Karlsruher Institut für Technologie

DHBW
Duale Hochschule
Baden-Württemberg

Entwicklung einer Registry für typgebundene Operationen auf FAIR Digital Objects

Bachelorarbeit des Studiengangs Angewandte Informatik (B. Sc.)
an der Dualen Hochschule Baden-Württemberg Karlsruhe
von

Maximilian Inckmann

2. September 2024

Bearbeitungszeitraum:	10. Juni 2024 - 02. September 2024
Kurs:	KA-TINF21B1
Dualer Partner:	Karlsruher Institut für Technologie, Eggenstein-Leopoldshafen
Betreuer des dualen Partners:	M. Sc. Nicolas Blumenröhr
Gutachter der dualen Hochschule:	Prof. Dr. Dirk Eisenbiegler

Erklärung

Ich versichere hiermit, dass ich meine Bachelorarbeit mit dem Thema: '*Entwicklung einer Registry für typgebundene Operationen auf FAIR Digital Objects*' selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt, falls beide Fassungen gefordert sind.

Karlsruhe, 2. September 2024 .....

gez. Maximilian Inckmann

Danksagung

Diese Arbeit wäre ohne die Unterstützung vieler Personen nicht möglich gewesen.

Zunächst möchte ich mich bei meinem Betreuer Nicolas Blumenröhr für die vielen Gespräche, aus denen die Idee für diese Arbeit entstanden ist, bedanken. Seine Geduld mit mir, seine stets konstruktive Kritik und die vielen Anregungen haben zur Verbesserung dieser Arbeit beigetragen. Vielen Dank für deine Zeit, deine Mühen und dein Feedback!

Ein Dankeschön geht auch an Volker Hartmann und Andreas Pfeil für die Einarbeitung, Betreuung zweier Praxisphasen, die vielen interessanten Gespräche und die Unterstützung während meines dualen Studiums. Matthias Leander-Knoll hat mir eine Praxisphase in der Abteilung Systeme und Server (SYS) ermöglicht, in welcher ich einiges über den Betrieb von Rechenzentren, die Verwaltung von Servern und Cloud-Technologien lernen durfte. Meinen Kollegen in der Abteilung DEM möchte ich für ihre offene Art, die interessanten Gespräche und die großartige Zusammenarbeit von Tag 1 an danken.

Besonders großer Dank gilt Dr. Rainer Stotzka für das entgegengebrachte Vertrauen und die Möglichkeit, mein duales Studium in seiner Abteilung am SCC zu absolvieren. Deine Wertschätzung, dein Engagement und deine immer gut gelaunte freundliche Art haben mich schwer beeindruckt und mich motiviert. Vielen Dank für die schönen drei Jahre mit dir als Vorgesetzten, Kollegen und Vorbild!

Ohne meine Eltern, ihre Ermunterung und Unterstützung wäre ich vermutlich nicht nach Karlsruhe gegangen und hätte diese wertvollen Erfahrungen nicht sammeln können. Meine Mutter kann meine Begeisterung für Informatik und Technik zwar bis jetzt noch nicht teilen, hat aber mich trotzdem immer unterstützt und hat meine, für sie vollkommen unverständlichen, Arbeiten Korrektur gelesen. Mein Vater hat mich von klein auf an die Welt der Technik herangeführt, mich immer unterstützt und meine Arbeiten gelesen, obwohl Lesen definitiv nicht zu seinen Lieblingsbeschäftigungen gehört. Vielen Dank für Alles!

Abstract

FAIR Digital Objects (FAIR-DOs) are an innovative technology for managing and processing research data according to the FAIR principles. Machine readability, interpretability, and actionability are the three development stages towards an automatable abstraction layer for various research data and technologies. To date, the typing of FAIR-DOs has been implemented using a Data Type Registry (DTR), which enables machine interpretability. In the active development of FAIR-DOs, the execution of operations on FAIR-DOs for the automated processing of data (machine actionability) is the next development step.

This work presents a concept for type-associated and technology-agnostic FAIR Digital Object Operations (FAIR-DO-Ops). The underlying conceptual data model is implemented with IDORIS as a prototype for a novel DTR, and thus enables machine-actionable FAIR-DOs. In addition to the functionality of the previous reference DTR, IDORIS allows the management of FAIR-DO-Ops, the association of FAIR-DO-Ops with data types and the mapping of inheritance mechanisms in a graph data structure as well as their validation. Previous developments and recommendations for DTRs and FAIR-DOs are taken into account and developed further. It is now possible to map machine-actionable data processing for FAIR-DOs in a typed, extensible and comprehensible way. These results are a significant contribution to state-of-the-art on FAIR-DOs and open up perspectives for further future research and development.

