten ”Wenn-Dann-Strukturen” im klassischen Sinne vorsieht. Das Matching-Ergebnis ergibt sich im E-PIX immer aus einer gewichteten Bewertung mehrerer Kriterien und nicht aus einer festgelegten Regelabfolge. Demgegenüber verfolgt HAPI FHIR MDM einen Ansatz, bei dem die Matching-Logik auf explizit definierten Regeln basiert, die die Bedingungen für eine Übereinstimmung eindeutig festlegen. Damit entspricht das Verfahren einem regelbasierten Record Linkage.

Der Matching-Prozess zeigt in beiden Lösungen strukturelle Parallelen, weist jedoch ebenfalls konzeptionelle Unterschiede auf. Sowohl in E-PIX als auch in HAPI FHIR MDM ist es möglich, verschiedene *Matching-Typen* zu definieren und manuelle Abgleiche durchzuführen, sofern eine eindeutige Zuordnung automatisiert nicht erfolgen kann. Die Kategorisierung von Matching-Ergebnissen unterscheidet sich dabei hinsichtlich der Differenzierung, der Terminologie und der Auslegung. In beiden Lösungen wird eine Nicht-Übereinstimmung als ”No Match” klassifiziert. Der Matching-Typ ”Possible Match” ist ebenfalls in beiden Lösungen verfügbar und wird verwendet, wenn teilweise Übereinstimmungen vorliegen, die jedoch nicht ausreichen, um automatisch eine verlässliche Zusammenführung vorzunehmen. In diesen Fällen erfolgt bei beiden Record Linkage Lösungen eine manuelle Prüfung zur finalen Entscheidungsfindung. Bei als zusammengehörig identifizierten Datensätzen unterscheiden sich jedoch die Bezeichnungen und zugrundeliegenden Kriterien. Der E-PIX verwendet für vollständig identische Datensatzpaare die Bezeichnung ”Perfect Match”, die eine exakte Übereinstimmung aller Ausprägungen der Matching-Variablen voraussetzt. In HAPI FHIR MDM hingegen wird der Begriff ”Match” verwendet, sobald sämtliche vordefinierten Matching-Regeln erfüllt sind, unabhängig davon ob die Ausprägungen der einzelnen Felder exakt übereinstimmen. Ein direktes Pendant zum ”Perfect Match” existiert in HAPI FHIR MDM somit nicht, kann jedoch durch eine entsprechend strikte Konfiguration der Matching-Kriterien in HAPI FHIR MDM funktional angenähert werden.

Wird beispielsweise in einer Regel gefordert, dass alle Matching-Attribute exakt übereinstimmen müssen, um als zusammengehörig identifiziert zu werden, entspricht der Typ ”Match” in HAPI MDM dem ”Perfect Match” aus dem E-PIX. Die Gleichwertigkeit der beiden Typen hängt folglich von der jeweiligen Konfiguration des Matching-Prozesses ab. Der E-PIX verfügt zusätzlich über den Matching-Typ ”Automatic Match”, bei dem eine hinreichend hohe Ähnlichkeit der Matching-Variablen vorliegt, um eine automatische Zusammenführung ohne manuelle Prüfung zu erlauben, auch wenn keine exakte Übereinstimmung gegeben ist. Ähnlich wie der Vergleich zwischen den Matching-Typen ”Perfect Match” im E-PIX und ”Match” in HAPI FHIR MDM lässt sich auch für den Matching-Typ ”Automatic Match” eine funktionale Parallele in HAPI FHIR MDM herstellen. Dies gilt dann, wenn die zugrunde liegenden Matching-Regeln in HAPI FHIR MDM so definiert sind, dass bereits bestimmte Ähnlichkeiten, ohne vollständige Übereinstimmung aller Merkmalsausprägungen, ausreichen, um den Status ”Match” zu vergeben. In diesem Fall erfüllt der Matching-Typ ”Match” in HAPI FHIR MDM eine vergleichbare Funktion wie der ”Automatic Match” im E-PIX, bei dem ebenfalls eine automatisierte Verknüpfung auf Basis hinreichender Ähnlichkeit erfolgt. Bei HAPI FHIR MDM ist eine derartige Differenzierung zwischen zusammengehörigen Datensätzen jedoch nicht möglich, da ausschließlich der Matching-Typ ”Match” verwendet wird. Ein weiterer Matching-Typ des E-PIX ist ”Multiple Match”, der dann vergeben wird, wenn mehrere potenziell zutreffende Identitäten einer Referenzidentität zugeordnet werden können. Für diesen Fall bietet HAPI MDM keinen Matching-Typ, stattdessen verwendet HAPI MDM zusätzlich den Typ ”Possible Duplicate”, der dann zur Anwendung kommt, wenn zwischen zwei bereits bestehenden Goldenen Ressourcen ein potenzielles Duplikat vermutet wird [2, 3]. Hinsichtlich der Anwendbarkeit zeigt sich, dass beide Lösungen eine vergleichbare Grundfunktionalität für das Matching bieten, indem verschiedene Matching-Typen definiert und manuelle Prüfungen durchgeführt werden können.

Als nächstes werden die Konfigurationsmöglichkeiten beider Lösungen gegenübergestellt. Im E-PIX können Anpassungen sowohl über die bereitgestellte Web-Oberfläche als auch über eine XML-basierte Konfigurationsdatei vorgenommen werden. HAPI FHIR MDM hingegen verzichtet auf eine grafische Benutzeroberfläche zur Konfiguration und verwendet ausschließlich JSON-Dateien zur Definition des Matching-Verhaltens. Hinzukommt, dass die Matching-Regeln in der JSON-Datei von HAPI FHIR MDM nicht beliebig und zur Laufzeit angepasst werden können. Stattdessen ist es erforderlich, sich im Vorfeld auf eine feste Konfiguration festzulegen und diese über Maven zu kompilieren. Soll eine Anpassung erfolgen, muss die entsprechende Konfigurationsdatei ausgetauscht und der Code erneut kompiliert werden. Da die Konfiguration anschließend fest im Server integriert ist, hat ein isolierter Austausch der JSON-Datei zunächst keine Wirkung. Im Record Linkage Prozess ist es jedoch üblich, unterschiedliche Konfigurationen einzusetzen, da diese regelmäßig an die spezifischen Anforderungen eines jeweiligen Forschungsvorhabens angepasst werden müssen. Im E-PIX ist es daher möglich, für verschiedene Projekte auch unterschiedliche Konfigurationen für den Abgleich zu hinterlegen. Das Fehlen einer Benutzeroberfläche reduziert die Benutzerfreundlichkeit und erfordert ein höheres Maß an

technischem Know-how für die Anpassung der Konfigurationsoptionen. Die fehlende Möglichkeit, Konfigurationsdateien ohne erneutes Kompilieren auszutauschen, reduziert die Flexibilität erheblich und schränkt außerdem die praktische Anwendbarkeit von HAPI FHIR MDM im Vergleich zu E-PIX ein. Beide Lösungen bieten grundlegende Möglichkeiten zur *Vorverarbeitung* von Daten, insbesondere im Hinblick auf die Vereinheitlichung der Schreibweise. So besteht im E-PIX ebenso wie in HAPI FHIR MDM die Option, sämtliche Zeichen in Großbuchstaben zu überführen sowie diakritische Zeichen zu entfernen, um eine konsistente Vergleichsbasis für den Matching-Prozess zu schaffen. Darüber hinaus verfügt der E-PIX über weitergehende Funktionen zur Datenaufbereitung. Hierzu zählt die Möglichkeit, bestimmte Zeichenfolgen gezielt durch vordefinierte Alternativen zu ersetzen sowie Regeln zu definieren, anhand derer Datensätze gefiltert werden, bevor sie in das System übernommen werden. Eine vergleichbare Funktionalität ist in HAPI FHIR MDM in dieser Form nicht vorgesehen. Ein weiterer Unterschied zeigt sich in der Konfiguration der Datenvalidierung. E-PIX erlaubt die Definition von Regeln zur Überführung eingehender Daten in ein spezifisches, validiertes Format. HAPI FHIR MDM hingegen bietet keine direkt integrierte Möglichkeit zur formalen Validierung oder Formattransformation der Daten im Rahmen der Vorverarbeitung. Stattdessen wird lediglich überprüft, ob in der jeweiligen Ressource mindestens eine der definierten Matching-Variablen vorhanden ist [2, 3]. Auch in diesem Punkt bietet der E-PIX eine größere Flexibilität in Bezug auf die Anpassung, da mehr Konfigurationsoptionen zur Verfügung stehen als bei HAPI FHIR MDM.

Sowohl E-PIX als auch HAPI FHIR MDM verwenden *Blocking-Mechanismen*, um den Suchraum möglicher Duplikate effizient einzugrenzen, unterscheiden sich jedoch in ihrer Herangehensweise. E-PIX setzt auf ein scorebasiertes Blocking, bei dem für jede potenzielle Übereinstimmung eine vorläufige Bewertung anhand definierter Matching-Kriterien und Vergleichsalgorithmen durchgeführt werden kann. In diesem Kontext können Schwellenwerte festgelegt werden, die bestimmen, ab welchem Punkt ein Datensatz als relevanter Kandidat für einen Abgleich berücksichtigt wird. Zudem erlaubt der E-PIX die Festlegung, ob ein Feld als Text oder Zahl interpretiert wird, was wiederum Einfluss auf den verwendeten Vergleichsalgorithmus hat. Bei HAPI FHIR MDM können im Rahmen des Blockings lediglich die Blocking-Variablen angegeben werden, jedoch keine Schwellenwerte oder Algorithmen, die einen weniger strengen Vergleich ermöglichen würden. Dies kann dazu führen, dass zusammengehörige Datensätze bereits vor dem Matching-Prozess ausgeschlossen werden, wenn die Daten leichte Abweichungen oder Fehler enthalten [2, 3]. Durch das Setzen von Schwellenwerten und der gezielten Zuordnung von Algorithmen ermöglicht der E-PIX eine flexiblere Anpassung der Blocking-Mechanismen im Vergleich zu HAPI FHIR MDM.

Sowohl HAPI FHIR MDM als auch der E-PIX bieten umfangreiche Konfigurationsmöglichkeiten im Rahmen der Matching-Variablen, wobei sich die Optionen in Bezug auf die zugrunde liegenden Ansätze des Record Linkage deutlich unterscheiden. In beiden Lösungen ist es möglich,

für jede Matching-Variable einen Algorithmus auszuwählen, der den Vergleich der Ausprägungen zweier Datensätze steuert. Die beiden Lösungen unterscheiden sich dabei unter anderem hinsichtlich der Algorithmen, die zur Verfügung stehen. Beide Record Linkage Lösungen nutzen etablierte *Matching-Algorithmen* wie die ”Kölner Phonetik”, den ”Jaccard-Koeffizienten”, den ”Sørensen-Dice-Koeffizienten” und einen exakten Abgleich. Der E-PIX bietet neben den in beiden Lösungen implementierten Matching-Algorithmen zusätzlich die Möglichkeit, die Levenshtein-Distanz zu verwenden, die in der Praxis häufig zur Bewertung der Ähnlichkeit von Zeichenketten eingesetzt wird. In HAPI FHIR MDM ist die Levenshtein-Distanz hingegen nicht standardmäßig integriert und muss bei Bedarf durch die Implementierung einer eigenen Java-Klasse manuell ergänzt werden. HAPI FHIR MDM bietet jedoch ebenfalls eine erweiterte Auswahl an Algorithmen, die über diese gemeinsame Basis hinausgeht. Dazu gehören zusätzliche phonetische Verfahren wie ”Soundex” ”Double Metaphone”, ”Caverphone 1 & 2”, ”NYSIIS”, ”Match Rating Approach” und ”Metaphone”, die besonders bei internationalen oder sprachlich variierenden Namen nützlich sind. Darüber hinaus umfasst HAPI FHIR MDM Fuzzy-Matching-Verfahren, darunter ”Jaro-Winkler” und ”Cosine Similarity” sowie spezialisierte Algorithmen für Teilzeichenfolgen, numerische Werte, Spitznamen, Namensvariationen und leere Felder. Viele dieser Verfahren sind auch in numerisch optimierten Varianten verfügbar [2, 3].

Beide Lösungen bieten die Möglichkeit, einen Schwellenwert zu definieren, der festlegt, ab welchem Wert die verglichenen Ausprägungen einer Matching-Variable als übereinstimmend gelten. Dieser Schwellenwert kann bei beiden Lösungen innerhalb eines Bereichs von 0.0 bis 1.0 festgelegt werden, wobei in beiden Lösungen 0.0 eine vollständige Nichtübereinstimmung (0 %) und 1.0 eine vollständige Übereinstimmung (100 %) repräsentiert. Sobald der festgelegte Schwellenwert erreicht oder überschritten wird, klassifiziert der E-PIX und auch HAPI FHIR MDM die Ausprägungen der Matching-Variablen als übereinstimmend. Im E-PIX kann unabhängig vom gewählten Algorithmus stets ein individueller Schwellenwert zur Bewertung der Ähnlichkeit definiert werden. Im Gegensatz dazu erlaubt HAPI FHIR MDM die Konfiguration eines Schwellenwerts nur bei Verwendung spezifischer Algorithmen, nämlich der Levenshtein-Distanz, des Sørensen-Dice-Koeffizienten und des Jaccard-Koeffizienten. Da der E-PIX auf einem probabilistischen Record Linkage Modell basiert, bietet die Lösung zusätzlich die Möglichkeit, Gewichte für die einzelnen Matching-Variablen festzulegen. Diese Flexibilität zur Gewichtung der Variablen ist in HAPI FHIR MDM hingegen nicht vorgesehen [2, 3]. In Bezug auf die verfügbaren Matching-Algorithmen lässt sich festhalten, dass HAPI FHIR MDM eine größere Auswahl an Algorithmen bietet, wodurch die Flexibilität bei der Wahl des passenden Verfahrens höher ist als beim E-PIX. Hinsichtlich der Anwendbarkeit der Algorithmen bietet E-PIX hingegen mehr Möglichkeiten, da hier für alle verfügbaren Algorithmen Schwellenwerte für die Ähnlichkeit gesetzt werden können. Bei HAPI FHIR MDM ist dies nur eingeschränkt und vereinzelt möglich.

Beide Record Linkage Lösungen bieten die Möglichkeit, *Multiple-Value-Felder* zu berücksichtigen, jedoch unterscheidet sich die Handhabung dieser Felder zwischen den Lösungen. In HAPI

FHIR MDM wird die Verarbeitung von Multiple-Value-Feldern durch das Element ”fhirPath” in der JSON-Datei gesteuert, wodurch lediglich festgelegt werden kann, dass ausschließlich der erste Wert eines Multiple-Value-Feldes miteinander verglichen wird. Es gibt jedoch keine Möglichkeit, explizite Strafen für ungenaue Übereinstimmungen festzulegen, wie es im E-PIX der Fall ist. Im E-PIX können durch diese spezifische Strafregeln wie ”penalty-not-a-perfect-match” oder ”penalty-one-short” auch kleine Abweichungen zwischen nahezu identischen, aber nicht exakt übereinstimmenden Daten und fehlende Werte in Multiple-Value-Feldern berücksichtigt werden. Diese Strafen führen dazu, dass das Gesamtgewicht des Datensatzpaares entsprechend verringert wird, wodurch dieses weniger wahrscheinlich als übereinstimmend erkannt wird. Dies ermöglicht eine fein abgestimmte Behandlung von Abweichungen in Multiple-Value-Feldern. In HAPI FHIR MDM wird aufgrund des deterministischen Ansatzes kein Gesamtgewicht für die Datenpaare berechnet, daher können keine Strafen umgesetzt werden, die dazu führen würden, dass das Datenpaar weniger schnell als Übereinstimmend angesehen wird [2, 3]. In Bezug auf die Flexibilität im Umgang mit Multiple-Value-Feldern bietet der E-PIX demnach deutlich mehr Konfigurationsmöglichkeiten, was aufgrund des probabilistischen Record Linkage Ansatz ermöglicht wird.

Die beiden Record Linkage Lösungen unterscheiden sich vor allem hinsichtlich der *Klassifikation* der Datenpaare. Im E-PIX werden die Ausprägungen aller Matching-Variablen miteinander verglichen, wobei für jede Variable entschieden wird, ob die Übereinstimmung den festgelegten Schwellenwert überschreitet. Dieses Vorgehen wird auch in HAPI FHIR MDM umgesetzt. Der zentrale Unterschied zwischen den beiden Record Linkage Lösungen zeigt sich in der Art und Weise, wie die Ergebnisse dieser Vergleiche im weiteren Prozess verarbeitet werden. Im E-PIX fließen nach dem Abgleich zusätzlich die konfigurierten Gewichtungen der einzelnen Variablen in die Berechnung eines Gesamtgewichts für das Datenpaar ein. Dieses Gesamtgewicht wird anhand der festgelegten Schwellenwerte bewertet, um eine endgültige Klassifizierung des Datensatzpaares vorzunehmen. In HAPI FHIR MDM erfolgt die Entscheidungsfindung hingegen ausschließlich auf der Grundlage der festgelegten Matching-Regeln und der Übereinstimmung der Matching-Variablen. Hier wird geprüft, welche Variablen eine Übereinstimmung oder Nichtübereinstimmung aufweisen und ob diese Ergebnisse mit den definierten Regeln für die Zuordnung eines bestimmten Matching-Typs übereinstimmen. HAPI FHIR MDM berechnet daher kein Gesamtgewicht für ein Datensatzpaar, sondern trifft eine Klassifizierung basierend auf der Erfüllung spezifischer Regeln und der Übereinstimmung der jeweiligen Variablen [2, 3]. Die Verwendung der unterschiedlichen Matching-Verfahren wirkt sich auf die Anwendbarkeit und Flexibilität der beiden Lösungen aus. Während bei HAPI FHIR MDM die Matching-Regeln festlegen, wann Datensätze als zusammengehörig klassifiziert werden, ermöglicht der probabilistische Ansatz des E-PIX eine differenziertere Bewertung der Zusammengehörigkeit von Datensätzen. Hierbei können die Schwellenwerte, die das Gesamtgewicht überschreiten müssen, um eine Übereinstimmung zu erkennen, individuell konfiguriert werden. Dies erlaubt eine feinere Abstufung zwischen Datensätzen, bei denen alle Matching-Variablen exakt übereinstim-

men, und solchen, bei denen eine sehr hohe Ähnlichkeit vorliegt, jedoch Unterschiede vorliegen. Dadurch kann der E-PIX zusätzliche Differenzierungen zwischen zusammengehörigen Datensätzen vornehmen, was die Flexibilität erhöht. Eine solch feine Abstufung, wie sie bei probabilistischen Verfahren durch das Gesamtgewicht eines Datensatzpaares und dem Setzen von Schwellenwerten möglich ist, findet bei einem deterministischen Ansatz nicht statt. Hinsichtlich der Anwendbarkeit lässt sich sagen, dass die im probabilistischen Ansatz von E-PIX mögliche feine Abstufung der Übereinstimmung die sorgfältige Konfiguration durch qualifizierte Anwender erfordert. Die strikt definierten Regeln von HAPI FHIR MDM hingegen machen die Lösung einfacher handhabbar.

Ein weiterer Unterschied betrifft den Umgang mit sensiblen Daten. E-PIX bietet die Möglichkeit, wahlweise ein Matching auf Basis unverschlüsselter Daten (Klartext) oder ein datenschutzfreundliches Verfahren im Sinne des *PPRL* durchzuführen. Diese Option besteht in HAPI FHIR MDM nicht, hier erfolgt der Datenabgleich ausschließlich auf der Grundlage von Klartextdaten.