In summary, the conceptual data model for type-associated FAIR Digital Object Operations and its realization with IDORIS enable machine-actionable FAIR-DOs.

Zusammenfassung

FAIR Digital Objects (FAIR-DOs) sind eine innovative Technologie, um Forschungsdaten gemäß den FAIR-Prinzipien zu verwalten und zu verarbeiten. Maschinenlesbarkeit, -interpretierbarkeit und -prozessierbarkeit sind die drei Entwicklungsstufen zu einer automatisierbaren Abstraktionsschicht für diverse Forschungsdaten und Technologien. Bislang wird mithilfe einer Data Type Registry (DTR) die Typisierung von FAIR-DOs umgesetzt, was die Maschineninterpretierbarkeit ermöglicht. In der aktiven Entwicklung von FAIR-DOs ist die Ausführung von Operationen auf FAIR-DOs zur automatisierten Verarbeitung von Daten (Maschinenprozessierbarkeit) der nächste Entwicklungsschritt.

In dieser Arbeit wird ein Konzept für typgebundene und technologie-agnostische FAIR Digital Object Operations (FAIR-DO-Ops) ausgearbeitet. Das zugrundeliegende konzeptuelle Datenmodell wird mit IDORIS als Prototyp für eine neuartige DTR umgesetzt und ermöglicht somit maschinenprozssierbare FAIR-DOs. Ergänzend zur Funktionalität der bisherigen Referenz-DTR erlaubt IDORIS die Verwaltung von FAIR-DO-Ops, die Assoziation von FAIR-DO-Ops mit Datentypen und die Abbildung von Vererbungsmechanismen in einer Graphdatenstruktur sowie deren Validierung. Bisherige Entwicklungen und Empfehlungen für DTRs und FAIR-DOs werden berücksichtigt und weiterentwickelt. Es ist nun möglich maschinenprozessierbare Datenverarbeitung für FAIR-DOs typisiert, erweiterbar und nachvollziehbar abzubilden. Diese Ergebnisse sind ein wesentlicher Beitrag zur internationalen Forschung an FAIR-DOs und eröffnen Perspektiven für weiterführende zukünftige Entwicklungen.

Zusammenfassend ermöglichen das konzeptuelle Datenmodell für typgebundene FAIR Digital Object Operations und dessen Realisierung mit IDORIS maschinenprozessierbare FAIR-DOs.

Inhaltsverzeichnis

1. Motivation

Eine Voraussetzung für eine sowohl moderne, nachhaltige, als auch effiziente Forschung ist die (Wieder-)Verwendung von maschinenlesbaren, -interpretierbaren und -prozessierbaren Daten [1, 2]. Die **F**indable, **A**ccessible, **I**nteroperable und **R**eusable (FAIR)-Prinzipien sind "[...]ein Mindestmaß gemeinschaftlich vereinbarter Leitprinzipien und Praktiken[...]" [3] für eine sowohl maschinen- als auch menschennutzbare Repräsentation und Verwaltung von (Meta-)Daten. Gerade im Hinblick auf die wichtige Rolle der Digitalisierung und Automatisierung von Informationsverarbeitungsprozessen sind die FAIR-Prinzipien für die "Transparenz, Reproduzierbarkeit und Wiederverwendbarkeit" [3] essenziell für maschinengetriebene Datenanalyse [1]. Konkret bedeutet dies, dass bei der Verwaltung von (Forschungs-) Daten auf, in Abschnitt 2.1 referenzierte, Qualitätsmerkmale geachtet werden muss, um die Nachhaltigkeit und Aussagekraft der (Meta-)Daten sicherstellen zu können.

Das Konzept der "Digital Objects" [4] dient als Grundlage für die Realisierung von FAIR Data mit FAIR Digital Objects (FAIR-DOs). Das FAIR-DO-Konzept wendet verschiedene Mechanismen und Prinzipien von FAIR und der Objektorientierung an, um maschineninterpretierbare (eng. "machine-interpretable"), maschinenprozessierbare (eng."machine-actionable") und persistente Entitäten für einen "global integrierten Datenraum" [5] zu schaffen, welche optimale Voraussetzungen für Wiederverwendbarkeit und automatisierte Auswertung durch Maschinen bieten.

In der Abteilung Data Exploitation Methods (DEM) des Scientific Computing Center (SCC) am Karlsruher Institut für Technologie (KIT) wird unter anderem im Rahmen der Projekte Helmholtz Metadata Collaboration Platform (HMC) und Nationale Forschungsdaten-Infrastruktur (NFDI) an nachhaltigem Forschungsdatenmanagement geforscht. Ziel ist es, Forschungsdateninfrastrukturen sowie Metadatennutzung nach den FAIR-Kriterien [3, 5] mithilfe von FAIR-DOs aufzubauen und zu etablieren. Dazu wird sowohl in nationalen (HMC, NFDI) als auch in internationalen Forschungsgruppen und Gremien (wie z. B. der Research Data Alliance (RDA) oder dem FAIR Digital Object Forum (FDO-Forum)) an Empfehlungen mitgewirkt und entsprechende technische Lösungen entwickelt. Diese Konzepte und Lösungen werden in verschiedenen Projekten, unter anderem der digitalen Geisteswissenschaften sowie Materialwissenschaften, eingesetzt und weiterentwickelt.

Zum Zeitpunkt der Veröffentlichung sind das FAIR-DO-Konzept, das zugehörige Datenmodell sowie die technische Umsetzung noch nicht standardisiert und erlauben daher sowohl konzeptuelle als auch technische Modifikationen innerhalb des wissenschaftlichen Diskurses. Eine Betrachtung des Status quo zeigt, dass die derzeitigen (prototypischen) Systeme teilweise unausgereift sind und einige Konzepte von FAIR-DOs in Hinblick auf deren Bezug zur Objektorientierung mit diesen nicht umgesetzt werden können. Dazu zählt besonders die derzeitige Data Type Registry (DTR) [6], welche die Definition und Registrierung von Datentypen, Typprofilen und Operationen ermöglicht. Dies ist vor allem der Tatsache geschuldet, dass die zugrundeliegende Software der bisherigen DTR ein Metadatenrepositorium ist, welches JSON-Dokumente gegen anwendungsfallbezogene Schemata validiert und mit Persistent Identifier (PIDs) zugänglich macht [7]. Mit dem bisherigen System kann eine grundlegende Maschineninterpretierbarkeit im FAIR-DO-Kontext erreicht werden, sodass die Technologie bereits produktiv im Rahmen einiger Projekte verwendet wird.

Spezielle Logik für Mechanismen, wie etwa Polymorphie, Vererbung oder Attribut-Overloading existieren nicht und können nicht ohne Weiteres in das bestehende System integriert werden. Die fehlenden Konzepte, fehlenden Vererbungsmechanismen und die fehlende funktionale Erweiterbarkeit der ePIC-DTR zeigen deutlich den Bedarf für die Entwicklung einer neuartigen DTR.

Maschineninterpretierbarkeit ist eine zwingende Voraussetzung für die Realisierung der Maschinenprozessierbarkeit [2, 1], damit automatisierte Analysen von FAIR-DOs, deren Daten und möglichen Verarbeitungsmechanismen aufgrund der Typisierung möglich sind. Die Maschinenprozessierbarkeit soll mithilfe des Typisierungssystems Informationen über die auf einem FAIR-DO ausführbaren Operationen ermitteln. Mit diesen sogenannten FAIR Digital Object Operations (FAIR-DO-Ops) können Arbeitsabläufe abgebildet, nachvollzogen und ausgeführt werden. Ein Beispiel für einen FAIR-DO-Ops-Workflow beinhaltet die Extraktion einer ORCiD aus einer Zeichenkette, das Einsetzen in eine URL und die typisierte Rückgabe des Ergebnis einer HTTP-Anfrage. Dies ist nur möglich, wenn sowohl die syntaktischen als auch semantischen Informationen der FAIR-DOs und des Typisierungssystems ausreichende Aussagekraft besitzen, was jedoch momentan zumindest nicht in ausreichender maschineninterpretierbarer Form gegeben ist. Beispielsweise können derzeit keine Vererbungshierarchien abgebildet oder aufgelöst werden, sodass auch keine Erweiterungen abgebildet werden können. Außerdem existiert noch kein Mechanismus, um Datentypen oder Kernel Information Profile (KIPs) mit FAIR Digital Object Operations (FAIR-DO-Ops) zu verknüpfen oder diese zu vererben, wodurch auch keine wiederverwendbaren Verknüpfungen zwischen FAIR-DOs und FAIR-DO-Ops möglich sind. FAIR-DO-Ops sind durch das FAIR-DO-Konzept abstrakt formuliert, haben jedoch bislang weder eine "offizielle" Bezeichnung noch ein konkretes Daten- oder Ausführungsmodell. Es existieren lediglich mehrere Ideen und Ansätze, welche höchstens im Rahmen eines Proof-of-Concept (PoC) umgesetzt wurden, was ein hohes Maß an Freiheit für diese Arbeit bedeutet.