Sowohl E-PIX als auch HAPI FHIR MDM bieten Funktionen zur Verwaltung von personenbezogenen Daten und deren Verknüpfungen. Beide Lösungen ermöglichen mit einem *Verknüpfungsmanagement* eine gezielte Suche und Abfrage von Verknüpfungen, sei es anhand der zugewiesenen IDs oder durch Filterung nach spezifischen Kriterien. Darüber hinaus bieten beide Lösungen die Möglichkeit, Verknüpfungen zu aktualisieren, zu löschen und neue zu erstellen. Eine weitere Funktion von HAPI FHIR MDM ist die Möglichkeit, eine Link-Historie anzuzeigen, die alle Verknüpfungen und deren Revisionsstempel im Zusammenhang mit goldenen Ressourcen speichert. Diese Funktionalität wird im E-PIX ähnlich durch ein Protokoll ersetzt, das sämtliche Änderungen und Aktivitäten im Kontext der Verwaltung von Personenidentitäten und deren Verknüpfungen aufzeichnet. Zusätzlich bietet E-PIX eine Funktion zur Anzeige von Statistiken, die weiterführende Informationen zu den gesuchten Personen bereitstellt [2, 3].

Sowohl E-PIX als auch HAPI FHIR MDM stellen Lösungen für das Identitätsmanagement und die Duplikaterkennung dar, die unterschiedliche Ansätze in Bezug auf Architektur und Matching-Prozess verfolgen (siehe Tabelle 5.1). Der E-PIX bietet eine hohe Flexibilität in Bezug auf die Zugriffsmöglichkeiten, der Konfiguration sowie der Datenaufbereitung, während HAPI FHIR MDM in diesen Hinsichten insgesamt weniger Möglichkeiten bietet. Jedoch besteht in HAPI FHIR MDM eine größere Auswahl an Algorithmen und es werden zusätzlich spezielle Abgleichsfunktionen wie eine Prüfung auf Spitznamen angeboten. HAPI FHIR MDM ist vollständig auf die FHIR-Umgebung ausgerichtet und ermöglicht dadurch eine hohe Interoperabilität, da die Lösung auf umfassenden Standardisierungen basiert. Der E-PIX unterstützt darüber hinaus jedoch auch weitere Standards wie beispielsweise verschiedene IHE-Profile neben FHIR. Letztlich bieten beide Lösungen umfangreiche Anpassungsmöglichkeiten im Matching-Prozess, wobei E-PIX ein probabilistisches Verfahren nutzt, das eine feinere Anpassung und differenzierte Handhabung von Duplikaten ermöglicht, als der deterministische Ansatz von Hapi FHIR MDM.

Tabelle 5.1.: Zusammenfassung des Vergleichs der Konzepte von E-PIX und HAPI FHIR MDM

	E-PIX	**HAPI FHIR MDM**
Systemarchitektur	Eigenständige Identity-Management-Lösung mit optionalem FHIR-Gateway, Open-Source, nicht-kommerziell, Java-basiert	In die FHIR-Infrastruktur eingebettet, HAPI FHIR JPA Server, Open-Source, nicht-kommerziell, Java-basiert
Schnittstellen	Web-Oberfläche, SOAP-Schnittstelle, FHIR REST API	FHIR REST API
Standardkonformität	FHIR, IHE PIX/PDQ	FHIR, Erweiterungen des Frameworks sind notwendig, um zusätzliche Standards zu unterstützen
Identitätsverwaltung	Haupt-/Nebenidentitäten, MPI-ID als gemeinsamer Identifikator	Goldene-/Quellressourcen, Golden-ID als gemeinsamer Identifikator
Matching-Verfahren	Probabilistisch	Deterministisch
Matching-Typen	Perfect Match, Automatic Match, Possible Match (Nicht unter der Verwendung des FHIR-Gateways), No Match, Multiple Match	Match, Possible Match, No Match, Possible Duplicate
Konfiguration	Web-UI/XML-Datei	JSON-Datei
Datenaufbereitung	Großschreibung, Diakritikentfernung, Ersetzungsregeln, Validierungsregeln	Großschreibung, Diakritikentfernung
Blocking/Indexing	Scorebasiert mit Schwellenwerten	Exakter Abgleich der Blocking-Variablen
Matching-Algorithmen	Kölner Phonetik, Levenshtein, Jaccard, Sørensen-Dice, deterministischer Abgleich	Kölner Phonetik, (Levenshtein), Jaccard, Dice, Soundex, Jaro-Winkler, Cosine Similarity, Metaphone, Double Metaphone, Caverphone 1 & 2, NYSIIS, Matching Rating Approach
Multiple-Value-Felder	Unterstützt, inkl. Strategie für Abweichungen	Unterstützt via FHIRPath ohne Strategie für Abweichungen
Klassifizierung	Gesamtgewicht mit Schwellenwerten und Wahrscheinlichkeiten	Regelbasierte Entscheidung
Privacy Preserving Matching	Ja	Nein
Verknüpfungsmanagement	Suchen und Aktualisieren von Verknüpfungen, Protokoll mit Änderungsverlauf & Statistiken	Suchen und Aktualisieren von Verknüpfungen, Link-Historie mit Revisionsstempeln

5.2. Testdurchläufe E-PIX

Im Folgenden werden die Ergebnisse der Testdurchläufe des E-PIX dargestellt. Hervorzuheben ist, dass hierbei ausschließlich die Datensätze als zusammengehörig identifiziert werden konnten, die vom E-PIX als ”Perfect-Match” oder ”Automatic-Match” klassifiziert wurden, da eine Ausgabe von ”Possible-Matches” über das FHIR-Gateway derzeit noch nicht unterstützt wird.

Testdurchlauf 1 und 2

In den ersten beiden Testdurchläufen wurden die in Kapitel 4.2.4 beschriebenen Konfigurationen 1 und 2 des E-PIX evaluiert (siehe Anhang D.1 und D.2). In beiden Konfigurationen dienten die Pflichtvariablen ”Vorname”, ”Nachname”, ”Geburtsdatum”, ”Geschlecht” und ”Postleitzahl” als Matching-Variablen, wobei die Levenshtein-Distanz als Ähnlichkeitsmaß verwendet wurde. Zusätzlich kam in der zweiten Konfiguration die ”Kölner Phonetik” für den Abgleich des Vor- und Nachnamen zum Einsatz, um den Einfluss unterschiedlicher Ähnlichkeitsmaße auf die Verknüpfungsqualität zu untersuchen. Die Gewichte und Schwellenwerte wurden in beiden Konfigurationen gleich gesetzt, sodass die Auswirkungen der variierenden Ähnlichkeitsmetriken auf die Matching-Entscheidungen analysiert werden konnten.

Tabelle 5.2.: Gegenüberstellung der Ergebnisse der Testdurchläufe 1 und 2 des E-PIX, Verwendete Abkürzungen: Konfig - Konfiguration, TP – True Positives, FP – False Positives, TN – True Negatives, FN – False Negatives

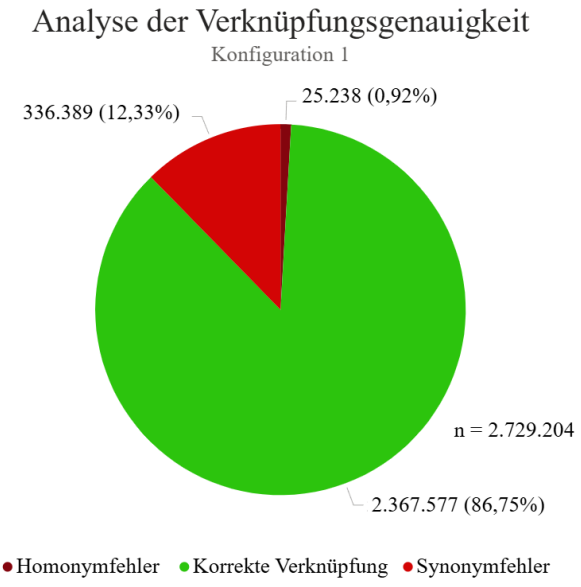
Nutzung von Basiskonfigurationen							
Konfig.	**TP**	**TN**	**FP**	**FN**	**Precision**	**Recall**	**F1-Score**
1	2.264.883	102.694	25.238	336.389	0,9939	0,8706	0,9287
2	2.387.545	102.724	26.074	212.861	0,9892	0,9182	0,9510

Die Ergebnisse in Tabelle 5.2 zeigen, dass die Verknüpfungsqualität mit beiden Konfigurationen insgesamt sehr hoch ausfällt, mit Werten über 85% in allen relevanten Metriken. Die zweite Konfiguration erzielt jedoch insgesamt eine höhere Verknüpfungsqualität als die erste (siehe Tabelle 5.2). In der ersten Konfiguration wurde eine Precision von 99,39% erreicht, während im zweiten Testdurchlauf eine leicht reduzierte Precision von 98,92% gemessen wurde. Dies bedeutet, dass im zweiten Durchlauf mehr Datensätze fälschlicherweise als zusammengehörig verknüpft wurden. Entsprechend zeigt sich auch in den absoluten Zahlen ein leichter Anstieg der Homonymfehler von 25.238 im ersten Testdurchlauf auf 26.074 im zweiten Testdurchlauf (siehe Abbildung 5.1). Durch den Einsatz der Kölner Phonetik in der zweiten Konfiguration konnte der Recall im Vergleich zum ersten Testdurchlauf gesteigert werden. Während in der ersten Konfiguration ein Recall von 87,06% erreicht wurde, liegt der Recall im zweiten Durchlauf

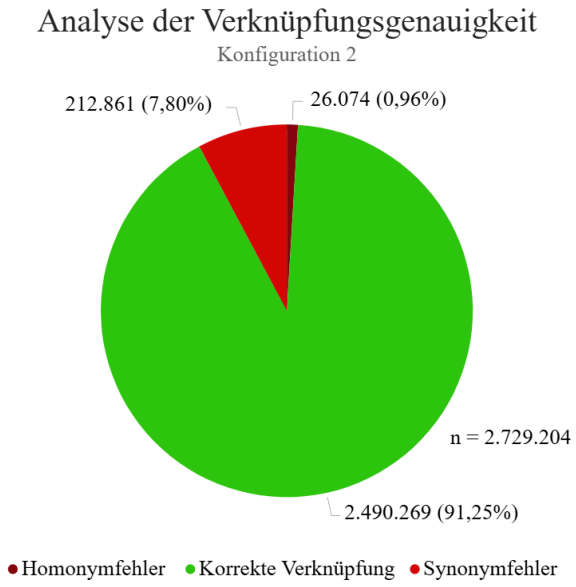
bei 91,82% (vgl. Tabelle 5.2). Dies verdeutlicht, dass in der ersten Konfiguration deutlich mehr Synonymfehler auftraten. Betrachtet man die Kreisdiagramme in Abbildung 5.1 wird deutlich, dass im ersten Testdurchlauf 336.389 falsch negative Ergebnisse auftraten und sich die Anzahl im zweiten Testdurchlauf auf 212.861 reduziert hat.

Der F1-Score fällt ebenso wie der Recall in der zweiten Konfiguration mit 95,10% höher aus als in der ersten Konfiguration mit 92,87% (siehe Tabelle 5.2. Dies verdeutlicht, dass die Balance zwischen Recall und Precision durch die Kombination der Kölner Phonetik für die Variablen Vor- und Nachname mit der Levenshtein-Distanz für Geburtsdatum, Geschlecht und Postleitzahl besser ausfällt, bei der ausschließlichen Verwendung der Levenshtein-Distanz auf alle Matching-Variablen. Der F1-Score spiegelt somit wider, dass in der zweiten Konfiguration beide Ziele gleichzeitig in hohem Maße erreicht werden. Der E-PIX gewährleistet unter Einbeziehung der Kölner Phonetik für Namensfelder und der Levenshtein-Distanz für die weiteren Pflichtvariablen damit eine ausgewogenere Datenverknüpfung als bei der ausschließlichen Nutzung der Levenshtein-Distanz.

In Abbildung 5.1 wird außerdem erneut die hohe Verknüpfungsqualität beider Konfigurationen deutlich, denn der größte Teil der Datensätze wurde unter Verwendung von Konfiguration 1 mit 86,76% sowie unter Verwendung von Konfiguation 2 mit 91,25% korrekt erkannt. Die Möglichkeit, jeder Matching-Variable ein spezifisches Ähnlichkeitsmaß zuzuordnen, kann demnach zu unterschiedlichen Matching-Entscheidungen führen und erlaubt zugleich eine flexible Anpassung an die jeweils verfügbaren Variablen.



(a) Konfiguration 1 des E-PIX



(b) Konfiguration 2 des E-PIX

Abbildung 5.1.: Verteilung korrekter Verknüpfungen und Fehlerarten in Konfiguration 1 und 2 des E-PIX.

Testdurchlauf 3

Im dritten Testdurchlauf wurde die Konfigurierbarkeit des E-PIX im Hinblick auf den Umgang mit Multiple-Value-Feldern untersucht. Zur Sicherstellung der Vergleichbarkeit kamen erneut die Pflichtvariablen ”Vorname”, ”Nachname”, ”Geschlecht”, ”Geburtsdatum” und ”Postleitzahl” in Kombination mit der Levenshtein-Distanz zum Einsatz. Darüber hinaus wurden in beiden Konfigurationen identische Gewichte und Schwellenwerte festgelegt, um die Gegenüberstellung der Ergebnisse zu gewährleisten (siehe Anhang D.3 und D.1).

Tabelle 5.3.: Gegenüberstellung der Ergebnisse der Testdurchläufe 3 und 1 des E-PIX, Verwendete Abkürzungen: Konfig - Konfiguration, TP – True Positives, FP – False Positives, TN – True Negatives, FN – False Negatives

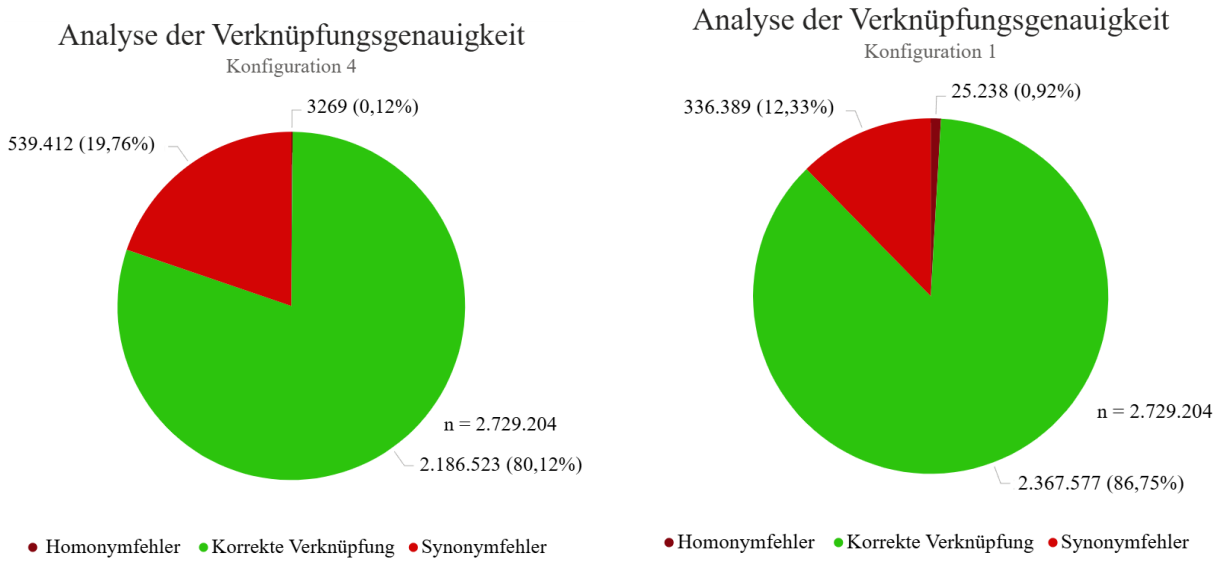
Nutzung der Multiple-Value-felder-Optionen								
Konfig.	**Beschreibung**	**TP**	**TN**	**FP**	**FN**	**Precision**	**Recall**	**F1-Score**
4	Ohne Multiple-Value-Funktion	2.081.929	104.594	3.269	539.412	0,9984	0,7942	0,8847
1	Mit Multiple-Value-Funktion	2.264.883	102.694	25.238	336.389	0,9939	0,8706	0,9287

Ein Vergleich der dritten Konfiguration ohne Verwendung der Multiple-Value-Funktion mit der ersten Konfiguration, in der diese Funktion integriert ist, zeigt, dass die Verknüpfungsqualität durch die Nutzung der Multiple-Value-Funktion insgesamt verbessert wird (siehe Tabelle 5.3). Die Precision ist unter Verwendung der dritten Konfiguration mit 99,84% zwar minimal höher als in der ersten Konfiguration mit 99,39%, was darauf hinweist, dass in der dritten Konfiguration weniger falsch positive Verknüpfungen entstanden sind, jedoch liegen diese Homonymfehler bei beiden Konfigurationen unter 1%, wie in Abbildung 5.2 dargestellt.

Stärker unterscheidet sich hingegen der Recall. Unter Verwendung der Multiple-Value-Funktion wird ein Recall von 87,06% erreicht, während dieser ohne Integration der Funktion lediglich 79,42% beträgt. Dies bedeutet, dass ohne den Einsatz dieser Multiple-Value-Funktion signifikant mehr Synonymfehler auftreten, also tatsächlich zusammengehörige Datensätze nicht korrekt identifiziert werden. Abbildung 5.2 verdeutlicht diesen Unterschied, denn ohne Multiple-Value-Funktion liegt der Anteil der Synonymfehler bei 19,76% (539.412), mit der Funktion reduziert sich dieser auf 12,33% (3.269).

Der F1-Score liegt ebenso wie der Recall bei Verwendung der Multiple-Value-Funktion mit 92,87% höher als in einer Konfiguration ohne diese Funktion, in der ein F1-Score von 88,47%

erreicht wurde. Dies verdeutlicht, dass durch die Funktion des E-PIX ein ausgewogeneres Verhältnis zwischen Recall und Precision erreicht wird, was wiederum auf eine insgesamt gesteigerte Verknüpfungsqualität hinweist.Trotz dieser Unterschiede bildet in beiden Konfigurationen der Anteil korrekt erkannter Datensätze erneut die größte Gruppe und liegt jeweils über 80% (siehe Abbildung 5.2). Dies bestätigt die insgesamt hohe Verknüpfungsqualität des E-PIX, die durch den Einsatz der Multiple-Value-Funktion erhöht werden kann.



(a) Konfiguration ohne Multiple-Value-Funktion (b) Konfiguration mit Multiple-Value-Funktion

Abbildung 5.2.: Verteilung korrekter Verknüpfungen und Fehlerarten in beiden Konfigurationen des E-PIX.

Testdurchlauf 4

In der vierten Konfiguration wurde neben den Pflichtvariablen zusätzlich die Krankenversichertennummer als Matching-Variable herangezogen (siehe Anhang D.4). Durch diese Erweiterung reduzierte sich die Grundgesamtheit des Testdatensatzes aus dem Krebsregister Mecklenburg-Vorpommern von 2.729.204 auf 257.393 Patientendatensätze, da die Krankenversichertennummer nicht in allen Datensätzen verfügbar war.