Um die oben genannten Funktionalitäten bereitzustellen, wurde ein neuartiges integriertes Datentyp- und Operations-Register mit Vererbungssystem (auf Englisch: Integrated Data Type and Operations Registry with Inheritance System (IDORIS)) entwickelt. IDORIS ist in der Lage, komplexe Umgebungen von FAIR-DOs mit ihren Datentypen, Typprofilen und Operationen im Datenmodell abzubilden, sowohl intelligente Anfragen als auch die Validierung von Eingaben mithilfe der darunterliegenden Graphdatenbank durchzuführen. Im Laufe der Entwicklung von IDORIS wurden bestehende Konzepte, Empfehlungen sowie Quasi-Standards berücksichtigt und weiterentwickelt. Zudem wurde ein Konzept für die technische Realisierung von FAIR-DO-Ops entwickelt und in das Datenmodell von IDORIS eingebunden. Bei der Entwicklung wurde die spätere Erweiterbarkeit berücksichtigt.

2. Stand der Technik

2.1. Konzept der FAIR Digital Objects

Die hohe Datenheterogenität, fehlende Schematisierung sowie unpassende, unvollständige oder nicht existierende Metadaten erschweren, dass (Forschungs-)daten außerhalb ihres Kontextes gefunden, betrachtet, verstanden und wiederverwendet werden können. Dies ist ein Problem, da bereits existierendes Wissen und bereits vorhandene Daten nicht interpretiert werden können und somit kostspielig erneut erhoben werden müssen. Der Begriff der Daten umfasst für diese Arbeit sowohl Metadaten, Nutzdaten als auch Software und andere spezifische Informationen. In Hinblick auf die Nachhaltigkeit und Effizienz der Forschung ist es daher umso wichtiger, dass Forschungsdaten und ihre Metadaten von höchster Qualität sind und auffindbar, zugreifbar, interoperabel und wiederverwendbar werden. Aus diesen Anforderungen wurden die FAIR Prinzipien entwickelt, welche eine minimale Menge von qualitativen Anforderungen und Prinzipien als Orientierungshilfe zur Realisierung eines nachhaltigen Forschungsdatenmanagements spezifizieren [1, 3, 8].

Basierend auf digitalen Objekten [4] und Konzepten der objektorientierten Programmierung setzen FAIR-DOs diese Anforderungen an FAIRe Daten um und stellen ein Konzept für die maschinenlesbare, maschineninterpretierbare und maschinenprozessierbare Organisation von (Meta-)Daten bereit [1, 5, 9]. Ziel von FAIR-DOs ist es, eine einheitliche Abstraktionsebene für (Meta-)Daten und bereits existierende Forschungsdateninfrastrukturen bereitzustellen [1]. FAIR-DOs sind eine abstrakte Repräsentation von Daten, die mithilfe von PIDs referenziert werden, zusätzliche Metadaten enthalten und durch ihre Typisierung mit ausführbaren Operationen assoziiert werden können.

FAIR-DOs verweisen daher auf externe herkömmliche Informationsquellen (z. B. (Meta-)Datenrepositorien, Archive, URLs, APIs, …), welche im FAIR-DO-Kontext aufgrund der fehlenden direkten maschineninterpretierbaren Aussagekraft oft als Bitströme bezeichnet werden, referenzieren andere FAIR-DOs und enthalten weitere Metadaten (z. B. Datumsangaben, Autoren, Lizenz, …). Diese Metadaten werden typisiert in einem persistent aufrufbaren Metadatenrecord gespeichert und mithilfe von PIDs referenziert. Typisierung geschieht sowohl für einzelne Einträge in diesen Metadatenrecords (auch: FAIR-DO-Record), als auch für die Struktur der Metadatenrecords selbst. Dadurch wird zusätzlich zu der Maschinenlesbarkeit einzelner Record-Einträge die Maschineninterpretierbarkeit von FAIR-DOs, deren Inhalte, Verknüpfung mit anderen FAIR-DOs und externen Ressourcen ermöglicht [1]. PIDs, wie beispielsweise DOIs, ISBNs oder ORCiDs, verhindern ungültige Verweise (die z. B. durch Einstellen des Services, API-Änderungen, … entstehen) zwischen FAIR-DOs und stellen somit die Semantik von (Meta-)Daten permanent und für die Nachwelt nachvollziehbar zur Verfügung. FAIR-DOs sollen selbst beim Entfernen der eigentlichen (Meta-)Daten bestehen bleiben, um die Integrität der Verweise zu garantieren und elementare Informationen bereitzustellen.