Tabelle 5.4.: Darstellung der Ergebnisse von Testdurchlauf 4, Verwendete Abkürzungen: Konfig. - Konfiguration, TP – True Positives, FP – False Positives, TN – True Negatives, FN – False Negatives

Nutzung der Krankenversichertennummer als zusätzliche Matching-Variable							
Konfig.	**TP**	**TN**	**FP**	**FN**	**Precision**	**Recall**	**F1-Score**
6	120.883	126.202	0	10.308	1	0,9214	0,9591

Durch die Einbeziehung der Krankenversichertennummer als zusätzliche Matching-Variable konnte eine maximale Precision von 1 erreicht werden (vgl. Tabelle 5.4). Dies verdeutlicht, dass keine falsch negativen Ergebnisse entstanden sind. Auch in Abbildung 5.3 wird dies deutlich, da Homonymfehler keinen messbaren Anteil ausmachen.

Der Recall liegt bei 92,14%, was darauf hinweist, dass die überwiegende Mehrheit der tatsächlich zusammengehörigen Datensätze korrekt erkannt wurde. In Abbildung 5.3 wird ersichtlich, dass lediglich 4% der Datensätze fälschlicherweise nicht als zusammengehörig identifiziert wurden, was die hohe Erkennungsrate unterstreicht.

Der F1-Score von 95,91% zeigt außerdem, dass nur ein geringfügiges Ungleichgewicht zwischen Precision und Recall besteht. Insgesamt konnte in diesem Testdurchlauf ein Anteil von 96% korrekt identifizierter Datensätze festgestellt werden, der die hohe Verknüpfungsqualität erneut bestätigt.

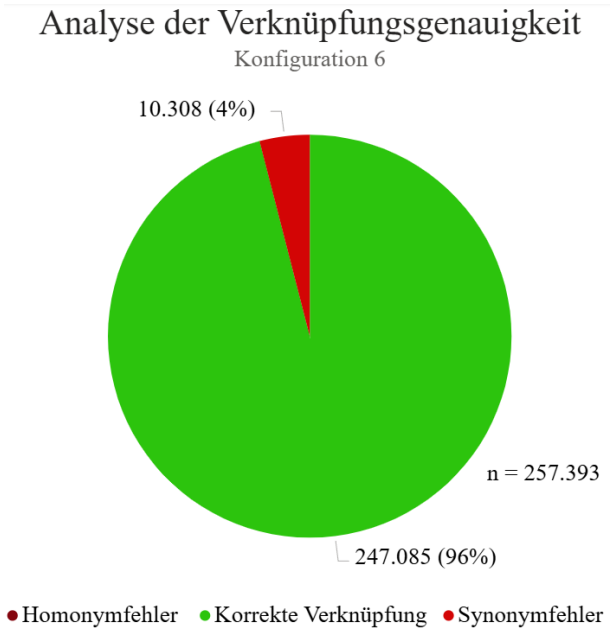


Abbildung 5.3.: Darstellung der Verteilung korrekter Verknüpfungen und Fehlerarten von Konfiguration 4 des E-PIX

5.3. Testdurchläufe HAPI FHIR MDM

Analog zu den Tests mit dem E-PIX sollte auch HAPI FHIR MDM, wie in Kapitel 4 erläutert, im Hinblick auf die Verknüpfungsqualität analysiert werden. Ziel der Untersuchung wäre es gewesen, neben einem konzeptionellen Vergleich zusätzlich die Verknüpfungsqualität von HAPI FHIR MDM zu bewerten und die Ergebnisse anhand der Metriken Precision, Recall und F1-Score zu quantifizieren. Der Testdatensatz aus dem Krebsregister Mecklenburg-Vorpommern mit 2.729.204 Datensätzen wurde bewusst in dieser Größe gewählt, um belastbare Aussagen

zur Leistungsfähigkeit der Record Linkage Lösungen treffen zu können. Für die Durchführung der Tests wurde HAPI FHIR MDM gemäß den Empfehlungen der offiziellen Dokumentation konfiguriert [3] und die Testläufe wurden auf derselben Hardwareumgebung wie die Tests des E-PIX durchgeführt, um eine direkte Vergleichbarkeit zu gewährleisten.

Während der Durchführung des Tests traten jedoch erhebliche technische Schwierigkeiten auf. Die Verarbeitung der umfangreichen Datenmenge führte zu extrem langen Laufzeiten, die teilweise mehrere Tage überschritten ,und zu abbrechenden Prozessen. Dadurch konnte die ursprünglich geplante Analyse der Verknüpfungsqualität anhand der 2.729.204 Datensätze nicht durchgeführt werden. Die zentrale Erkenntnis aus diesen Tests besteht demnach darin, dass die Verarbeitung großer Datenmengen in HAPI FHIR MDM gegenwärtig eine erhebliche Herausforderung darstellt. Im Folgenden werden daher verschiedene Szenarien vorgestellt, die mehrmals getestet wurden, um die grundsätzliche Möglichkeit zu evaluieren, Ergebnisse für die vollständigen 2.729.204 Datensätze abzurufen.

Szenario1

In einem ersten Versuch wurden 300.000 Patientenressourcen über die Schnittstelle ”POST/fhir/Patient” übertragen. Anschließend sollte die gesamte Menge der Verknüpfungsergebnisse über ”GET /fhir/$mdm-query-links” abgerufen werden (siehe Abbildung 5.4).

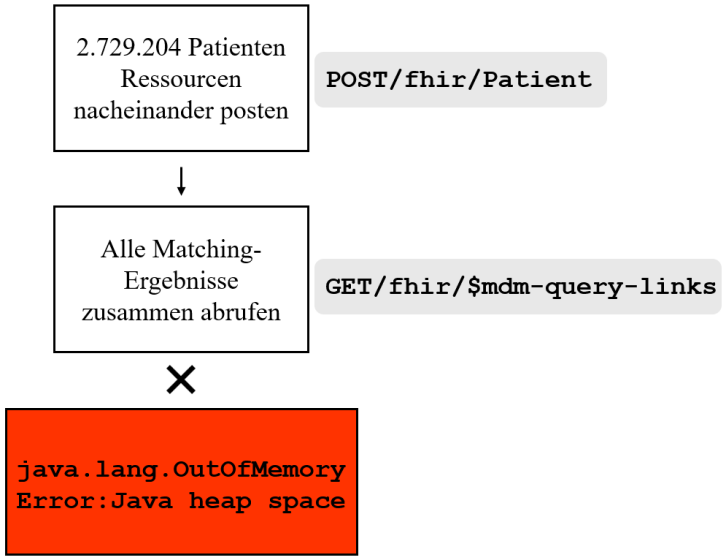


Abbildung 5.4.: Darstellung von Szenario 1

Dabei zeigte sich, dass mit einem Heap von 8 GB ein vollständiger Abruf der 300.000 Ergebnisse nicht möglich war, denn während des Testdurchlaufs trat die Fehlermeldung ”java.lang.OutOfMemoryError:Java heap space” auf, was darauf hinweist, dass der für die Java Virtual Machine (JVM) verfügbare ”Heap-Speicher” vollständig erschöpft war (siehe Abbildung 5.5). Der ”Heap” ist der Speicherbereich, in dem Java Objekte im Arbeitsspeicher ablegt.

Ist dieser Speicher aufgebraucht, können keine weiteren Objekte erstellt werden, was letztlich zum Abbruch der laufenden Verarbeitung geführt hat.

```
java.util.concurrent.ExecutionException: java.lang.OutOfMemoryError: Java heap space
        at java.base/java.util.concurrent.FutureTask.report(FutureTask.java:122)
        at java.base/java.util.concurrent.FutureTask.get(FutureTask.java:191)
        at org.apache.catalina.core.ContainerBase.threadStart(ContainerBase.java:1097)
        at org.apache.catalina.core.ContainerBase$ContainerBackgroundProcessorMonitor.run(ContainerBase.java:1141)
        at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:572)
        at java.base/java.util.concurrent.FutureTask.runAndReset(FutureTask.java:358)
        at java.base/java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:305)
        at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1144)
        at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:642)
        at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:63)
        at java.base/java.lang.Thread.run(Thread.java:1583)
Caused by: java.lang.OutOfMemoryError: Java heap space

2025-09-29T18:58:15.287+02:00 ERROR 19352 --- [         empi-2] .f.j.s.c.i.RetryingMessageHandlerWrapper : Failure 1 processing message in channel[empi]: ja
va.lang.OutOfMemoryError: Java heap space
2025-09-29T18:58:15.294+02:00 ERROR 19352 --- [         empi-2] .f.j.s.c.i.RetryingMessageHandlerWrapper : Failure
```

Abbildung 5.5.: Abbruch der Abfrage von mdm Links

Der Server, auf dem die Tests mit dem vollständigen Datensatz von 2.729.204 Patientendatensätzen durchgeführt werden sollten, verfügt zwar über insgesamt 20 GB Arbeitsspeicher, dennoch ist bei einer hochgerechneten Ergebnismenge von mehreren Millionen Datensätzen auch dieser Speicher nicht ausreichend, um die Verarbeitung ohne einen Abbruch durchzuführen. Insbesondere im Kontext von HAPI FHIR MDM ist die Speicherbelastung sehr hoch, da für jeden Patientendatensatz umfangreiche Objekte erzeugt, verglichen und zwischengespeichert werden müssen, um mögliche Duplikate zu identifizieren und die Master-Data-Management-Logik korrekt auszuführen. Bei einer sehr großen Datenmenge summiert sich diese Last nahezu exponentiell, da jeder neue Datensatz nicht nur isoliert verarbeitet, sondern auch in Relation zu bereits bestehenden Datensätzen abgeglichen werden muss. Dies führt zu einem schnellen Anstieg des Speicherverbrauchs. Um nicht alle Testergebnisse ingesamt abzurufen, wird auf der offiziellen Seite von HAPI FHIR MDM empfohlen, die Abfrage in kleinere Einheiten aufzuteilen und schrittweise zu verarbeiten, um einen stabilen Betrieb sicherzustellen [3].

Szenario 2

In einem zweiten Szenario wurde das Abrufen der Ergebnisse demnach an die Empfehlungen angepasst. Erneut wurden die Patientenressourcen über die Schnittstelle ”POST/fhir/Patient” übertragen, während parallel versucht wurde, die Ergebnisse portioniert über die Parameter ”_offset” und ”_count” abzurufen. Dabei hat sich deutlich gezeigt, dass dieses Vorgehen zu inkonsistenten Ergebnissen führt und für große Datenmengen, wie sie in diesem Testszenario vorliegen, ungeeignet ist (vgl. Abbildung 5.6).

Die Mechanik von ”_offset” und ”_count” in HAPI FHIR orientiert sich an der klassischen Datenbanklogik. Mit ”_count” wird gesteuert, wie viele Datensätze pro Seite zurückgegeben werden sollen und mit ”_offset” wird angegeben, wie viele Datensätze am Anfang der Ergebnismenge übersprungen werden, bevor die Ausgabe beginnt. Eine Abfrage mit ”offset=200” und ”count=100” liefert also die Datensätze 201 bis 300 des Gesamtergebnisses.

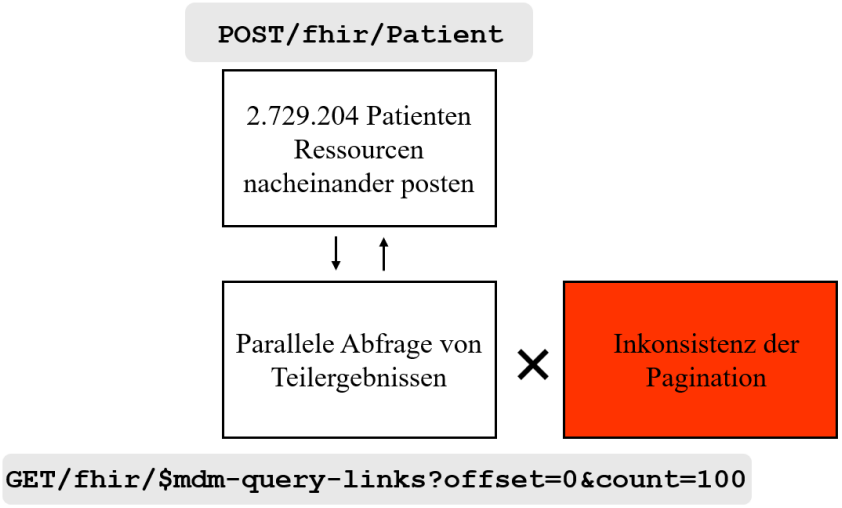


Abbildung 5.6.: Darstellung von Szenario 2

Die inkonsistente Pagination ist entstanden, weil sich der Datenbestand während der laufenden Abfragen verändert hat, da parallel kontinuierlich neue Patientendatensätze geladen wurden. Bei jeder neuen Abfrage wurde die Anzahl der übersprungenen Datensätze anhand des aktuellen Datenbestands bestimmt, wurden jedoch in der Zwischenzeit Datensätze hinzugefügt oder verändert, verschob sich die Reihenfolge. Infolgedessen wurden bestimmte Datensätze doppelt ausgegeben, während andere vollständig übersprungen wurden. Dieses Verhalten führte demnach zu der inkonsistenten Pagination, da die Ergebnisse der einzelnen Seiten nicht mehr lückenlos und eindeutig reproduzierbar waren.

Szenario 3

Um dieses Problem zu umgehen, wurde ein drittes Szenario getestet. Zunächst wurden erneut die Patientendatensätze in Form von FHIR-Patientressourcen übertragen, anschließend erfolgte das seitenweise Abrufen der Ergebnisse unter Verwendung der Parameter ”_offset” und ”_count” (vgl. Abbildung 5.7).


```
{
 "resourceType": "Parameters",
 "parameter": [ {
  "name": "prev",
  "valueUri": "http://localhost:8080/fhir/$mdm-query-links?resourceType=Patient&_offset=0&_count=100"
 }, {
  "name": "self",
  "valueUri": "http://localhost:8080/fhir/$mdm-query-links?resourceType=Patient&_offset=100&_count=100"
 }, {
  "name": "next",
  "valueUri": "http://localhost:8080/fhir/$mdm-query-links?resourceType=Patient&_offset=200&_count=100"
 }, {
```


Abbildung 5.7.: Darstellung der Abfrage in Szenario 3

In diesem Szenario lässt sich zwar die Problematik der inkonsistenten Pagination beheben, da sich der Datenbestand während der Abfragen nicht mehr verändert, dennoch bleibt die Methode zeitaufwendig und ineffizient. Der Kern des Problems liegt in der Funktionsweise von ”_offset”. Bei jeder Abfrage mit einem bestimmten offset muss die Datenbank intern so viele Zeilen überspringen, wie im Parameter angegeben, bevor die gewünschten Datensätze zurückgegeben werden können. Eine Abfrage mit ”_offset=0” und ”_count=100” ist schnell, da lediglich die ersten 100 Datensätze gelesen werden. Bei einer Abfrage mit ”_offset=1.000.000” und ”_count=100” muss die Datenbank hingegen die ersten 1.000.000 Datensätze sequentiell durchlaufen, um die 100 Datensätze ab Position 1.000.001 auszugeben. Dieser Überspringvorgang muss bei jeder neuen Abfrage erneut durchgeführt werden, was die Performance erheblich beeinträchtigt hat (vgl. Abbildung 5.8).

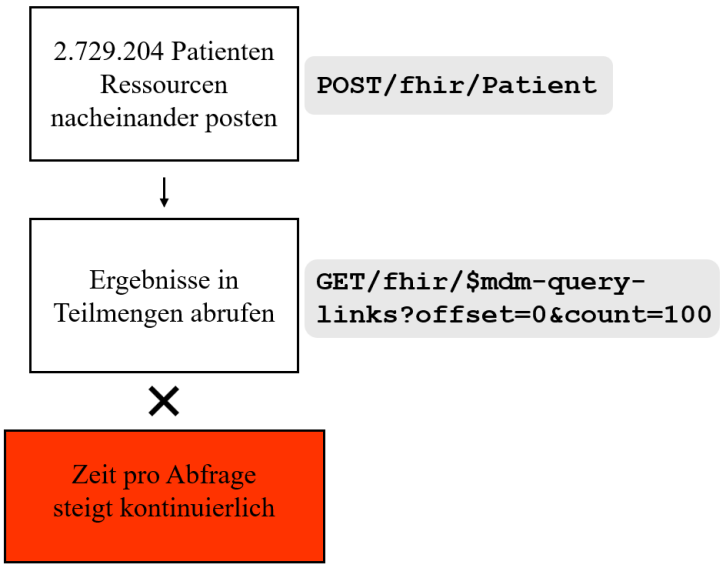


Abbildung 5.8.: Darstellung von Szenario 3

Bei einem Datenbestand von 2.729.204 Patientendatensätzen und einer Seitengröße von ”_count=100”, um einen Abbruch zu verhindern, wären rund 27.292 Einzelabfragen erforderlich. Die Abfragen verlangsamte mit zunehmendem ”_offset” erheblich. Die Bearbeitungszeit pro Abfrage steigt somit kontinuierlich an, sodass insbesondere die letzten Seiten extrem viel Zeit benötigt. Dies bedeutet, dass sich die Gesamtverarbeitungszeit nicht nur linear mit der Datensatzmenge erhöht, sondern durch den wachsenden Offset zusätzlich stark ansteigt.

Für eine belastbare Bewertung der Verknüpfungsqualität sind jedoch sehr große Datensätze erforderlich. Die vorliegenden Tests zeigen, dass HAPI FHIR MDM unter den hier verfügbaren Rahmenbedingungen (2,7 Millionen Patientendatensätze bei verfügbarer Hardware) an seine Leistungsgrenzen stößt. Neben diesen technischen Limitierungen stellt insbesondere die unzureichende Dokumentation ein gravierendes Problem dar. Diese führte in der praktischen Anwendung wiederholt zu Schwierigkeiten, die weder mit den offiziell bereitgestellten Informationen noch mit den vorhandenen Beispielen gelöst werden können. Zwar vermittelt die Dokumenta-

tion einen Überblick über das konzeptionelle Modell und beschreibt grundlegende Funktionen, für die Umsetzung in reale Szenarien fehlen jedoch entscheidende Detailangaben. Ein Beispiel ist der Umgang mit großen Datenmengen. Offiziell wird empfohlen, die Abfrage der Matching-Ergebnisse über die Parameter ”offset” und ”count” zu paginieren, in der praktischen Anwendung zeigte sich jedoch, dass diese Paginierung inkonsistente Ergebnisse lieferte und die Laufzeit extrem erhöht.

Weitere Lücken haben sich bei der Beschreibung der Konfigurationsoptionen gezeigt. So wird etwa die Levenshtein-Distanz als verfügbarer Algorithmus aufgeführt. Tatsächlich ist dieser in HAPI FHIR MDM aber nicht implementiert und somit standardmäßig nicht nutzbar. Erst durch die Implementierung einer eigenen Java-Klasse konnte dieser Mechanismus in den Tests verwendet werden. An dieser Stelle wäre eine eindeutige Kennzeichnung, welche Algorithmen tatsächlich implementiert und nutzbar sind und welche durch Eigenentwicklungen ergänzt werden müssen, wünschenswert. Eine vergleichbare Lücke in der Dokumentation besteht bei den Konfigurationsmöglichkeiten für Schwellenwerte. Nur ein Teil der Algorithmen erlaubt Anpassungen, was in den Testläufen wiederholt zu Abbrüchen führte. Auch hier fehlt eine klare Kennzeichnung der jeweiligen Algorithmen. Die fehlende Transparenz erschwert zudem das Verständnis der Abgleichlogik. So ließ sich erst durch praktische Tests mit kleineren Datensätzen und durch direkten Austausch mit den Entwicklern rekonstruieren, dass Ressourcen vermutlich dann dieselbe ”GoldenID” erhalten, wenn sie als ”Match” oder ”Possible Match” identifiziert werden. In der Dokumentation selbst wird lediglich erwähnt, dass Quellressourcen mit Goldenen Ressourcen verknüpft werden, ohne die konkrete Rolle von ”SourceID” und ”GoldenID” zu erläutern. Gerade für die Analyse und Bewertung der Verknüpfung von Datensätzen ist diese Information jedoch von zentraler Bedeutung. Um hier Klarheit zu schaffen, war der direkte Kontakt zu einem Softwareentwickler von Smile Digital Health im Mai 2025 erforderlich, da auch eine Anfrage über die offizielle Google Group [67] unbeantwortet blieb. Problematisch ist zudem die Dokumentation der Methode ”queryLinks”, die dazu dient, bestehende MDM-Verknüpfungen zwischen Ressourcen abzufragen. Die Hinweise zu den Parametern der Methode sind hier sehr knapp gehalten. Eine umfassende Dokumentation existiert nicht, sodass sich ihre Funktionsweise weitgehend nur durch praktische Tests erschließen lässt [68].

Diese Erfahrungen mit HAPI FHIR MDM zeigen, dass die Record-Linkage-Lösung in ihrer aktuellen Dokumentation und unter den gegebenen Rahmenbedingungen nur eingeschränkt zuverlässig für große Datensätze eingesetzt werden kann. Eine korrekte Nutzung erfordert daher eigene Tests mit kleineren Datenmengen sowie Rücksprachen mit den Entwicklern.

Testdurchlauf

Um trotz der beschriebenen Einschränkungen zumindest eine exemplarische Auswertung der Verknüpfungsqualität vorzunehmen, wurde die in Kapitel 4.2.4 vorgestellte Basisfunktion von

HAPI FHIR MDM für einen Testdurchlauf herangezogen (siehe Anhang E.1). Dabei kamen die Matching-Variablen ”Vorname”, ”Nachname”, ”Geschlecht” und ”Geburtsdatum” zum Einsatz, deren Ähnlichkeit mithilfe der Levenshtein-Distanz ermittelt wurde. Auf diese Weise konnten insgesamt 159 Ergebnisse abgerufen werden (siehe Tabelle 5.5).

Es ist an dieser Stelle jedoch zu betonen, dass diese Anzahl von Ergebnissen keine belastbare Aussage über die Verknüpfungsqualität erlaubt. Die Darstellung der Resultate erfolgt daher vor allem, um zu verdeutlichen, dass die Vorgehensweise von HAPI FHIR MDM prinzipiell zu Ergebnissen führt, auch wenn unter den gegebenen Rahmenbedingungen keine aussagekräftige Bewertung möglich war. Demnach war es auch nicht möglich, die ermittelten Ergebnisse der Verknüpfungsqualität inhaltlich gegenüber den Resultaten von E-PIX zu stellen.

Tabelle 5.5.: Ergebnisse des Testdurchlaufs 1, verwendete Abkürzungen: Konfig - Konfiguration, TP – True Positives, FP – False Positives, TN – True Negatives, FN – False Negatives

Nutzung der Basiskonfigurationen							
Konfig.	**TP**	**TN**	**FP**	**FN**	**Precision**	**Recall**	**F1-Score**
1	108	24	0	27	1	0,80	0,8889

In den 159 abgerufenen Ergebnissen wurde, wie in Tabelle 5.5 ersichtlich, eine Precision von 1 erreicht. Dies bedeutet, dass keine falsch positiven Verknüpfungen entstanden sind. Dieses Ergebnis wird auch in Abbildung 5.9 deutlich, da kein Anteil an Homonymfehlern vorliegt. Der Recall liegt in diesem Testdurchlauf bei 80%, was darauf hinweist, dass nicht alle tatsächlich zusammengehörigen Datensätze erkannt und korrekt verknüpft wurden. Wie in Abbildung 5.9 dargestellt, sind 27 Synonymfehler aufgetreten, was einem Anteil von 16,98% entspricht. Dennoch zeigt sich, dass mit einem Anteil von 83,02% die Mehrheit der Datensätze korrekt verknüpft wurde, der daraus berechnete F1-Score liegt bei 88,89% und verdeutlicht, dass ein insgesamt ausgewogenes Verhältnis zwischen Precision und Recall vorliegt.

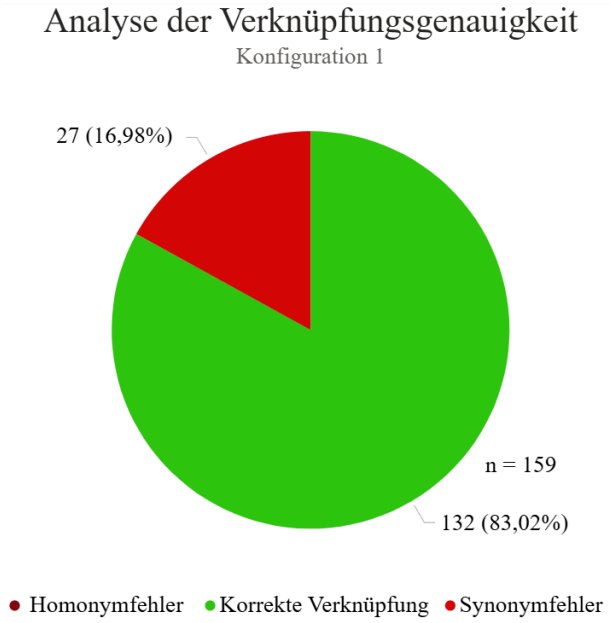


Abbildung 5.9.: Verteilung korrekter Verknüpfungen und Fehlerarten in Konfiguration 1 von HAPI FHIR MDM

6. Fazit

6.1. Zusammenfassung der Ergebnisse

Das erste Ziel dieser Arbeit war eine Gegenüberstellung der funktionalen und konzeptionellen Aspekte zwischen dem E-PIX und HAPI FHIR MDM. Ziel 1 wurde erreicht, indem sowohl die vorhandene Dokumentation als auch die Ergebnisse praktischer Tests mit kleinen Datensätzen ausgewertet wurden. Die Analyse hat gezeigt, dass beide Konzepte die wesentlichen Komponenten zur Durchführung eines Record Linkage Prozesses enthalten. Dazu zählen die Identitätsverwaltung, Mechanismen zur Datenvorverarbeitung, eine Klassifizierung nach Matching-Typen, Blocking-Verfahren, Konfigurationsmöglichkeiten für den Umgang mit Multiple-Value-Feldern, Optionen für den manuellen Abgleich potenzieller Matches, verfügbare Matching-Algorithmen sowie Funktionen zur Verwaltung der Verknüpfungen. Damit verfügen beide Konzepte grundsätzlich über die notwendigen Bausteine, um das Record Linkage umzusetzen. Deutlich wurde jedoch, dass die Unterschiede weniger in der grundsätzlichen Funktionsabdeckung, sondern vielmehr im Umfang und in der Ausgestaltung der Konfigurationsoptionen liegen. Insgesamt bietet E-PIX in den untersuchten Bereichen mehr Anpassungsmöglichkeiten als HAPI FHIR MDM, wodurch eine flexiblere Feinabstimmung des Matching-Prozesses möglich ist. Hervorzuheben ist, dass HAPI FHIR MDM jedoch eine breitere Auswahl an Matching-Algorithmen bereitstellt.

Vorteile ergeben sich beim E-PIX nicht nur in Bezug auf die Flexibilität, sondern auch hinsichtlich der Anwendbarkeit. Durch die vorhandene Weboberfläche und der Möglichkeit individuelle Konfigurationen für verschiedene Projekte zu nutzen, können Konfigurationen vergleichsweise benutzerfreundlich erstellt werden, auch durch Anwender ohne tiefgehendes technisches Vorwissen. HAPI FHIR MDM setzt hingegen ausschließlich auf eine Konfiguration über JSON-Dateien, was ein höheres Maß an technischer Kompetenz erfordert und die Benutzerfreundlichkeit einschränkt. Darüber hinaus wirkt sich die Qualität der Dokumentation auf die praktische Anwendbarkeit aus. Während für den E-PIX das Anwenderhandbuch der unabhängigen Treuhandstelle der Universitätsmedizin Greifswald eine verlässliche Grundlage geboten hat, zeigte die Dokumentation von HAPI FHIR MDM auf der offiziellen Website verschiedene Lücken, die erst durch praktische Tests oder Rückfragen bei den Entwicklern geschlossen werden konnten. Gleichzeitig muss jedoch berücksichtigt werden, dass die größere Vielfalt an Konfigurationsmöglichkeiten beim E-PIX auch eine entsprechend geschulte Person für die Konfiguration voraussetzt. Demgegenüber erfordert HAPI FHIR MDM durch den regelbasierten, deterministischen Ansatz weniger Schulungsaufwand. Dies geht jedoch zulasten der Feinabstimmung, die beim E-PIX mehr Spielraum für eine präzisere Konfiguration bietet.

Das zweite Ziel dieser Arbeit war eine Analyse der Verknüpfungsqualität von E-PIX und HAPI FHIR MDM. Ziel 2 dieser Arbeit konnte nicht im vollen Umfang erreicht werden, da in den Testdurchläufen mit HAPI FHIR MDM Schwierigkeiten bei der Abfrage großer Datenmengen sowie eine unzureichende Dokumentation der hierfür vorgesehenen Möglichkeiten auftraten. Während es im E-PIX möglich war, die Verknüpfungsqualität auf Basis des gesamten Datensatzes von 2.729.204 Patientendatensätzen aus dem Krebsregister Mecklenburg-Vorpommern zu bestimmen, zeigten die Tests mit HAPI FHIR MDM, dass die Verarbeitung und Abfrage umfangreicher Mengen an verknüpften Datensätzen derzeit eine zentrale Herausforderung darstellt. Dies stellt insbesondere ein Problem für die Bewertung der Verknüpfungsqualität dar, da aussagekräftige Ergebnisse die Verarbeitung großer Testdatensätze voraussetzen.

Die Testergebnisse des E-PIX zeigen in allen Testdurchläufen eine hohe Verknüpfungsqualität von über 87% in den Metriken ”Precision”, ”Recall” und ”F1-Score”, mit Ausnahme des Durchlaufs, in dem die Funktion für Multiple-Value-Felder bewusst raus genommen wurde. Dies verdeutlicht, dass eine hohe Flexibilität in der Anpassung des Matching-Prozesses zu einer hohen Verknüpfungsqualität beitragen kann. Gleichzeitig muss berücksichtigt werden, dass die gewählten Schwellenwerte, Gewichtungen und Algorithmen einen signifikanten Einfluss auf die Ergebnisse haben. Insbesondere die Funktion für den Umgang mit Multiple-Value-Feldern hat gezeigt, dass eine derartige Erweiterung des Record Linkage Konzepts von HAPI FHIR MDM sinnvoll sein könnte, da diese die Qualität der Verknüpfungen erhöhen kann. Bei HAPI FHIR MDM konnte aufgrund der Einschränkungen bei der Abfrage sehr großer Datenmengen lediglich 159 Datensätze abgerufen und ausgewertet werden. Diese haben mit einer Precision von 1, einem Recall von 80% und einem F1-Score von 88,89% ebenfalls eine zuverlässige Verknüpfungsqualität gezeigt, jedoch ist das Ergebnis aufgrund der geringen Datenmenge nicht aussagekräftig. Die zur Verfügung stehenden Testergebnisse von HAPI FHIR MDM war schlicht zu klein, um belastbare Aussagen über die Verknüpfungsqualität zu treffen.

6.2. Diskussion

In dieser Arbeit konnte ein Testdatensatz aus dem Krebsregister Mecklenburg-Vorpommern verwendet werden, der aus Echtdaten besteht und als Referenz für den Record Linkage Prozess dienen konnte. Es ist jedoch zu beachten, dass trotz eines zuvor durchgeführten manuellen Abgleichs fehlerhafte Angaben über die tatsächlichen Verknüpfungen nicht vollständig ausgeschlossen werden können, auch wenn ein solcher Abgleich eine sehr verlässliche Grundlage für die Erstellung eines Referenzdatensatzes darstellt. Auch wenn ein allgemeiner, standardisierter Testdatensatz für die Bewertung von Record Linkage Verfahren in der wissenschaftlichen Forschung wünschenswert wäre, ist dies nicht umsetzbar, da Datensätze stets anwendungsspezifisch sind. Ein solcher Datensatz könnte daher nie vollständig gültig sein, weshalb Vergleiche mit anderen Analysen nur eingeschränkt möglich bleiben.

Außerdem hat die Konfiguration von Schwellenwerten, Gewichten und Algorithmen einen erheblichen Einfluss auf die Ergebnisse. Je nach Forschungsziel kann die Minimierung von Homonymfehler entscheidend sein, während in anderen Szenarien die Vermeidung von Synonymfehlern im Vordergrund steht. Die optimalen Parameter sind daher kontextabhängig und müssen sorgfältig abgestimmt werden. Derzeit ist eine manuelle Anpassung der Konfigurationsoptionen durch geschultes Personal erforderlich, da unsachgemäß vorgenommene Änderungen die Verknüpfungsergebnisse signifikant beeinträchtigen können. Die Testergebnisse des E-PIX zeigen zudem, dass die Verwendung eindeutiger Identifikatoren im Record Linkage Prozess sehr hilfreich ist, da durch die Verwendung der Krankenversichertennummer als weitere Matching-Variable, keine falsch positiven Verknüpfungen auftraten. Demgegenüber steht jedoch die eingeschränkte Verfügbarkeit solcher Identifikationsnummern, insbesondere in Deutschland, wo ihre Nutzung aus ethischen und rechtlichen Gründen häufig nicht gestattet ist oder zweckgebunden ist, sodass Forschung nicht explizit eingeschlossen wird.

Das Record Linkage kann bei einer hohen Verknüpfungsqualität wertvolle Erkenntnisse ermöglichen und potenzielle Fehlinterpretationen von Forschungsergebnissen verhindern. Daher ist die Evaluation bestehender Record Linkage Lösungen und der Identifikation von Möglichkeiten zur Optimierung wichtig. Ziel dieser Arbeit war deshalb, die Konzepte, die Flexibilität und die Verknüpfungsqualität der Lösungen E-PIX und HAPI FHIR MDM systematisch gegenüberzustellen. Zwar konnten konzeptionelle Unterschiede sowie die Anwendbarkeit und Flexibilität der beiden Record Linkage Lösungen analysiert und gegenübergestellt werden, jedoch haben die Testergebnisse zur Ermittlung der Verknüpfungsqualität von HAPI FHIR MDM die gegenwärtigen technischen Grenzen von HAPI FHIR MDM aufgezeigt und die zentrale Bedeutung einer umfassenden Dokumentation im Umgang mit großen Datenmengen verdeutlicht. Im Rahmen dieser Arbeit wurden verschiedene Szenarien erprobt, um eine Abfrage großer Ergebnismengen in HAPI FHIR MDM zu ermöglichen. Diese Ansätze orientierten sich jedoch ausschließlich an den in der offiziellen Dokumentation empfohlenen Verfahren. Daraus ergibt sich die weiterführende Frage, ob alternative Strategien jenseits der vorgeschlagenen Vorgehensweisen existieren, etwa durch eine angepasste Konfiguration der Speicherressourcen, die eine vollständige Abfrage aller Ergebnisse ohne deren Aufteilung in Teilmengen erlauben würden. Ob und in welchem Umfang solche Ansätze praktikabel sind, konnte im gegebenen Untersuchungsrahmen aufgrund der Fokussierung auf die dokumentierten Empfehlungen nicht überprüft werden.

6.3. Ausblick

Aus den in dieser Arbeit gewonnenen Erkenntnissen ergibt sich die Schlussfolgerung, mit dem Hersteller ”Smile Digital Health” in Kontakt zu treten, um die Ergebnisse zu teilen und gegebenenfalls Wege zu entwickeln, HAPI FHIR MDM auch mit sehr großen Datenmengen evaluieren zu können. Dies würde nicht nur eine belastbare Bewertung der Verknüpfungsqualität ermögli-

chen, sondern zugleich bestehende Lücken in der Dokumentation sichtbar machen, deren Schließung künftigen Anwendern zugutekäme. Die in dieser Arbeit erprobten empfohlenen Maßnahmen zur Paginierung führten selbst nach mehrfachen Durchläufen nicht zu einer vollständigen Ergebnisabfrage. Künftige Untersuchungen könnten daher auf dieser Erkenntnis aufbauen und alternative Ansätze wie etwa eine Erweiterung der Speicherressourcen oder die Verarbeitung in größeren Batches testen, um eine praktikable Lösung für den Abruf umfangreicher Ergebnismengen aus HAPI FHIR MDM zu identifizieren.

Darüber hinaus haben die konzeptionellen Vergleiche und die Ergebnisse zur Verknüpfungsqualität im E-PIX verdeutlicht, dass es empfehlenswert wäre, in HAPI FHIR MDM eine Funktion zur detaillierten Verarbeitung von Multiple-Value-Feldern zu implementiert, da dies zu einer präziseren Klassifizierung beitragen und die Qualität der Verknüpfungen erhöhen könnte.

Da eine gänzlich fehlerfreie Verknüpfung von Datensätzen nicht möglich ist, besteht die wesentliche Herausforderung darin, die Konfigurationsoptionen projektspezifisch so zu gestalten, dass verbleibende Fehler keine gravierenden Auswirkungen auf die Validität der Forschungsergebnisse haben. Dies setzt jedoch voraus, dass die verwendeten Record Linkage Lösungen ein hinreichend breites Spektrum an Konfigurationsoptionen bereitstellen, um den unterschiedlichen Anforderungen verschiedener Forschungsvorhaben gerecht werden zu können. Der Vergleich der Konzepte dieser Arbeit zeigt, dass sowohl E-PIX als auch HAPI FHIR MDM umfangreiche Möglichkeiten zur Konfiguration des Matching-Prozesses bereitstellen, deren Potenzial jedoch nur dann ausgeschöpft werden kann, wenn die Parameter sorgfältig und kontextgerecht konfiguriert werden. Eine zentrale Herausforderung besteht weiterhin darin, diese Anpassungen projektspezifisch vorzunehmen und dabei das Risiko zu minimieren, dass durch suboptimale Konfigurationen wertvolles Potenzial ungenutzt bleibt. Vor diesem Hintergrund wäre es für zukünftige Arbeiten lohnenswert zu prüfen, inwiefern Verfahren des maschinellen Lernens eine sinnvolle Erweiterung der bestehenden Konzepte darstellen könnten. Durch den Einsatz lernender Modelle wäre es denkbar, Konfigurationen auf Grundlage historischer Matching-Ergebnisse und zuvor gewählter Parameter zu optimieren. Solche Modelle könnten aus vergangenen Konfigurationen und deren Resultaten Muster ableiten und dadurch Empfehlungen für zukünftige Einstellungen generieren. Eine zentrale Herausforderung bestünde jedoch in der Bereitstellung ausreichender und geeigneter Trainingsdaten, um verlässliche Vorhersagen zu ermöglichen. Maschinelles Lernen würde die notwendige manuelle Anpassung nicht vollständig ersetzen, könnte jedoch als unterstützendes Werkzeug dienen, um Konfigurationsprozesse effizienter zu gestalten.

Literaturverzeichnis

[1] Lisbach B. Linguistisches Identity Matching - Paradigmenwechsel in der Suche und im Abgleich von Personendaten. 2011:1-88. Doi:https://doi.org/10.1007/978-3-8348-9791-6.

[2] Unabhängige Treuhandstelle der Universitätsmedizin Greifswald. Anwenderhandbuch: E-PIX. 2025. https://www.ths-greifswald.de/wp-content/uploads/tools/e-pix/2025-06-25_epix_handbuch_v2025.1.pdf [Zugriff: 31.01.2025 12:04].