2.2. Modellierung von FAIR-DOs im ePIC Referenzsystem

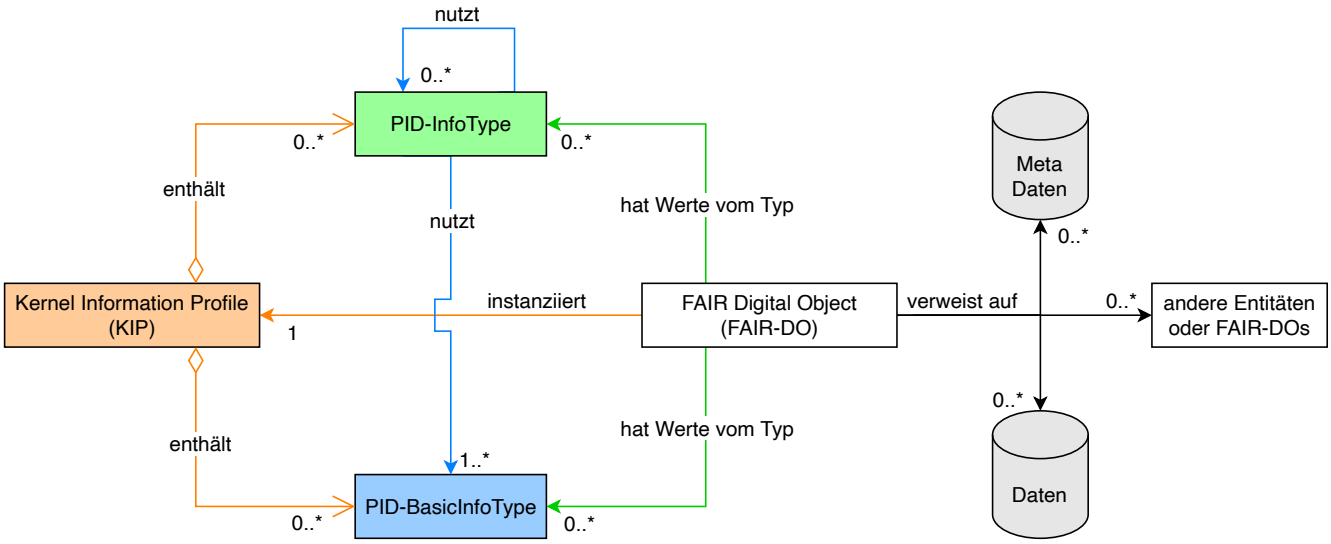

Abbildung 2.1.: Zusammenhänge im bestehenden FAIR-DO Modell

Im FDO-Forum [10] werden verschiedenste Ansätze zur Umsetzung des FAIR-DO-Konzepts diskutiert, von denen in dieser Arbeit jedoch nur ein Ansatz der RDA [11, 12, 13], welcher im Rahmen von HMC [14] näher spezifiziert und prototypisch umgesetzt wurde [15], betrachtet wird. Andere Ansätze nutzen beispielsweise Signposting [16], wo innerhalb einer Webseite auf andere (semantisch angereicherte) Uniform Resource Locators (URLs) verwiesen wird. Der "HMC- und RDA-Ansatz" setzt hingegen auf starke Persistenz der FAIR-DOs in Handle.net [17]-Records, welche jeweils durch eine global eindeutige Handle-PID referenzierbar sind. Dieses Modell mit den zusammenhängenden Komponenten ist in Abbildung 2.1 abstrakt beschrieben.

`PID-BasicInfoType` und `PID-InfoType` (zusammenfassend im Folgenden als Datentyp bezeichnet) sind "schematische Beschreibungen von Datenstrukturen" [15] und werden ebenfalls über eine PID referenziert [18]. `PID-BasicInfoTypes` repräsentieren dabei Typen für einfache Werte (z. B. Datumsformat, Format einer Handle-PID, …), die von den primitiven JSON Datentypen (number, string, boolean, …) abstammen und zusätzlich mithilfe eines regulären Ausdrucks validiert werden können. In `PID-InfoTypes` können Informationen zu Datentypen hinzugefügt (siehe blaue Pfeile in der Abbildung) oder mehrere Datentypen kombiniert werden, weshalb diese `PID-InfoTypes` (auf den Anwendungsfall) spezialisierter sind. Werte zu den Datentypen (also PID-BasicInfoType oder PID-InfoType) können primitive (Meta-)Daten, wie beispielsweise Datumsangaben, aber auch Verweise auf externe Quellen oder andere FAIR-DOs sein. Externe Quellen sind zum Beispiel (Meta-)Datenrepositorien, in denen sich die speziellen (Meta-)Daten der vom FAIR-DO abstrahierten Informationen (z. B. Bilder, Bilddatensätze, Annotationen, Sensordaten, …) befinden, ORCiD.org-Profile [19] mit persistenten Informationen über Autoren oder beliebig andere Ressourcen. Basierend auf den zugrundeliegenden Datentypen können die maschinenlesbaren Werte in den FAIR-DO-Records validiert werden.