[3] Hapi FHIR. Master Data Management (MDM); 2025. https://hapifhir.io/hapi-fhir/docs/server_jpa_mdm/mdm.html [Zugriff: 30.01.2025 17:36].

[4] HL7. FHIR; 2025. https://hl7.org/fhir/ [Zugriff: 17.03.2025 08:56].

[5] EPIX. Unabhängige Treuhandstelle Universitätsmedizin Greifswald, E-PIX Verbreitung; 2025. https://www.ths-greifswald.de/forscher/e-pix/#verbreitung [Zugriff: 30.01.2025 14:17].

[6] Unabhängige Treuhandstelle der Universitätsmedizin Greifswald. Implementation Guide FHIR-Gateway; 2025. https://www.ths-greifswald.de/wp-content/uploads/tools/fhirgw/ig/2024-3-0/ImplementationGuide-markdown-RecordLinkageundIdentittsmanagement.html [Zugriff: 08.02.2025 06:50].

[7] Dusetzina SB MAMAGLCW Tyree S. Linking Data for Health Services Research: A Framework and Instructional Guide. Agency for Healthcare Research and Quality (US) Report. 2014:1-77. https://www.ncbi.nlm.nih.gov/books/NBK253313/.

[8] Intemann T, Kaulke K, Kipker DK, Lettieri V, Stallmann C, Schmidt CO, et al. White Paper - Verbesserung des Record Linkage für die Gesundheitsforschung in Deutschland : August 2023. NFDI4Health - Nationale Forschungsdateninfrastruktur für personenbezogene Gesundheitsdaten (Projekt). 2023. Doi:10.4126/FRL01-006461895.

[9] Doidge JC, Harron KL. Reflections on modern methods: linkage error bias. International Journal of Epidemiology. 2019 Oct. Doi:10.1093/ije/dyz203.

[10] Sariyar M, Borg A, Pommerening K. Controlling false match rates in record linkage using extreme value theory. Journal of Biomedical Informatics. 2011 aug;44(4):648-54. Doi:10.1016/j.jbi.2011.02.008.

[11] March S, Antoni M, Kieschke J, Kollhorst B, Maier B, Müller G, et al. Quo vadis Datenlinkage in Deutschland? Eine erste Bestandsaufnahme. Gesundheitswesen, Georg Thieme Verlag KG. 2018 Feb;57(03):e20-31. Doi:10.1055/s-0043-125070.

[12] Bialke M, Bahls T, Havemann C, Piegsa J, Weitmann K, Wegner T, et al. MOSAIC – A

Modular Approach to Data Management in Epidemiological Studies. Methods of Information in Medicine. 2015 Jul;54(04):364-71. Doi:10.3414/ME14-01-0133.

[13] Unabhängige Treuhandstelle der Universitätsmedizin Greifswald. Broschüre, E-PIX - Record Linkage und Identitätsmanagement. 2022. https://www.ths-greifswald.de/wp-content/uploads/2022/12/e-pix_Broschuere_V06_Stand_1-Dez.pdf[Zugriff:31.01.2025 11:00].

[14] Smile Digital Health. Offizielle Website; 2025. https://www.smiledigitalhealth.com/training/fhir-essentials-with-smile? [Zugriff: 30.01.2025 07:15].

[15] ADHA. Australian Digital Health Agency; 2025. https://developer.digitalhealth.gov.au/initiatives/interoperability-and-digital-health-standards/about-digital-health-standards [Zugriff: 30.01.2025 09:08].

[16] TeW hatu Ora. Health New Zealand; 2025. https://www.tewhatuora.govt.nz/health-services-and-programmes/digital-health/data-and-digital-standards/new-zealand-fhir-registry? [Zugriff: 30.01.2025 12:33].

[17] NHS. National Health Service Vereinigtes Königreich; 2025. https://digital.nhs.uk/services/fhir-apis? [Zugriff: 30.01.2025 14:03].

[18] G7. Open Standards and Interoperability – Final Report; 2021. https://assets.publishing.service.gov.uk/media/61d82fa48fa8f50594b5930a/G7-open-standards-final-report.pdf? [Zugriff: 30.01.2025 14:13].

[19] Kim JW, Choi H, Lim Hj, Oh M, Ahn JJ. Evaluating Linkage Quality of Population-Based Administrative Data for Health Service Research. Journal of Korean Medical Science. 2024;39(14). Doi:10.3346/jkms.2024.39.e127.

[20] Kvalsvig A, Gibb S, Teng A. Linkage error and linkage bias: A guide for IDI users. University of Otago. 2019:1-32. https://vhin.co.nz/wp-content/uploads/2019/11/Linkage-error-and-linkage-bias.pdf.

[21] Katie Harron. Data linkage in medical research. BMJ Medicine. 2022;1(1):1-3. Doi:10.1136/bmjmed-2021-000087.

[22] Adelaide A, Bakker B, de Groot M, Grootheest G, van der Laan J, Smit J, et al. Record linkage in health data. CBS, Den Haag. 2014:3-64. https://www.biolink-nl.eu/public/2014%20Record%20linkage%20simulation.pdf?

[23] Eisinger-Mathason TK, Leshin J, Lahoti V, Fridsma DB, Mucaj V, Kho AN. Data linkage multiplies research insights across diverse healthcare sectors. Communications Medicine. 2025;5(1):58. Doi:10.1038/s43856-025-00769-y.

[24] Harron KL, Doidge JC, Knight HE, Gilbert RE, Goldstein H, Cromwell DA, et al. A

guide to evaluating linkage quality for the analysis of linked data. International Journal of Epidemiology. 2017 sep;46(5):1699-710. Doi:10.1093/ije/dyx177.

[25] Harron K, Doidge JC, Goldstein H. Assessing data linkage quality in cohort studies. Annals of Human Biology. 2020 feb;47(2):218-26. Doi:10.1080/03014460.2020.1742379.

[26] Bergman L, Beelen ML, Gallee MP, Hollema H, Benraadt J, van Leeuwen FE. Risk and prognosis of endometrial cancer after tamoxifen for breast cancer. The Lancet. 2000 sep;356(9233):881-7. Doi:10.1016/S0140-6736(00)02677-5.

[27] Bozkurt O, de Boer A, Grobbee DE, de Leeuw PW, Kroon AA, Schiffers P, et al. Variation in Renin-Angiotensin System and Salt-Sensitivity Genes and the Risk of Diabetes Mellitus Associated With the Use of Thiazide Diuretics. American Journal of Hypertension. 2009 may;22(5):545-51. Doi:10.1038/ajh.2009.38.

[28] Morgan CL, Currie CJ, Peters JR. Relationship between diabetes and mortality: a population study using record linkage. Diabetes care. 2000;23(8):1103-7. Doi:10.2337/diacare.23.8.1103.

[29] Harron K, Dibben C, Boyd J, Hjern A, Azimaee M, Barreto ML, et al. Challenges in administrative data linkage for research. Big Data & Society. 2017 dec;4(2):1-12. Doi:10.1177/2053951717745678.

[30] Chiu M, Lebenbaum M, Lam K, Chong N, Azimaee M, Iron K, et al. Describing the linkages of the immigration, refugees and citizenship Canada permanent resident data and vital statistics death registry to Ontario's administrative health database. BMC Medical Informatics and Decision Making. 2016 oct;16(1). Doi:10.1186/s12911-016-0375-3.

[31] Weiand S. Vergleich von Record-Linkage Methoden anhand der Mikro-Simulation eines bundesweiten Schülerregisters. GRLC Working Paper Series. 2022:1-84. Doi:10.17185/duepublico/76361.

[32] Lait AJ, Randell B. An assessment of name matching algorithms. Technical Report Series-University of Newcastle Upon Tyne Computing Science. 1996. `http://homepages.cs.ncl.ac.uk/brian.randell/Genealogy/NameMatching.pdf`.

[33] Sayers A, Ben-Shlomo Y, Blom AW, Steele F. Probabilistic record linkage. International Journal of Epidemiology. 2015 dec;45(3):954-64. Doi:10.1093/ije/dyv322.

[34] Newcombe HB, Kennedy JM, Axford SJ, James AP. Automatic Linkage of Vital Records. Science. 1959;130(3381):954-9. Doi:10.1126/science.130.3381.954.

[35] Fellegi IP, Sunter AB. A Theory for Record Linkage. Journal of the American Statistical Association. 1969;64(328):1183-210. Doi:10.1080/01621459.1969.10501049.

[36] Hernández MA, Stolfo SJ. The Merge/Purge Problem for Large Databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 1995. p. 127-38.

Doi:10.1145/223784.223807.

[37] O'Hare K, Jurek-Loughrey A, Campos C. In: A Review of Unsupervised and Semi-supervised Blocking Methods for Record Linkage; 2019. p. 79-105. Doi:10.1007/978-3-030-01872-6-<4.

[38] Blakely T, Salmond C. Probabilistic record linkage and a method to calculate the positive predictive value. International journal of epidemiology. 2002;31(6):1246-52. Doi:10.1093/ije/31.6.1246.

[39] Grannis SJ, Overhage JM, Hui S, McDonald CJ. Analysis of a probabilistic record linkage technique without human review. In: AMIA annual symposium proceedings. vol. 2003; 2003. p. 259. https://pmc.ncbi.nlm.nih.gov/articles/PMC1479910/pdf/amia2003_0259.pdf,.

[40] Zhu VJ, Overhage MJ, Egg J, Downs SM, Grannis SJ. An Empiric Modification to the Probabilistic Record Linkage Algorithm Using Frequency-Based Weight Scaling. Journal of the American Medical Informatics Association. 2009 sep;16(5):738-45. Doi:10.1197/jamia.M3186.

[41] Jaro, Matthew A. Probabilistic linkage of large public health data files. Statistics in medicine. 1995;14(5-7):491-8. Doi:10.1002/sim.4780140510.

[42] Doidge J, Christen P, Harron K. Quality assessment in data linkage. GOVUK Report UK Government Analysis Function and Office for National Statistics. 2020:1-16. https://www.gov.uk/government/publications/joined-up-data-in-government-the-future-of-data-linking-methods/quality-assessment-in-data-linkage.

[43] Winkler WE. Data cleaning methods. Technical Report, US Bureau of the Census, Washington, DC. 2003:1-6. https://www.researchgate.net/publication/2938171_Data_Cleaning_Methods.

[44] Winkler WE. Frequency-based matching in Fellegi-Sunter model of record linkage. US Census Bureau, Working Paper RR2000-06. 2000:1-13. https://www.researchgate.net/publication/228768460_Frequency-based_matching_in_the_Fellegi-Sunter_model_of_record_linkage,.

[45] Schmidtmann I, Sariyar M, Borg A, Gerold-Ay A, Heidinger O, Hense HW, et al. Quality of record linkage in a highly automated cancer registry that relies on encrypted identity data. GMS Medizinische Informatik. 2016:1-11. Doi:10.3205/mibe000164.

[46] Gomaa W, Fahmy A. A Survey of Text Similarity Approaches. international journal of Computer Applications. 2013 04;68. Doi:10.5120/11638-7118.

[47] Levenshtein VI. Binary Codes Capable of Correcting Deletions, Insertions, and Rever-

sals. Soviet Physics Doklady. 1966;10:707-10. https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf.

[48] Christen P. Data Matching-Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer; 2012. Doi:10.1007/978-3-642-31164-2.

[49] Jaccard, Paul. Étude comparative de la distribution florale dans une portion des Alpes et du Jura. 1901. Doi:10.5169/seals-266450.

[50] Raykar N, Kumbharkar DP, Jayatilal DDH. De-duplication avoidance in regional names using an approach based on pronunciation. International Journal of Advances in Electrical Engineering. 2023 Jan;4(1):10-7. Doi:10.22271/27084574.2023.v4.i1a.32.

[51] Khan ABA, Ghazanfar MS, Khan SI; IEEE. Application of phonetic encoding for analyzing similarity of patient's data: Bangladesh perspective. 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC). 2017:664-7. Doi:10.1109/R10-HTC.2017.8289046.

[52] Hampf C, Geidel L, Zerbe N, Bialke M, Stahl D, Blumentritt A, et al. Assessment of scalability and performance of the record linkage tool E-PIX® in managing multi-million patients in research projects at a large university hospital in Germany. Journal of Translational Medicine. 2020 feb;18(1):1-11. Doi:10.1186/s12967-020-02257-4.

[53] Goldberg JM, Mernitz M. Automatisiertes Record Linkage in prosopographischen Datenbeständen am Beispiel historischer Quellen Leipzigs. Zeitschrift für digitale Geisteswissenschaften – ZfdG. 2023. Doi:10.17175/2023_001_v2.

[54] March S, Andrich S, Drepper J, Horenkamp-Sonntag D, Icks A, Ihle P, et al. Gute Praxis Datenlinkage (GPD). Das Gesundheitswesen. 2019 aug;81(08/09):636-50. Doi:10.1055/a-0962-9933.

[55] Zhu Y, Matsuyama Y, Ohashi Y, Setoguchi S. When to conduct probabilistic linkage vs. deterministic linkage? A simulation study. Journal of Biomedical Informatics. 2015 aug;56:80-6. Doi:10.1016/j.jbi.2015.05.012.

[56] Tromp M, Reitsma J, Ravelli ACJ, Méray N, Bonsel G. Record linkage: making the most out of errors in linking variables. AMIA Annual Symposium proceedings. 2006:779—783. https://europepmc.org/articles/PMC1839331.

[57] Benson T, Grieve G. Principles of Health Interoperability: FHIR, HL7 and SNOMED CT. Health Information Technology Standards. 2021. Doi:10.1007/978-3-030-56883-2.

[58] International. HL7 Health Level Seven; 2025. https://www.hl7.org/index.cfm [Zugriff: 17.03.2025 07:00].

[59] Deutschland. HL7 Health Level Seven; 2025. https://hl7.de/ [Zugriff: 17.03.2025 07:33].

[60] Integrating the Healthcare Enterprise. IHE; 2025. https://www.ihe.net/ [Zugriff: 19.03.2025 16:00].

[61] Spezifikation. openEHR; 2025. https://openehr.org/specification-program/ [Zugriff: 19.03.2025 16:34].

[62] Open industry specifications m, software for e health. openEHR; 2025. https://specifications.openehr.org/ [Zugriff: 19.03.2025 16:35].

[63] Digital Imaging and Communications in Medicine. DICOM; 2025. https://www.dicomstandard.org/ [Zugriff: 17.03.2025 17:47].

[64] Foundation S. Apache JMeter; 2025. https://jmeter.apache.org/ [Zugriff: 27.06.2025 11:10].

[65] Java-Bibliothek java-string-similarity. Normalized levenshtein; 2025. https://github.com/tdebatty/java-string-similarity#normalized-levenshtein [Zugriff: 29.06.2025 13:09].

[66] Derczynski, Leon. Complementarity, F-score, and NLP Evaluation. Proceedings of the Tenth International Conference on Language Resources and Evaluation, European Language Resources Association (ELRA). 2016:261-6. https://aclanthology.org/L16-1040.

[67] Smile Digital Health. HAPI FHIR, Google Groups; 2025. https://groups.google.com/g/hapi-fhir [Zugriff: 11.05.2025 18:45].

[68] HAPI FHIR. MdmProviderDstu3Plus Methode queryLinks; 2025. https://hapifhir.io/hapi-fhir/apidocs/hapi-fhir-server-mdm/ca/uhn/fhir/mdm/provider/MdmProviderDstu3Plus.html#queryLinks(org.hl7.fhir.instance.model.api.IPrimitiveType,org.hl7.fhir.instance.model.api.IPrimitiveType,org.hl7.fhir.instance.model.api.IPrimitiveType,org.hl7.fhir.instance.model.api.IPrimitiveType,org.hl7.fhir.instance.model.api.IPrimitiveType,org.hl7.fhir.instance.model.api.IPrimitiveType,ca.uhn.fhir.rest.server.servlet.ServletRequestDetails,org.hl7.fhir.instance.model.api.IPrimitiveType) [Zugriff: 21.08.2025 13:03].

A. Ressourcentypen in HL7 FHIR

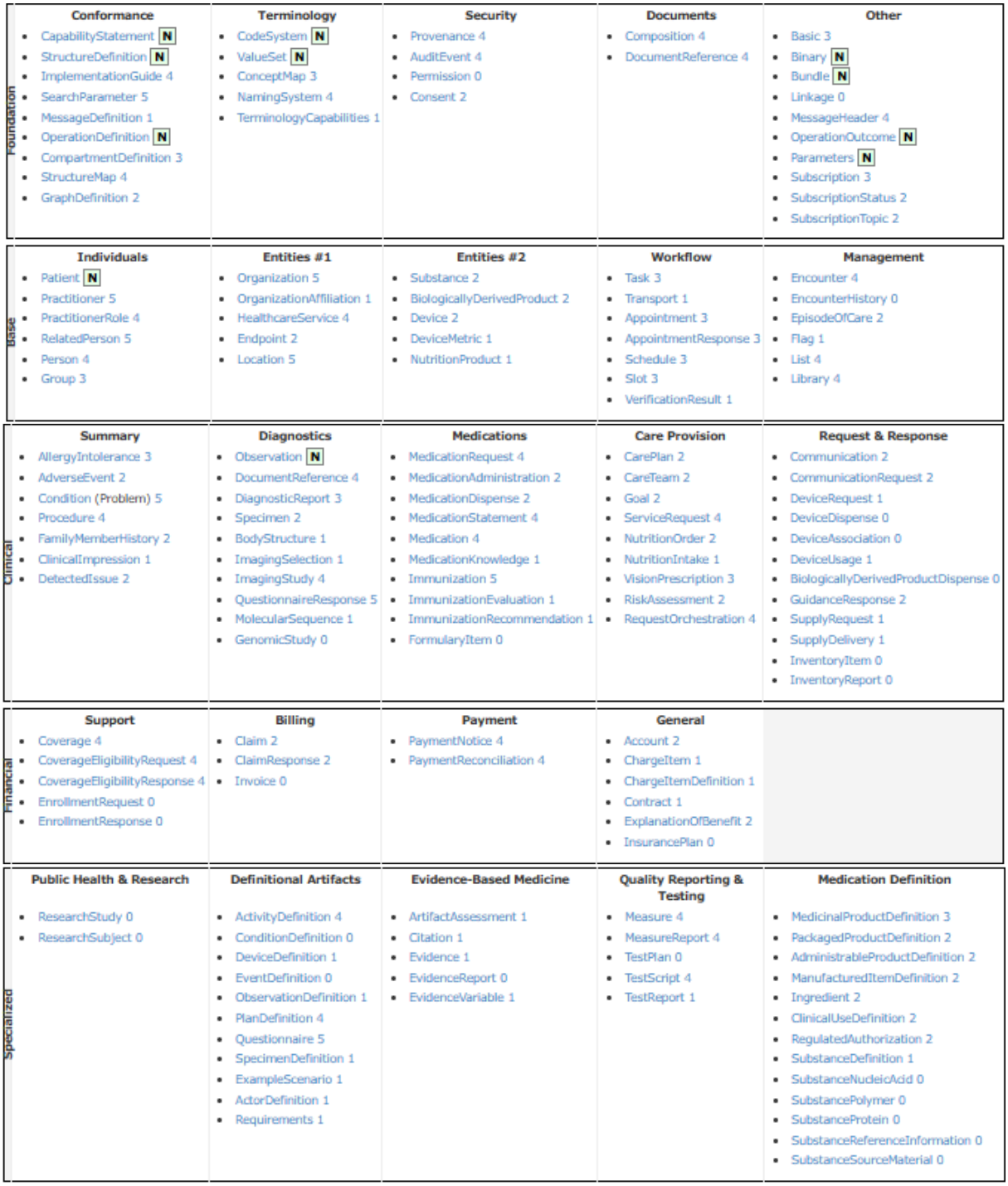


Abbildung A.1.: Darstellung der Ressourcentypen in HL7 FHIR (entnommen aus: [4])