Aussagekräftige und hochqualitative Metadaten im FAIR-DO-Record sind einer der wichtigsten Aspekte von FAIR Digital Objects [15]. Die Angaben im FAIR-DO sollten dabei sowohl einige generelle Informationen (Lizenz, Ursprung, Datumsangaben, …), als auch anwendungsfallspezifische Informationen (Verweis auf das getestete Material in den Materialwissenschaften, Messgenauigkeiten, Verweis auf das annotierte Werk in den Geisteswissenschaften, Bildformatinformationen, …) in maschinenlesbarer Form enthalten, um die Aussagekraft von FAIR-DOs sicherzustellen [15]. Dies wird mithilfe von Kernel Information Profile (KIPs) umgesetzt. Kernel Information Profile (KIPs) spezifizieren die elementaren Attribute für jedes instanziierende

FAIR-DO (siehe orangene Pfeile in der Abbildung), deren Datentyp, ob diese verpflichtend oder optional sind, ob sie mehrfach vorkommen dürfen (auch Kardinalität genannt) und möglicherweise zulässige Werte [15]. Alle instanziierenden FAIR-DOs müssen mindestens Werte für die verpflichtenden Attribute spezifizieren und dürfen, abhängig von der Konfiguration des KIP, eigene Attribute zu den im KIP spezifizierten optionalen Attributen hinzufügen. Jedes Attribut verweist auf einen Datentyp (PID-BasicInfoType, PID-InfoType) und kann weitere Informationen hinzufügen. Basierend darauf kann ein FAIR-DO in Gänze syntaktisch validiert werden (für eine semantische Validierung müsste die semantische Korrektheit der Daten geprüft werden können), weshalb ein KIP als Policy für verschiedenste Anwendungszwecke dienen kann.

Durch die (semantisch angereicherten) Verknüpfungen zu anderen FAIR-DOs, deren ebenso hohe Qualität hinsichtlich Maschinenlesbarkeit und der Typisierung (mit Datentypen und KIPs), wird die gesamte FAIR-DO-Umgebung und jedes einzelne FAIR-DO in ihrem Kontext maschineninterpretierbar [9, 1]. Dies erlaubt "eine schnelle Entscheidungsfindung basierend auf dem PID-Record bevor versucht wird die referenzierten Inhalte abzufragen" [15]. FAIR-DOs können als eine sinngemäße "Übertragung objektorientierter Prinzipien auf (Meta-)Daten" [20] zusammengefasst werden und harmonisieren hoch-diverse Dateninfrastrukturen sowohl technologieoffen als auch langlebig auf einer einheitlichen Abstraktionsebene [1].

Eine zentrale technische Infrastrukturkomponente für das oben beschriebene FAIR-DOs ist die ePIC DTR, welche alle Datentypen und KIPs verwaltet und mit einer Handle-PID referenzierbar macht [18]. Die Notwendigkeit einer DTR ergibt sich daraus, dass Datentypen und KIPs "standardisiert, einzigartig und auffindbar" [12] sein müssen. FAIR-DOs sind mithilfe ihrer PIDs auflösbar, jedoch existiert kein Index oder Register wo alle FAIR-DOs aufgelistet sind. Wenn nun Datentypen ausschließlich in FAIR-DOs abgelegt würden, könnten diese nicht gefunden und wiederverwendet werden. Durch diese mögliche Mehrfacherzeugung von identischen oder stark ähnlichen Datentypen könnte keine Semantik allein durch die Verwendung von standardisierten und universellen Datentypen abgeleitet werden, was die Maschineninterpretierbarkeit der Daten erheblich beeinträchtigt [12]. Aus diesem Grund werden Register für die Verwaltung von Datentypen und KIPs benötigt, welche die Empfehlungen der RDA erfüllen sollten [12]. Zu diesen Empfehlungen der DTR-Arbeitsgruppe der RDA gehören beispielsweise die auflösbare und permanente Identifizierung jedes Datentyps mit einer PID, das Referenzieren von relevanten Standards, Empfehlungen oder RFCs, die Möglichkeit komplexe Datentypen aus anderen zusammenzusetzen und eine gemeinsame API für alle DTRs [12, 18].