B. Beispiel-Konfiguration E-PIX

```
<ma:MatchingConfiguration xmlns:ma="http://www.ttp.icmvc.emau.org/deduplication/config/model"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ttp.icmvc.emau.org/deduplication/config/model matching-config-2.9.0.xsd ">
  <matching-mode>MATCHING_IDENTITIES</matching-mode>
  <mpi-generator>org.emau.icmvc.ttp.epix.gen.impl.EAN13Generator</mpi-generator>
  <mpi-prefix>1001</mpi-prefix>
  <use-notifications>true</use-notifications>
  <persist-mode>IDENTIFYING</persist-mode>
  <required-fields>
    <name>firstName</name>
    <name>lastName</name>
    <name>birthDate</name>
    <name>gender</name>
  </required-fields>
  <value-fields-mapping>
    <value1>identity_person_id</value1>
    <value3>postalcode</value3>
  </value-fields-mapping>
  <preprocessing-config>
    <preprocessing-field>
      <field-name>firstName</field-name>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>æ</input-pattern>
        <output-pattern>a</output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>-</input-pattern>
        <output-pattern> </output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>œ</input-pattern>
        <output-pattern>o</output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>?</input-pattern>
        <output-pattern></output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>Dr.</input-pattern>
        <output-pattern></output-pattern>
```

```
41        </simple-transformation-type>
42        <simple-transformation-type xsi:type="ma:SimpleTransformation">
43          <input-pattern>Prof.</input-pattern>
44          <output-pattern></output-pattern>
45        </simple-transformation-type>
46        <simple-transformation-type xsi:type="ma:SimpleTransformation">
47          <input-pattern>med.</input-pattern>
48          <output-pattern></output-pattern>
49        </simple-transformation-type>
50        <simple-transformation-type xsi:type="ma:SimpleTransformation">
51          <input-pattern>rer.</input-pattern>
52          <output-pattern></output-pattern>
53        </simple-transformation-type>
54        <simple-transformation-type xsi:type="ma:SimpleTransformation">
55          <input-pattern>nat.</input-pattern>
56          <output-pattern></output-pattern>
57        </simple-transformation-type>
58        <simple-transformation-type xsi:type="ma:SimpleTransformation">
59          <input-pattern>Ing.</input-pattern>
60          <output-pattern></output-pattern>
61        </simple-transformation-type>
62        <simple-transformation-type xsi:type="ma:SimpleTransformation">
63          <input-pattern>Dipl.</input-pattern>
64          <output-pattern></output-pattern>
65        </simple-transformation-type>
66        <simple-transformation-type xsi:type="ma:SimpleTransformation">
67          <input-pattern>,</input-pattern>
68          <output-pattern></output-pattern>
69        </simple-transformation-type>
70        <simple-transformation-type xsi:type="ma:SimpleTransformation">
71          <input-pattern>-</input-pattern>
72          <output-pattern></output-pattern>
73        </simple-transformation-type>
74        <complex-transformation-type xsi:type="ma:ComplexTransformation">
75          <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.ToUpperCaseTransformation</qualified-class-
              name>
76        </complex-transformation-type>
77        <complex-transformation-type xsi:type="ma:ComplexTransformation">
78          <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.CharsMutationTransformation</qualified-class-
              name>
79        </complex-transformation-type>
80        <complex-transformation-type xsi:type="ma:ComplexTransformation">
81          <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.CharNormalizationTransformation</qualified-
              class-name>
```

```
82        </complex-transformation-type>
83      </preprocessing-field>
84      <preprocessing-field>
85        <field-name>lastName</field-name>
86        <simple-transformation-type xsi:type="ma:SimpleTransformation">
87          <input-pattern>æ</input-pattern>
88          <output-pattern>a</output-pattern>
89        </simple-transformation-type>
90        <simple-transformation-type xsi:type="ma:SimpleTransformation">
91          <input-pattern>œ</input-pattern>
92          <output-pattern>o</output-pattern>
93        </simple-transformation-type>
94        <simple-transformation-type xsi:type="ma:SimpleTransformation">
95          <input-pattern>?</input-pattern>
96          <output-pattern></output-pattern>
97        </simple-transformation-type>
98        <simple-transformation-type xsi:type="ma:SimpleTransformation">
99          <input-pattern>Dr.</input-pattern>
100         <output-pattern></output-pattern>
101       </simple-transformation-type>
102       <simple-transformation-type xsi:type="ma:SimpleTransformation">
103         <input-pattern>Prof.</input-pattern>
104         <output-pattern></output-pattern>
105       </simple-transformation-type>
106       <simple-transformation-type xsi:type="ma:SimpleTransformation">
107         <input-pattern>med.</input-pattern>
108         <output-pattern></output-pattern>
109       </simple-transformation-type>
110       <simple-transformation-type xsi:type="ma:SimpleTransformation">
111         <input-pattern>rer.</input-pattern>
112         <output-pattern></output-pattern>
113       </simple-transformation-type>
114       <simple-transformation-type xsi:type="ma:SimpleTransformation">
115         <input-pattern>nat.</input-pattern>
116         <output-pattern></output-pattern>
117       </simple-transformation-type>
118       <simple-transformation-type xsi:type="ma:SimpleTransformation">
119         <input-pattern>Ing.</input-pattern>
120         <output-pattern></output-pattern>
121       </simple-transformation-type>
122       <simple-transformation-type xsi:type="ma:SimpleTransformation">
123         <input-pattern>Dipl.</input-pattern>
124         <output-pattern></output-pattern>
125       </simple-transformation-type>
126       <simple-transformation-type xsi:type="ma:SimpleTransformation">
127         <input-pattern>,</input-pattern>
128         <output-pattern></output-pattern>
```

```
129         </simple-transformation-type>
130         <simple-transformation-type xsi:type="ma:SimpleTransformation">
131           <input-pattern>-</input-pattern>
132           <output-pattern></output-pattern>
133         </simple-transformation-type>
134         <complex-transformation-type xsi:type="ma:ComplexTransformation">
135           <qualified-class-name>org.emau.icmvc.ttp.deduplication.
                preprocessing.impl.ToUpperCaseTransformation</qualified-class-
                name>
136         </complex-transformation-type>
137         <complex-transformation-type xsi:type="ma:ComplexTransformation">
138           <qualified-class-name>org.emau.icmvc.ttp.deduplication.
                preprocessing.impl.CharsMutationTransformation</qualified-class-
                name>
139         </complex-transformation-type>
140         <complex-transformation-type xsi:type="ma:ComplexTransformation">
141           <qualified-class-name>org.emau.icmvc.ttp.deduplication.
                preprocessing.impl.CharNormalizationTransformation</qualified-
                class-name>
142         </complex-transformation-type>
143       </preprocessing-field>
144     </preprocessing-config>
145     <matching>
146       <threshold-possible-match>2.5</threshold-possible-match>
147       <threshold-automatic-match>14.5</threshold-automatic-match>
148       <use-cemfim>false</use-cemfim>
149       <parallel-matching-after>1000</parallel-matching-after>
150       <number-of-threads-for-matching>4</number-of-threads-for-matching>
151       <field>
152         <name>firstName</name>
153         <blocking-threshold>0.3</blocking-threshold>
154         <matching-threshold>0.4</matching-threshold>
155         <weight>7</weight>
156         <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
              </algorithm>
157         <multiple-values>
158           <separator> </separator>
159           <penalty-not-a-perfect-match>0.1</penalty-not-a-perfect-match>
160           <penalty-one-short>0</penalty-one-short>
161           <penalty-both-short>0.2</penalty-both-short>
162         </multiple-values>
163       </field>
164       <field>
165         <name>lastName</name>
166         <matching-threshold>0.4</matching-threshold>
167         <weight>6</weight>
168         <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
```

```
            </algorithm>
169       </field>
170       <field>
171         <name>birthDate</name>
172         <blocking-threshold>0.4</blocking-threshold>
173         <blocking-mode>NUMBERS</blocking-mode>
174         <matching-threshold>0.7</matching-threshold>
175         <weight>8</weight>
176         <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
                </algorithm>
177       </field>
178       <field>
179         <name>gender</name>
180         <blocking-threshold>1.0</blocking-threshold>
181         <matching-threshold>0.75</matching-threshold>
182         <weight>3</weight>
183         <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
                </algorithm>
184       </field>
185     </matching>
186 </ma:MatchingConfiguration>
```

Listing B.1: Beispiel einer Konfiguration im XML-Format für den E-PIX [6]

C. Beispiel-Konfiguration HAPI FHIR MDM

```
{
    "version": "1",
    "mdmTypes": ["Organization", "Patient", "Practitioner"],
    "candidateSearchParams": [
        {
            "resourceType": "Patient",
            "searchParams": ["phone"]
        },
        {
            "resourceType": "Patient",
            "searchParams": ["birthdate"]
        },
        {
            "resourceType": "*",
            "searchParams": ["identifier"]
        }
    ],
    "candidateFilterSearchParams": [
        {
            "resourceType": "Patient",
            "searchParam": "active",
            "fixedValue": "true"
        }
    ],
    "matchFields": [
        {
            "name": "birthday",
            "resourceType": "Patient",
            "resourcePath": "birthDate",
            "matcher": {
                "algorithm": "STRING"
            }
        },
        {
            "name": "phone",
            "resourceType": "Patient",
            "resourcePath": "telecom.value",
            "matcher": {
                "algorithm": "STRING"
            }
```

```
41          },
42          {
43              "name": "firstname-meta",
44              "resourceType": "Patient",
45              "fhirPath": "name.given.first()",
46              "matcher": {
47                  "algorithm": "METAPHONE"
48              }
49          },
50          {
51              "name": "lastname-meta",
52              "resourceType": "Patient",
53              "resourcePath": "name.family",
54              "matcher": {
55                  "algorithm": "METAPHONE"
56              }
57          },
58          {
59              "name": "firstname-jaro",
60              "resourceType": "Patient",
61              "resourcePath": "name.given",
62              "similarity": {
63                  "algorithm": "JARO_WINKLER",
64                  "matchThreshold": 0.8
65              }
66          },
67          {
68              "name": "lastname-jaro",
69              "resourceType": "Patient",
70              "resourcePath": "name.family",
71              "similarity": {
72                  "algorithm": "JARO_WINKLER",
73                  "matchThreshold": 0.8
74              }
75          },
76          {
77              "name": "org-name",
78              "resourceType": "Organization",
79              "resourcePath": "name",
80              "matcher": {
81                  "algorithm": "STRING"
82              }
83          }
84      ],
85      "matchResultMap": {
86          "firstname-meta,lastname-meta,birthday": "MATCH",
87          "firstname-meta,lastname-meta,phone": "MATCH",
```

```
88          "firstname-jaro,lastname-jaro,birthday": "POSSIBLE_MATCH",
89          "firstname-jaro,lastname-jaro,phone": "POSSIBLE_MATCH",
90          "lastname-jaro,phone,birthday": "POSSIBLE_MATCH",
91          "firstname-jaro,phone,birthday": "POSSIBLE_MATCH",
92          "org-name": "MATCH"
93      },
94      "eidSystems": {
95          "Organization": "https://hapifhir.org/identifier/naming/business-
                number",
96          "Practitioner": "https://hapifhir.org/identifier/naming/license-
                number"
97      }
98  }
```

Listing C.1: Beispiel-Konfiguration für HAPI FHIR MDM [3]

D. Test-Konfiguration E-PIX

```
<ma:MatchingConfiguration xmlns:ma="http://www.ttp.icmvc.emau.org/deduplication/config/model"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ttp.icmvc.emau.org/deduplication/config/model matching-config-2.9.0.xsd ">
  <matching-mode>MATCHING_IDENTITIES</matching-mode>
  <mpi-generator>org.emau.icmvc.ttp.epix.gen.impl.EAN13Generator</mpi-generator>
  <mpi-prefix>1001</mpi-prefix>
  <use-notifications>true</use-notifications>
  <persist-mode>IDENTIFYING</persist-mode>
  <required-fields>
    <name>firstName</name>
    <name>lastName</name>
    <name>birthDate</name>
    <name>gender</name>
    <name>value3</name>
  </required-fields>
  <value-fields-mapping>
    <value1>identity_person_id</value1>
    <value3>postalcode</value3>
    <value4>source_name</value4>
    <value5>date_of_diagnosis</value5>
    <value6>precision_dod</value6>
    <value7>insurance_nr</value7>
    <value8>uuid</value8>
  </value-fields-mapping>
  <preprocessing-config>
    <preprocessing-field>
      <field-name>firstName</field-name>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>æ</input-pattern>
        <output-pattern>a</output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>-</input-pattern>
        <output-pattern> </output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>œ</input-pattern>
        <output-pattern>o</output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
```

```
41          <input-pattern>?</input-pattern>
42          <output-pattern></output-pattern>
43        </simple-transformation-type>
44        <simple-transformation-type xsi:type="ma:SimpleTransformation">
45          <input-pattern>Dr.</input-pattern>
46          <output-pattern></output-pattern>
47        </simple-transformation-type>
48        <simple-transformation-type xsi:type="ma:SimpleTransformation">
49          <input-pattern>Prof.</input-pattern>
50          <output-pattern></output-pattern>
51        </simple-transformation-type>
52        <simple-transformation-type xsi:type="ma:SimpleTransformation">
53          <input-pattern>med.</input-pattern>
54          <output-pattern></output-pattern>
55        </simple-transformation-type>
56        <simple-transformation-type xsi:type="ma:SimpleTransformation">
57          <input-pattern>rer.</input-pattern>
58          <output-pattern></output-pattern>
59        </simple-transformation-type>
60        <simple-transformation-type xsi:type="ma:SimpleTransformation">
61          <input-pattern>nat.</input-pattern>
62          <output-pattern></output-pattern>
63        </simple-transformation-type>
64        <simple-transformation-type xsi:type="ma:SimpleTransformation">
65          <input-pattern>Ing.</input-pattern>
66          <output-pattern></output-pattern>
67        </simple-transformation-type>
68        <simple-transformation-type xsi:type="ma:SimpleTransformation">
69          <input-pattern>Dipl.</input-pattern>
70          <output-pattern></output-pattern>
71        </simple-transformation-type>
72        <simple-transformation-type xsi:type="ma:SimpleTransformation">
73          <input-pattern>,</input-pattern>
74          <output-pattern></output-pattern>
75        </simple-transformation-type>
76        <complex-transformation-type xsi:type="ma:ComplexTransformation">
77          <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.ToUpperCaseTransformation</qualified-class-
              name>
78        </complex-transformation-type>
79        <complex-transformation-type xsi:type="ma:ComplexTransformation">
80          <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.CharsMutationTransformation</qualified-class-
              name>
81        </complex-transformation-type>
82        <complex-transformation-type xsi:type="ma:ComplexTransformation">
83          <qualified-class-name>org.emau.icmvc.ttp.deduplication.
```

```
            preprocessing.impl.CharNormalizationTransformation</qualified-
            class-name>
84        </complex-transformation-type>
85      </preprocessing-field>
86      <preprocessing-field>
87        <field-name>lastName</field-name>
88        <simple-transformation-type xsi:type="ma:SimpleTransformation">
89          <input-pattern>æ</input-pattern>
90          <output-pattern>a</output-pattern>
91        </simple-transformation-type>
92        <simple-transformation-type xsi:type="ma:SimpleTransformation">
93          <input-pattern>œ</input-pattern>
94          <output-pattern>o</output-pattern>
95        </simple-transformation-type>
96        <simple-transformation-type xsi:type="ma:SimpleTransformation">
97          <input-pattern>?</input-pattern>
98          <output-pattern></output-pattern>
99        </simple-transformation-type>
100       <simple-transformation-type xsi:type="ma:SimpleTransformation">
101         <input-pattern>Dr.</input-pattern>
102         <output-pattern></output-pattern>
103       </simple-transformation-type>
104       <simple-transformation-type xsi:type="ma:SimpleTransformation">
105         <input-pattern>Prof.</input-pattern>
106         <output-pattern></output-pattern>
107       </simple-transformation-type>
108       <simple-transformation-type xsi:type="ma:SimpleTransformation">
109         <input-pattern>med.</input-pattern>
110         <output-pattern></output-pattern>
111       </simple-transformation-type>
112       <simple-transformation-type xsi:type="ma:SimpleTransformation">
113         <input-pattern>rer.</input-pattern>
114         <output-pattern></output-pattern>
115       </simple-transformation-type>
116       <simple-transformation-type xsi:type="ma:SimpleTransformation">
117         <input-pattern>nat.</input-pattern>
118         <output-pattern></output-pattern>
119       </simple-transformation-type>
120       <simple-transformation-type xsi:type="ma:SimpleTransformation">
121         <input-pattern>Ing.</input-pattern>
122         <output-pattern></output-pattern>
123       </simple-transformation-type>
124       <simple-transformation-type xsi:type="ma:SimpleTransformation">
125         <input-pattern>Dipl.</input-pattern>
126         <output-pattern></output-pattern>
127       </simple-transformation-type>
128       <simple-transformation-type xsi:type="ma:SimpleTransformation">
```

```
129         <input-pattern>,</input-pattern>
130         <output-pattern></output-pattern>
131       </simple-transformation-type>
132       <simple-transformation-type xsi:type="ma:SimpleTransformation">
133         <input-pattern>-</input-pattern>
134         <output-pattern></output-pattern>
135       </simple-transformation-type>
136       <complex-transformation-type xsi:type="ma:ComplexTransformation">
137         <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.ToUpperCaseTransformation</qualified-class-
              name>
138       </complex-transformation-type>
139       <complex-transformation-type xsi:type="ma:ComplexTransformation">
140         <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.CharsMutationTransformation</qualified-class-
              name>
141       </complex-transformation-type>
142       <complex-transformation-type xsi:type="ma:ComplexTransformation">
143         <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.CharNormalizationTransformation</qualified-
              class-name>
144       </complex-transformation-type>
145     </preprocessing-field>
146   </preprocessing-config>
147   <matching>
148     <threshold-possible-match>14.5</threshold-possible-match>
149     <threshold-automatic-match>14.5</threshold-automatic-match>
150     <use-cemfim>false</use-cemfim>
151     <parallel-matching-after>1000</parallel-matching-after>
152     <number-of-threads-for-matching>4</number-of-threads-for-matching>
153     <field>
154       <name>firstName</name>
155       <blocking-threshold>0.6</blocking-threshold>
156       <matching-threshold>0.8</matching-threshold>
157       <weight>7</weight>
158       <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
            </algorithm>
159       <multiple-values>
160         <separator> </separator>
161         <penalty-not-a-perfect-match>0.1</penalty-not-a-perfect-match>
162         <penalty-one-short>0</penalty-one-short>
163         <penalty-both-short>0.2</penalty-both-short>
164       </multiple-values>
165     </field>
166     <field>
167       <name>lastName</name>
168       <matching-threshold>0.8</matching-threshold>
```

```
169         <weight>8</weight>
170         <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
                </algorithm>
171       </field>
172       <field>
173         <name>birthDate</name>
174         <blocking-threshold>0.7</blocking-threshold>
175         <blocking-mode>NUMBERS</blocking-mode>
176         <matching-threshold>0.8</matching-threshold>
177         <weight>8</weight>
178         <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
                </algorithm>
179       </field>
180       <field>
181         <name>gender</name>
182         <matching-threshold>0.75</matching-threshold>
183         <weight>3</weight>
184         <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
                </algorithm>
185       </field>
186       <field>
187         <name>value3</name>
188         <matching-threshold>0.75</matching-threshold>
189         <weight>4</weight>
190                   <algorithm>org.emau.icmvc.ttp.deduplication.impl.
                        LevenshteinAlgorithm</algorithm>
191       </field>
192     </matching>
193 </ma:MatchingConfiguration>
```