Das European Persistent Identifier Consortium (ePIC) bietet verschiedene Services rund um Persistent Identifier (PIDs) an, darunter prototypische Data Type Registries (DTRs) [6, 21, 22, 18]. Die ePIC DTRs (Test- und Produktivumgebungen) sind Cordra-Instanzen [6, 7]. Cordra ist ein Metadatenrepositorium, welches Eingabedaten anhand eines vorab spezifizierten Schemas validiert, den Ergebnissen jeweils eine PID zuweist und dieses über das Handle-System versioniert auflösbar macht [23]. Zudem sind Verlinkungen innerhalb einer Cordra-Instanz möglich, was z. B. zum Referenzieren von Datentypen in KIPs genutzt wird [6, 23].

Wie in Abschnitt 2.4 deutlich wird, ist die derzeitige DTR ihren zukünftigen Aufgaben nicht mehr gewachsen und muss ersetzt oder erweitert werden. Es existieren andere Metadatenrepositorien, wie beispielsweise Metastore [24], welche einen vergleichbaren Funktionsumfang und Funktionsweise wie Cordra besitzen, aber aus gleichem Grund zukünftige Aufgaben nicht erfüllen können werden. Zum Zeitpunkt der Veröffentlichung dieser Arbeit ist nach umfangreicher Recherche keine DTR bekannt, die diese Probleme löst.