Listing D.1: Standardkonfiguration (Konfiguration 1)

```
1  <ma:MatchingConfiguration xmlns:ma="http://www.ttp.icmvc.emau.org/
       deduplication/config/model"
2    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3    xsi:schemaLocation="http://www.ttp.icmvc.emau.org/deduplication/config/
         model matching-config-2.9.0.xsd ">
4    <matching-mode>MATCHING_IDENTITIES</matching-mode>
5    <mpi-generator>org.emau.icmvc.ttp.epix.gen.impl.EAN13Generator</mpi-
         generator>
6    <mpi-prefix>1001</mpi-prefix>
7    <use-notifications>true</use-notifications>
8    <persist-mode>IDENTIFYING</persist-mode>
9    <required-fields>
10     <name>firstName</name>
11     <name>lastName</name>
12     <name>birthDate</name>
13     <name>gender</name>
```

```
    <name>value3</name>
  </required-fields>
  <value-fields-mapping>
    <value1>identity_person_id</value1>
    <value3>postalcode</value3>
    <value4>source_name</value4>
    <value5>date_of_diagnosis</value5>
    <value6>precision_dod</value6>
    <value7>insurance_nr</value7>
    <value8>uuid</value8>
  </value-fields-mapping>
  <preprocessing-config>
    <preprocessing-field>
      <field-name>firstName</field-name>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>æ</input-pattern>
        <output-pattern>a</output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>-</input-pattern>
        <output-pattern> </output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>œ</input-pattern>
        <output-pattern>o</output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>?</input-pattern>
        <output-pattern></output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>Dr.</input-pattern>
        <output-pattern></output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>Prof.</input-pattern>
        <output-pattern></output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>med.</input-pattern>
        <output-pattern></output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>rer.</input-pattern>
        <output-pattern></output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
```

```
61          <input-pattern>nat.</input-pattern>
62          <output-pattern></output-pattern>
63        </simple-transformation-type>
64        <simple-transformation-type xsi:type="ma:SimpleTransformation">
65          <input-pattern>Ing.</input-pattern>
66          <output-pattern></output-pattern>
67        </simple-transformation-type>
68        <simple-transformation-type xsi:type="ma:SimpleTransformation">
69          <input-pattern>Dipl.</input-pattern>
70          <output-pattern></output-pattern>
71        </simple-transformation-type>
72        <simple-transformation-type xsi:type="ma:SimpleTransformation">
73          <input-pattern>,</input-pattern>
74          <output-pattern></output-pattern>
75        </simple-transformation-type>
76        <simple-transformation-type xsi:type="ma:SimpleTransformation">
77          <input-pattern>-</input-pattern>
78          <output-pattern></output-pattern>
79        </simple-transformation-type>
80        <complex-transformation-type xsi:type="ma:ComplexTransformation">
81          <qualified-class-name>org.emau.icmvc.ttp.deduplication.
                preprocessing.impl.ToUpperCaseTransformation</qualified-class-
                name>
82        </complex-transformation-type>
83        <complex-transformation-type xsi:type="ma:ComplexTransformation">
84          <qualified-class-name>org.emau.icmvc.ttp.deduplication.
                preprocessing.impl.CharsMutationTransformation</qualified-class-
                name>
85        </complex-transformation-type>
86        <complex-transformation-type xsi:type="ma:ComplexTransformation">
87          <qualified-class-name>org.emau.icmvc.ttp.deduplication.
                preprocessing.impl.CharNormalizationTransformation</qualified-
                class-name>
88        </complex-transformation-type>
89      </preprocessing-field>
90      <preprocessing-field>
91        <field-name>lastName</field-name>
92        <simple-transformation-type xsi:type="ma:SimpleTransformation">
93          <input-pattern>æ</input-pattern>
94          <output-pattern>a</output-pattern>
95        </simple-transformation-type>
96        <simple-transformation-type xsi:type="ma:SimpleTransformation">
97          <input-pattern>œ</input-pattern>
98          <output-pattern>o</output-pattern>
99        </simple-transformation-type>
100       <simple-transformation-type xsi:type="ma:SimpleTransformation">
101         <input-pattern>?</input-pattern>
```

```
102         <output-pattern></output-pattern>
103       </simple-transformation-type>
104       <simple-transformation-type xsi:type="ma:SimpleTransformation">
105         <input-pattern>Dr.</input-pattern>
106         <output-pattern></output-pattern>
107       </simple-transformation-type>
108       <simple-transformation-type xsi:type="ma:SimpleTransformation">
109         <input-pattern>Prof.</input-pattern>
110         <output-pattern></output-pattern>
111       </simple-transformation-type>
112       <simple-transformation-type xsi:type="ma:SimpleTransformation">
113         <input-pattern>med.</input-pattern>
114         <output-pattern></output-pattern>
115       </simple-transformation-type>
116       <simple-transformation-type xsi:type="ma:SimpleTransformation">
117         <input-pattern>rer.</input-pattern>
118         <output-pattern></output-pattern>
119       </simple-transformation-type>
120       <simple-transformation-type xsi:type="ma:SimpleTransformation">
121         <input-pattern>nat.</input-pattern>
122         <output-pattern></output-pattern>
123       </simple-transformation-type>
124       <simple-transformation-type xsi:type="ma:SimpleTransformation">
125         <input-pattern>Ing.</input-pattern>
126         <output-pattern></output-pattern>
127       </simple-transformation-type>
128       <simple-transformation-type xsi:type="ma:SimpleTransformation">
129         <input-pattern>Dipl.</input-pattern>
130         <output-pattern></output-pattern>
131       </simple-transformation-type>
132       <simple-transformation-type xsi:type="ma:SimpleTransformation">
133         <input-pattern>,</input-pattern>
134         <output-pattern></output-pattern>
135       </simple-transformation-type>
136       <simple-transformation-type xsi:type="ma:SimpleTransformation">
137         <input-pattern>-</input-pattern>
138         <output-pattern></output-pattern>
139       </simple-transformation-type>
140       <complex-transformation-type xsi:type="ma:ComplexTransformation">
141         <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.ToUpperCaseTransformation</qualified-class-
              name>
142       </complex-transformation-type>
143       <complex-transformation-type xsi:type="ma:ComplexTransformation">
144         <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.CharsMutationTransformation</qualified-class-
              name>
```

```
145       </complex-transformation-type>
146       <complex-transformation-type xsi:type="ma:ComplexTransformation">
147         <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.CharNormalizationTransformation</qualified-
              class-name>
148       </complex-transformation-type>
149     </preprocessing-field>
150   </preprocessing-config>
151   <matching>
152     <threshold-possible-match>14.5</threshold-possible-match> <!-- Possible
          Match verhindern -->
153     <threshold-automatic-match>14.5</threshold-automatic-match> <!-- 1000
          auch der Fall, wenn Felder nur matchten ohne Perfekt Match -->
154     <use-cemfim>false</use-cemfim>
155     <parallel-matching-after>1000</parallel-matching-after>
156     <number-of-threads-for-matching>4</number-of-threads-for-matching>
157     <field>
158       <name>firstName</name>
159       <blocking-threshold>0.6</blocking-threshold>
160       <matching-threshold>0.8</matching-threshold>
161       <weight>7</weight>
162       <algorithm>org.emau.icmvc.ttp.deduplication.impl.
            ColognePhoneticAlgorithm</algorithm>
163       <multiple-values>
164         <separator> </separator>
165         <penalty-not-a-perfect-match>0.1</penalty-not-a-perfect-match>
166         <penalty-one-short>0</penalty-one-short>
167         <penalty-both-short>0.2</penalty-both-short>
168       </multiple-values>
169     </field>
170     <field>
171       <name>lastName</name>
172       <blocking-threshold>0.6</blocking-threshold>
173       <matching-threshold>0.8</matching-threshold>
174       <weight>6</weight>
175       <algorithm>org.emau.icmvc.ttp.deduplication.impl.
            ColognePhoneticAlgorithm</algorithm>
176     </field>
177     <field>
178       <name>birthDate</name>
179       <blocking-threshold>0.7</blocking-threshold>
180       <blocking-mode>NUMBERS</blocking-mode>
181       <matching-threshold>0.8</matching-threshold>
182       <weight>9</weight>
183       <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
            </algorithm>
184     </field>
```

```
185     <field>
186       <name>gender</name>
187       <matching-threshold>0.75</matching-threshold>
188       <weight>3</weight>
189       <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
              </algorithm>
190     </field>
191     <field>
192       <name>value3</name>
193       <matching-threshold>0.75</matching-threshold>
194       <weight>4</weight>
195       <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
              </algorithm>
196     </field>
197   </matching>
198 </ma:MatchingConfiguration>
```

Listing D.2: Standardkonfiguration mit Kölner Phonetik für Vor- und Nachname (Konfiguration 2)

```
1  <ma:MatchingConfiguration xmlns:ma="http://www.ttp.icmvc.emau.org/
       deduplication/config/model"
2    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3    xsi:schemaLocation="http://www.ttp.icmvc.emau.org/deduplication/config/
         model matching-config-2.9.0.xsd ">
4    <matching-mode>MATCHING_IDENTITIES</matching-mode>
5    <mpi-generator>org.emau.icmvc.ttp.epix.gen.impl.EAN13Generator</mpi-
         generator>
6    <mpi-prefix>1001</mpi-prefix>
7    <use-notifications>true</use-notifications>
8    <persist-mode>IDENTIFYING</persist-mode>
9    <required-fields>
10     <name>firstName</name>
11     <name>lastName</name>
12     <name>birthDate</name>
13     <name>gender</name>
14     <name>value3</name>
15   </required-fields>
16   <value-fields-mapping>
17     <value1>identity_person_id</value1>
18     <value3>postalcode</value3>
19     <value4>source_name</value4>
20     <value5>date_of_diagnosis</value5>
21     <value6>precision_dod</value6>
22     <value7>insurance_nr</value7>
23     <value8>uuid</value8>
24   </value-fields-mapping>
25   <preprocessing-config>
```

```
26      <preprocessing-field>
27        <field-name>firstName</field-name>
28        <simple-transformation-type xsi:type="ma:SimpleTransformation">
29          <input-pattern>æ</input-pattern>
30          <output-pattern>a</output-pattern>
31        </simple-transformation-type>
32        <simple-transformation-type xsi:type="ma:SimpleTransformation">
33          <input-pattern>-</input-pattern>
34          <output-pattern> </output-pattern>
35        </simple-transformation-type>
36        <simple-transformation-type xsi:type="ma:SimpleTransformation">
37          <input-pattern>œ</input-pattern>
38          <output-pattern>o</output-pattern>
39        </simple-transformation-type>
40        <simple-transformation-type xsi:type="ma:SimpleTransformation">
41          <input-pattern>?</input-pattern>
42          <output-pattern></output-pattern>
43        </simple-transformation-type>
44        <simple-transformation-type xsi:type="ma:SimpleTransformation">
45          <input-pattern>Dr.</input-pattern>
46          <output-pattern></output-pattern>
47        </simple-transformation-type>
48        <simple-transformation-type xsi:type="ma:SimpleTransformation">
49          <input-pattern>Prof.</input-pattern>
50          <output-pattern></output-pattern>
51        </simple-transformation-type>
52        <simple-transformation-type xsi:type="ma:SimpleTransformation">
53          <input-pattern>med.</input-pattern>
54          <output-pattern></output-pattern>
55        </simple-transformation-type>
56        <simple-transformation-type xsi:type="ma:SimpleTransformation">
57          <input-pattern>rer.</input-pattern>
58          <output-pattern></output-pattern>
59        </simple-transformation-type>
60        <simple-transformation-type xsi:type="ma:SimpleTransformation">
61          <input-pattern>nat.</input-pattern>
62          <output-pattern></output-pattern>
63        </simple-transformation-type>
64        <simple-transformation-type xsi:type="ma:SimpleTransformation">
65          <input-pattern>Ing.</input-pattern>
66          <output-pattern></output-pattern>
67        </simple-transformation-type>
68        <simple-transformation-type xsi:type="ma:SimpleTransformation">
69          <input-pattern>Dipl.</input-pattern>
70          <output-pattern></output-pattern>
71        </simple-transformation-type>
72        <simple-transformation-type xsi:type="ma:SimpleTransformation">
```

```
73          <input-pattern>,</input-pattern>
74          <output-pattern></output-pattern>
75        </simple-transformation-type>
76        <simple-transformation-type xsi:type="ma:SimpleTransformation">
77          <input-pattern>-</input-pattern>
78          <output-pattern></output-pattern>
79        </simple-transformation-type>
80        <complex-transformation-type xsi:type="ma:ComplexTransformation">
81          <qualified-class-name>org.emau.icmvc.ttp.deduplication.
                preprocessing.impl.ToUpperCaseTransformation</qualified-class-
                name>
82        </complex-transformation-type>
83        <complex-transformation-type xsi:type="ma:ComplexTransformation">
84          <qualified-class-name>org.emau.icmvc.ttp.deduplication.
                preprocessing.impl.CharsMutationTransformation</qualified-class-
                name>
85        </complex-transformation-type>
86        <complex-transformation-type xsi:type="ma:ComplexTransformation">
87          <qualified-class-name>org.emau.icmvc.ttp.deduplication.
                preprocessing.impl.CharNormalizationTransformation</qualified-
                class-name>
88        </complex-transformation-type>
89      </preprocessing-field>
90      <preprocessing-field>
91        <field-name>lastName</field-name>
92        <simple-transformation-type xsi:type="ma:SimpleTransformation">
93          <input-pattern>æ</input-pattern>
94          <output-pattern>a</output-pattern>
95        </simple-transformation-type>
96        <simple-transformation-type xsi:type="ma:SimpleTransformation">
97          <input-pattern>œ</input-pattern>
98          <output-pattern>o</output-pattern>
99        </simple-transformation-type>
100       <simple-transformation-type xsi:type="ma:SimpleTransformation">
101         <input-pattern>?</input-pattern>
102         <output-pattern></output-pattern>
103       </simple-transformation-type>
104       <simple-transformation-type xsi:type="ma:SimpleTransformation">
105         <input-pattern>Dr.</input-pattern>
106         <output-pattern></output-pattern>
107       </simple-transformation-type>
108       <simple-transformation-type xsi:type="ma:SimpleTransformation">
109         <input-pattern>Prof.</input-pattern>
110         <output-pattern></output-pattern>
111       </simple-transformation-type>
112       <simple-transformation-type xsi:type="ma:SimpleTransformation">
113         <input-pattern>med.</input-pattern>
```

```
114         <output-pattern></output-pattern>
115       </simple-transformation-type>
116       <simple-transformation-type xsi:type="ma:SimpleTransformation">
117         <input-pattern>rer.</input-pattern>
118         <output-pattern></output-pattern>
119       </simple-transformation-type>
120       <simple-transformation-type xsi:type="ma:SimpleTransformation">
121         <input-pattern>nat.</input-pattern>
122         <output-pattern></output-pattern>
123       </simple-transformation-type>
124       <simple-transformation-type xsi:type="ma:SimpleTransformation">
125         <input-pattern>Ing.</input-pattern>
126         <output-pattern></output-pattern>
127       </simple-transformation-type>
128       <simple-transformation-type xsi:type="ma:SimpleTransformation">
129         <input-pattern>Dipl.</input-pattern>
130         <output-pattern></output-pattern>
131       </simple-transformation-type>
132       <simple-transformation-type xsi:type="ma:SimpleTransformation">
133         <input-pattern>,</input-pattern>
134         <output-pattern></output-pattern>
135       </simple-transformation-type>
136       <simple-transformation-type xsi:type="ma:SimpleTransformation">
137         <input-pattern>-</input-pattern>
138         <output-pattern></output-pattern>
139       </simple-transformation-type>
140       <complex-transformation-type xsi:type="ma:ComplexTransformation">
141         <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.ToUpperCaseTransformation</qualified-class-
              name>
142       </complex-transformation-type>
143       <complex-transformation-type xsi:type="ma:ComplexTransformation">
144         <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.CharsMutationTransformation</qualified-class-
              name>
145       </complex-transformation-type>
146       <complex-transformation-type xsi:type="ma:ComplexTransformation">
147         <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.CharNormalizationTransformation</qualified-
              class-name>
148       </complex-transformation-type>
149     </preprocessing-field>
150   </preprocessing-config>
151   <matching>
152     <threshold-possible-match>14.5</threshold-possible-match> <!-- Possible
          Match verhindern -->
153     <threshold-automatic-match>14.5</threshold-automatic-match> <!-- 1000
```

```
        auch der Fall, wenn Felder nur matchten ohne Perfekt Match -->
154     <use-cemfim>false</use-cemfim>
155     <parallel-matching-after>1000</parallel-matching-after>
156     <number-of-threads-for-matching>4</number-of-threads-for-matching>
157     <field>
158       <name>firstName</name>
159       <blocking-threshold>0.6</blocking-threshold>
160       <matching-threshold>0.8</matching-threshold>
161       <weight>7</weight>
162       <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
              </algorithm>
163     </field>
164     <field>
165       <name>lastName</name>
166       <matching-threshold>0.8</matching-threshold>
167       <weight>6</weight>
168       <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
              </algorithm>
169     </field>
170     <field>
171       <name>birthDate</name>
172       <blocking-threshold>0.7</blocking-threshold>
173       <blocking-mode>NUMBERS</blocking-mode>
174       <matching-threshold>0.8</matching-threshold>
175       <weight>8</weight>
176       <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
              </algorithm>
177     </field>
178     <field>
179       <name>gender</name>
180       <matching-threshold>0.75</matching-threshold>
181       <weight>3</weight>
182       <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
              </algorithm>
183     </field>
184     <field>
185       <name>value3</name>
186       <matching-threshold>0.75</matching-threshold>
187       <weight>4</weight>
188       <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
              </algorithm>
189     </field>
190   </matching>
191 </ma:MatchingConfiguration>
```

Listing D.3: Konfiguration ohne Multiple-Value-Funktion (Konnfiguration 4)

```
<ma:MatchingConfiguration xmlns:ma="http://www.ttp.icmvc.emau.org/deduplication/config/model"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ttp.icmvc.emau.org/deduplication/config/model matching-config-2.9.0.xsd ">
  <matching-mode>MATCHING_IDENTITIES</matching-mode>
  <mpi-generator>org.emau.icmvc.ttp.epix.gen.impl.EAN13Generator</mpi-generator>
  <mpi-prefix>1001</mpi-prefix>
  <use-notifications>true</use-notifications>
  <persist-mode>IDENTIFYING</persist-mode>
  <required-fields>
    <name>firstName</name>
    <name>lastName</name>
    <name>birthDate</name>
    <name>gender</name>
    <name>gender</name>
    <name>value3</name>
    <name>value7</name>
  </required-fields>
  <value-fields-mapping>
    <value1>identity_person_id</value1>
    <value3>postalcode</value3>
    <value4>source_name</value4>
    <value5>date_of_diagnosis</value5>
    <value6>precision_dod</value6>
    <value7>insurance_nr</value7>
    <value8>uuid</value8>
  </value-fields-mapping>
  <preprocessing-config>
    <preprocessing-field>
      <field-name>firstName</field-name>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>æ</input-pattern>
        <output-pattern>a</output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>-</input-pattern>
        <output-pattern> </output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>œ</input-pattern>
        <output-pattern>o</output-pattern>
      </simple-transformation-type>
      <simple-transformation-type xsi:type="ma:SimpleTransformation">
        <input-pattern>?</input-pattern>
        <output-pattern></output-pattern>
```

```
45      </simple-transformation-type>
46      <simple-transformation-type xsi:type="ma:SimpleTransformation">
47        <input-pattern>Dr.</input-pattern>
48        <output-pattern></output-pattern>
49      </simple-transformation-type>
50      <simple-transformation-type xsi:type="ma:SimpleTransformation">
51        <input-pattern>Prof.</input-pattern>
52        <output-pattern></output-pattern>
53      </simple-transformation-type>
54      <simple-transformation-type xsi:type="ma:SimpleTransformation">
55        <input-pattern>med.</input-pattern>
56        <output-pattern></output-pattern>
57      </simple-transformation-type>
58      <simple-transformation-type xsi:type="ma:SimpleTransformation">
59        <input-pattern>rer.</input-pattern>
60        <output-pattern></output-pattern>
61      </simple-transformation-type>
62      <simple-transformation-type xsi:type="ma:SimpleTransformation">
63        <input-pattern>nat.</input-pattern>
64        <output-pattern></output-pattern>
65      </simple-transformation-type>
66      <simple-transformation-type xsi:type="ma:SimpleTransformation">
67        <input-pattern>Ing.</input-pattern>
68        <output-pattern></output-pattern>
69      </simple-transformation-type>
70      <simple-transformation-type xsi:type="ma:SimpleTransformation">
71        <input-pattern>Dipl.</input-pattern>
72        <output-pattern></output-pattern>
73      </simple-transformation-type>
74      <simple-transformation-type xsi:type="ma:SimpleTransformation">
75        <input-pattern>,</input-pattern>
76        <output-pattern></output-pattern>
77      </simple-transformation-type>
78      <simple-transformation-type xsi:type="ma:SimpleTransformation">
79        <input-pattern>-</input-pattern>
80        <output-pattern></output-pattern>
81      </simple-transformation-type>
82      <complex-transformation-type xsi:type="ma:ComplexTransformation">
83        <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.ToUpperCaseTransformation</qualified-class-
              name>
84      </complex-transformation-type>
85      <complex-transformation-type xsi:type="ma:ComplexTransformation">
86        <qualified-class-name>org.emau.icmvc.ttp.deduplication.
              preprocessing.impl.CharsMutationTransformation</qualified-class-
              name>
87      </complex-transformation-type>
```

```
88        <complex-transformation-type xsi:type="ma:ComplexTransformation">
89          <qualified-class-name>org.emau.icmvc.ttp.deduplication.
                preprocessing.impl.CharNormalizationTransformation</qualified-
                class-name>
90        </complex-transformation-type>
91      </preprocessing-field>
92      <preprocessing-field>
93        <field-name>lastName</field-name>
94        <simple-transformation-type xsi:type="ma:SimpleTransformation">
95          <input-pattern>æ</input-pattern>
96          <output-pattern>a</output-pattern>
97        </simple-transformation-type>
98        <simple-transformation-type xsi:type="ma:SimpleTransformation">
99          <input-pattern>œ</input-pattern>
100         <output-pattern>o</output-pattern>
101       </simple-transformation-type>
102       <simple-transformation-type xsi:type="ma:SimpleTransformation">
103         <input-pattern>?</input-pattern>
104         <output-pattern></output-pattern>
105       </simple-transformation-type>
106       <simple-transformation-type xsi:type="ma:SimpleTransformation">
107         <input-pattern>Dr.</input-pattern>
108         <output-pattern></output-pattern>
109       </simple-transformation-type>
110       <simple-transformation-type xsi:type="ma:SimpleTransformation">
111         <input-pattern>Prof.</input-pattern>
112         <output-pattern></output-pattern>
113       </simple-transformation-type>
114       <simple-transformation-type xsi:type="ma:SimpleTransformation">
115         <input-pattern>med.</input-pattern>
116         <output-pattern></output-pattern>
117       </simple-transformation-type>
118       <simple-transformation-type xsi:type="ma:SimpleTransformation">
119         <input-pattern>rer.</input-pattern>
120         <output-pattern></output-pattern>
121       </simple-transformation-type>
122       <simple-transformation-type xsi:type="ma:SimpleTransformation">
123         <input-pattern>nat.</input-pattern>
124         <output-pattern></output-pattern>
125       </simple-transformation-type>
126       <simple-transformation-type xsi:type="ma:SimpleTransformation">
127         <input-pattern>Ing.</input-pattern>
128         <output-pattern></output-pattern>
129       </simple-transformation-type>
130       <simple-transformation-type xsi:type="ma:SimpleTransformation">
131         <input-pattern>Dipl.</input-pattern>
132         <output-pattern></output-pattern>
```

```
133         </simple-transformation-type>
134         <simple-transformation-type xsi:type="ma:SimpleTransformation">
135           <input-pattern>,</input-pattern>
136           <output-pattern></output-pattern>
137         </simple-transformation-type>
138         <simple-transformation-type xsi:type="ma:SimpleTransformation">
139           <input-pattern>-</input-pattern>
140           <output-pattern></output-pattern>
141         </simple-transformation-type>
142         <complex-transformation-type xsi:type="ma:ComplexTransformation">
143           <qualified-class-name>org.emau.icmvc.ttp.deduplication.
                preprocessing.impl.ToUpperCaseTransformation</qualified-class-
                name>
144         </complex-transformation-type>
145         <complex-transformation-type xsi:type="ma:ComplexTransformation">
146           <qualified-class-name>org.emau.icmvc.ttp.deduplication.
                preprocessing.impl.CharsMutationTransformation</qualified-class-
                name>
147         </complex-transformation-type>
148         <complex-transformation-type xsi:type="ma:ComplexTransformation">
149           <qualified-class-name>org.emau.icmvc.ttp.deduplication.
                preprocessing.impl.CharNormalizationTransformation</qualified-
                class-name>
150         </complex-transformation-type>
151       </preprocessing-field>
152     </preprocessing-config>
153     <matching>
154       <threshold-possible-match>14.5</threshold-possible-match> <!-- Possible
            Match verhindern -->
155       <threshold-automatic-match>14.5</threshold-automatic-match> <!-- 1000
          auch der Fall, wenn Felder nur matchten ohne Perfekt Match -->
156       <use-cemfim>false</use-cemfim>
157       <parallel-matching-after>1000</parallel-matching-after>
158       <number-of-threads-for-matching>4</number-of-threads-for-matching>
159       <field>
160         <name>firstName</name>
161         <blocking-threshold>0.6</blocking-threshold>
162         <matching-threshold>0.8</matching-threshold>
163         <weight>7</weight>
164         <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
              </algorithm>
165         <multiple-values>
166           <separator> </separator>
167           <penalty-not-a-perfect-match>0.1</penalty-not-a-perfect-match>
168           <penalty-one-short>0</penalty-one-short>
169           <penalty-both-short>0.2</penalty-both-short>
170         </multiple-values>
```

```
171       </field>
172       <field>
173         <name>lastName</name>
174         <matching-threshold>0.8</matching-threshold>
175         <weight>6</weight>
176         <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
                </algorithm>
177       </field>
178       <field>
179         <name>birthDate</name>
180         <blocking-threshold>0.7</blocking-threshold>
181         <blocking-mode>NUMBERS</blocking-mode>
182         <matching-threshold>0.8</matching-threshold>
183         <weight>9</weight>
184         <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
                </algorithm>
185       </field>
186       <field>
187         <name>gender</name>
188         <matching-threshold>0.75</matching-threshold>
189         <weight>3</weight>
190         <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
                </algorithm>
191       </field>
192       <field>
193         <name>value3</name>
194         <matching-threshold>0.75</matching-threshold>
195         <weight>4</weight>
196         <algorithm>org.emau.icmvc.ttp.deduplication.impl.LevenshteinAlgorithm
                </algorithm>
197       </field>
198       <field>
199         <name>value7</name>
200         <matching-threshold>1.0</matching-threshold>
201         <weight>10</weight>
202                 <algorithm>org.emau.icmvc.ttp.deduplication.impl.
                      DeterministicAlgorithm</algorithm>
203       </field>
204     </matching>
205 </ma:MatchingConfiguration>
```

Listing D.4: Konfiguration mit der Krankenversichertennummer als zusätzlichen Matching-Variable (Konfiguration 6)

E. Test-Konfiguration HAPI FHIR MDM

```
{
  "version": "1",
  "mdmTypes": ["Patient"],
  "candidateSearchParams": [
      {
         "resourceType": "Patient",
         "searchParams": ["given"]
      },
      {
         "resourceType": "Patient",
         "searchParams": ["birthdate"]
      },
      {
         "resourceType": "Patient",
         "searchParams": ["gender"]
      }
   ],
  "candidateFilterSearchParams": [],
  "matchFields": [
    {
      "name": "given-name",
      "resourceType": "Patient",
      "resourcePath": "name.given",
      "similarity": {
            "algorithm": "LEVENSCHTEIN",
            "matchThreshold": 0.8
         }
    },
    {
      "name": "last-name",
      "resourceType": "Patient",
      "resourcePath": "name.family",
      "similarity": {
            "algorithm": "LEVENSCHTEIN",
            "matchThreshold": 0.8
         }
    },
    {
      "name": "birthday",
      "resourceType": "Patient",
```

```
        "resourcePath": "birthDate",
        "similarity": {
              "algorithm": "LEVENSCHTEIN",
              "matchThreshold": 0.8
            }
      },
      {
        "name": "gender",
        "resourceType": "Patient",
        "resourcePath": "gender",
        "similarity": {
              "algorithm": "LEVENSCHTEIN",
              "matchThreshold": 0.75
            }
      },
      {
        "name": "postal-code",
        "resourceType": "Patient",
        "resourcePath": "address.postalCode",
        "similarity": {
              "algorithm": "LEVENSCHTEIN",
              "matchThreshold": 0.75
            }
      }
    ],
    "matchResultMap": {
      "given-name,last-name,birthday,postal-code" : "POSSIBLE_MATCH",
      "given-name,last-name,gender,postal-code" : "POSSIBLE_MATCH",
      "given-name,birthday,gender,postal-code" : "POSSIBLE_MATCH",
      "last-name,birthday,gender,postal-code" : "POSSIBLE_MATCH",
      "given-name,last-name,birthday,gender,postal-code" : "MATCH"
    }
}
```

Listing E.1: Standardkonfiguration (Test 1 und Test3)

```
{
  "version": "1",
  "mdmTypes": ["Patient"],
  "candidateSearchParams": [
      {
        "resourceType": "Patient",
        "searchParams": ["given"]
      },
      {
        "resourceType": "Patient",
        "searchParams": ["birthdate"]
      },
```

```
13        {
14           "resourceType": "Patient",
15           "searchParams": ["gender"]
16        }
17     ],
18     "candidateFilterSearchParams": [],
19     "matchFields": [
20        {
21           "name": "given-name",
22           "resourceType": "Patient",
23           "resourcePath": "name.given",
24           "matcher": {
25                 "algorithm": "COLOGNE"
26              }
27        },
28        {
29           "name": "last-name",
30           "resourceType": "Patient",
31           "resourcePath": "name.family",
32           "matcher": {
33                 "algorithm": "COLOGNE"
34              }
35        },
36        {
37           "name": "birthday",
38           "resourceType": "Patient",
39           "resourcePath": "birthDate",
40           "similarity": {
41                 "algorithm": "LEVENSCHTEIN",
42                 "matchThreshold": 0.8
43              }
44        },
45        {
46           "name": "gender",
47           "resourceType": "Patient",
48           "resourcePath": "gender",
49           "similarity": {
50                 "algorithm": "LEVENSCHTEIN",
51                 "matchThreshold": 0.75
52              }
53        },
54        {
55           "name": "postal-code",
56           "resourceType": "Patient",
57           "resourcePath": "address.postalCode",
58           "similarity": {
59                 "algorithm": "LEVENSCHTEIN",
```

```
            "matchThreshold": 0.75
        }
    }
  ],
  "matchResultMap": {
    "given-name,last-name,birthday,postal-code" : "POSSIBLE_MATCH",
    "given-name,last-name,gender,postal-code" : "POSSIBLE_MATCH",
    "given-name,birthday,gender,postal-code" : "POSSIBLE_MATCH",
    "last-name,birthday,gender,postal-code" : "POSSIBLE_MATCH",
    "given-name,last-name,birthday,gender,postal-code" : "MATCH"
  }
}
```

Listing E.2: Standardkonfiguration mit Kölner Phonetik für Vor- und Nachname (Test 2)

```
{
  "version": "1",
  "mdmTypes": ["Patient"],
  "candidateSearchParams": [
      {
         "resourceType": "Patient",
         "searchParams": ["given"]
      },
      {
         "resourceType": "Patient",
         "searchParams": ["birthdate"]
      },
      {
         "resourceType": "Patient",
         "searchParams": ["gender"]
      }
   ],
  "candidateFilterSearchParams": [],
  "matchFields": [
    {
      "name": "given-name",
      "resourceType": "Patient",
      "fhirPath": "name.given[0]",
      "similarity": {
            "algorithm": "LEVENSCHTEIN",
            "matchThreshold": 0.8
         }
    },
    {
      "name": "last-name",
      "resourceType": "Patient",
      "resourcePath": "name.family",
      "similarity": {
```

```
            "algorithm": "LEVENSCHTEIN",
            "matchThreshold": 0.8
        }
    },
    {
      "name": "birthday",
      "resourceType": "Patient",
      "resourcePath": "birthDate",
      "similarity": {
            "algorithm": "LEVENSCHTEIN",
            "matchThreshold": 0.8
        }
    },
    {
      "name": "gender",
      "resourceType": "Patient",
      "resourcePath": "gender",
      "similarity": {
            "algorithm": "LEVENSCHTEIN",
            "matchThreshold": 0.75
        }
    },
    {
      "name": "postal-code",
      "resourceType": "Patient",
      "resourcePath": "address.postalCode",
      "similarity": {
            "algorithm": "LEVENSCHTEIN",
            "matchThreshold": 0.75
        }
    }
  ],
  "matchResultMap": {
    "given-name,last-name,birthday,postal-code" : "POSSIBLE_MATCH",
    "given-name,last-name,gender,postal-code" : "POSSIBLE_MATCH",
    "given-name,birthday,gender,postal-code" : "POSSIBLE_MATCH",
    "last-name,birthday,gender,postal-code" : "POSSIBLE_MATCH",
    "given-name,last-name,birthday,gender,postal-code" : "MATCH"
  }
}
```

Listing E.3: Konfiguration mit Multiple-Value-Funktion (Test 4)

```
{
  "version": "1",
  "mdmTypes": ["Patient"],
  "candidateSearchParams": [
      {
```

```
            "resourceType": "Patient",
            "searchParams": ["given"]
        },
        {
            "resourceType": "Patient",
            "searchParams": ["birthdate"]
        },
        {
            "resourceType": "Patient",
            "searchParams": ["gender"]
        }
    ],
    "candidateFilterSearchParams": [],
    "matchFields": [
    {
        "name": "given-name",
        "resourceType": "Patient",
        "resourcePath": "name.given",
        "similarity": {
                "algorithm": "LEVENSCHTEIN",
                "matchThreshold": 0.8
            }
    },
    {
        "name": "given-nickname",
        "resourceType": "Patient",
        "resourcePath": "name.given",
        "matcher": {
                "algorithm": "NICKNAME"
            }
    },
    {
        "name": "last-name",
        "resourceType": "Patient",
        "resourcePath": "name.family",
        "similarity": {
                "algorithm": "LEVENSCHTEIN",
                "matchThreshold": 0.8
            }
    },
    {
        "name": "birthday",
        "resourceType": "Patient",
        "resourcePath": "birthDate",
        "similarity": {
                "algorithm": "LEVENSCHTEIN",
                "matchThreshold": 0.8
```

```
            }
        },
        {
          "name": "gender",
          "resourceType": "Patient",
          "resourcePath": "gender",
          "similarity": {
                "algorithm": "LEVENSCHTEIN",
                "matchThreshold": 0.75
            }
        },
        {
          "name": "postal-code",
          "resourceType": "Patient",
          "resourcePath": "address.postalCode",
          "similarity": {
                "algorithm": "LEVENSCHTEIN",
                "matchThreshold": 0.75
            }
        }
    ],
    "matchResultMap": {
      "given-name,last-name,birthday,postal-code" : "POSSIBLE_MATCH",
      "given-nickname,last-name,birthday,postal-code" : "POSSIBLE_MATCH",
      "given-name,last-name,gender,postal-code" : "POSSIBLE_MATCH",
      "given-nickname,last-name,gender,postal-code" : "POSSIBLE_MATCH",
      "given-name,birthday,gender,postal-code" : "POSSIBLE_MATCH",
      "given-nickname,birthday,gender,postal-code" : "POSSIBLE_MATCH",
      "last-name,birthday,gender,postal-code" : "POSSIBLE_MATCH",
      "given-name,last-name,birthday,gender,postal-code" : "MATCH",
      "given-nickname,last-name,birthday,gender,postal-code" : "MATCH"
    }
}
```

Listing E.4: Konfiguration mit eingebauter Spitznamen Abgleich-Funktion (Test 5)

```
{
  "version": "1",
  "mdmTypes": ["Patient"],
  "candidateSearchParams": [
      {
         "resourceType": "Patient",
         "searchParams": ["given"]
      },
      {
         "resourceType": "Patient",
         "searchParams": ["birthdate"]
      },
```

```
      {
         "resourceType": "Patient",
         "searchParams": ["gender"]
      }
   ],
  "candidateFilterSearchParams": [],
  "matchFields": [
    {
      "name": "given-name",
      "resourceType": "Patient",
      "resourcePath": "name.given",
      "similarity": {
            "algorithm": "LEVENSCHTEIN",
            "matchThreshold": 0.8
         }
    },
    {
      "name": "last-name",
      "resourceType": "Patient",
      "resourcePath": "name.family",
      "similarity": {
            "algorithm": "LEVENSCHTEIN",
            "matchThreshold": 0.8
         }
    },
    {
      "name": "birthday",
      "resourceType": "Patient",
      "resourcePath": "birthDate",
      "similarity": {
            "algorithm": "LEVENSCHTEIN",
            "matchThreshold": 0.8
         }
    },
    {
      "name": "gender",
      "resourceType": "Patient",
      "resourcePath": "gender",
      "similarity": {
            "algorithm": "LEVENSCHTEIN",
            "matchThreshold": 0.75
         }
    },
    {
      "name": "postal-code",
      "resourceType": "Patient",
      "resourcePath": "address.postalCode",
```

```
60          "similarity": {
61                  "algorithm": "LEVENSCHTEIN",
62                  "matchThreshold": 0.75
63              }
64      },
65      {
66        "name": "insurance-number",
67        "resourceType": "Patient",
68        "fhirPath": "identifier.where(system = 'http://fhir.de/sid/gkv/kvid
              -10').value",
69        "matcher": {
70          "algorithm": "STRING",
71            "exact": true
72       }
73      }
74    ],
75    "matchResultMap": {
76      "given-name,last-name,birthday,postal-code" : "POSSIBLE_MATCH",
77      "given-name,last-name,gender,postal-code" : "POSSIBLE_MATCH",
78      "given-name,birthday,gender,postal-code" : "POSSIBLE_MATCH",
79      "last-name,birthday,gender,postal-code" : "POSSIBLE_MATCH",
80      "given-name,last-name,birthday,gender,postal-code" : "MATCH",
81      "insurance-number" : "MATCH"
82    }
83  }
```

Listing E.5: Konfiguration mit der Krankenversichertennummer als zusätzliche Matching-Variable