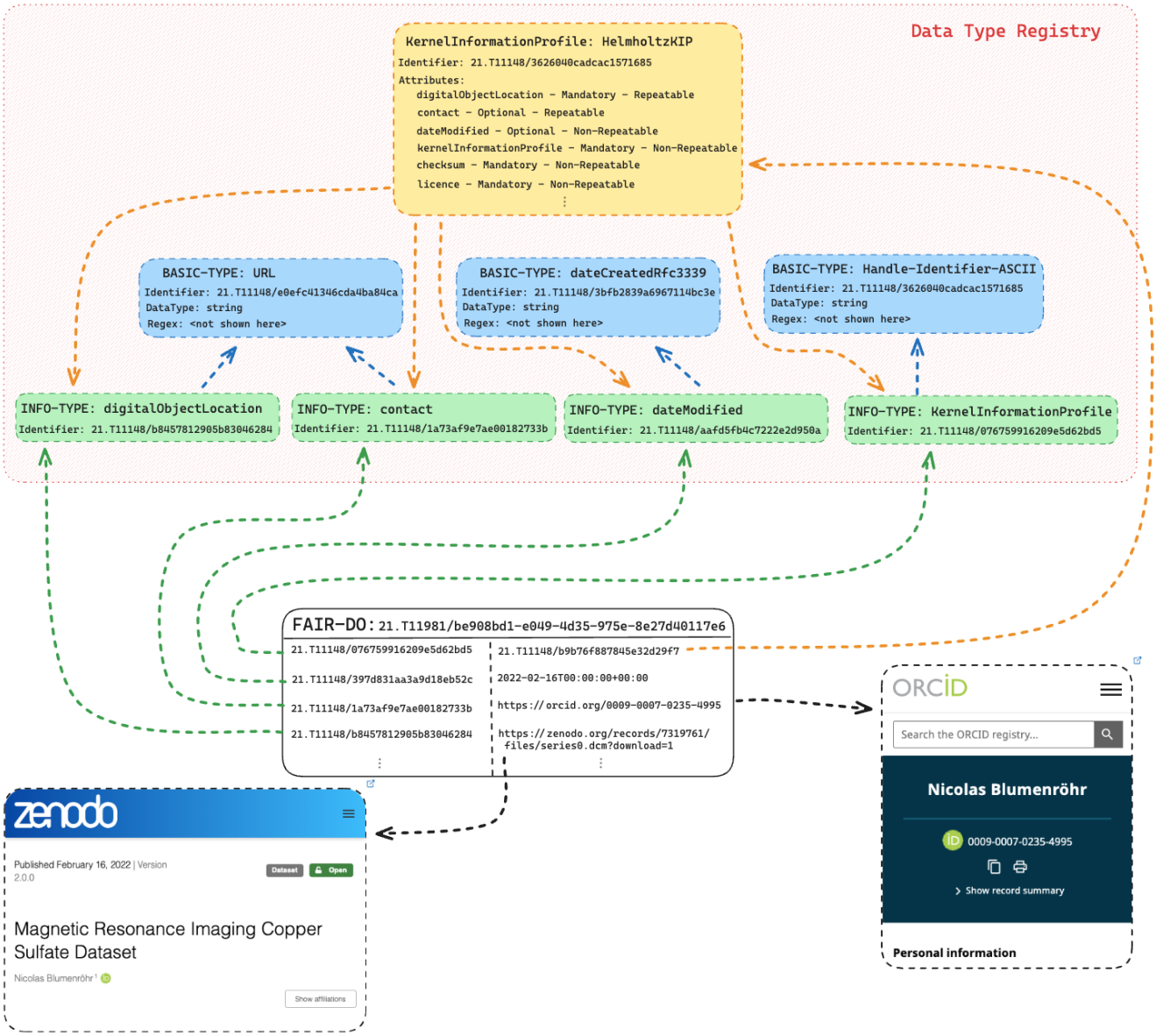

Abbildung 2.2.: Beispiel für ein FAIR-DO inklusive Verweise auf Datentypen, KIP und externe Ressourcen

Abbildung 2.2 zeigt ein Beispiel für ein gekürztes, aber echtes FAIR-DO, welches mit der Handle-PID `21.T11981/be908bd1-e049-4d35-975e-8e27d40117e6` auflösbar ist. Es ist klar erkennbar, dass der referenzierte Handle-Record eine Menge von Schlüssel-Wert-Paaren ist. Die Schlüssel verweisen auf Datentypen innerhalb der ePIC DTR, welche als rote Box gekennzeichnet ist. Die wenigen Beispielwerte sollen die Vielfältigkeit von möglichen Informationen in FAIR-DOs darstellen und entsprechen den jeweiligen Datentypen In diesem konkreten Beispiel wird auf einen Datensatz auf Zenodo [25], einem Datenrepositorium, verwiesen, eine Person mithilfe ihrer ORCiD referenziert und eine Datumsangabe bezüglich der letzten Änderung getätigt. Außerdem wird auf das KIP, welches von diesem FAIR-DO erfüllt wird und ebenfalls in der ePIC DTR abgelegt ist, verwiesen. Die Farben der einzelnen Elemente und Verbindungen zwischen diesen sind, zum besseren Verständnis, äquivalent zu denen in Abbildung 2.1.

2.3. Von Maschinenlesbarkeit zu Maschinenprozessierbarkeit

“Machine-Actionability” [2, 1] (zu Deutsch: Maschinenprozessierbarkeit) ist ein Ausdruck für die Fähigkeit einer Maschine (Meta-)Daten zu einem hohen Automatisierungsgrad, also idealerweise voll automatisiert, zu verarbeiten und zu wissen, welche Aktionen auf (Meta-) Daten ausgeführt werden können [2]. Es soll von Daten auf mögliche ausführbare Aktionen geschlossen werden, aus denen die “Bestmögliche” für den Anwendungsfall herausgesucht wird. Angenommen, eine Anwendung benötigt das ORCiD Profil, es ist jedoch lediglich eine URL auf das ORCiD-Profil gegeben. In diesem Fall könnte automatisch ein Verarbeitungsworkflow errechnet werden, der (1) die ORCiD aus der URL extrahiert, (2) diese ORCiD in eine URL für die `orcid.org`-API einsetzt, (3) diese API mithilfe eines API-Keys via HTTP abfragt und (4) sowohl das ORCiD-Profil, als auch den HTTP Status Code typisiert zurückgibt.

Um diese Kategorie von Problemen lösen zu können, ist ein “Verständnis” der (Meta-)Daten und eine Zuordnung von möglichen Aktionen zu den (Meta-) Daten notwendig, damit die Maschine (Meta-)Daten in ihrem jeweiligen Kontext korrekt interpretieren kann. Aufgrund der Debatte hinsichtlich der Fähigkeit von Computern Informationen semantisch verarbeiten zu können und diese tatsächlich zu verstehen [2, 26], wird in dieser Arbeit das “Verständnis” ausschließlich im symbolischen Sinn zur einfacheren Vorstellung referenziert, aber letztendlich die Maschineninterpretierbarkeit gemeint. Der Prozess von strukturierten Daten, über die Maschinenlesbarkeit und Maschineninterpretierbarkeit (quasi das Verstehen von (Meta-)Daten), zur Maschinenprozessierbarkeit mit den automatisiert ausführbaren Aktionen ist in Abbildung 2.3 dargestellt und wird nun im Kontext von FAIR-DOs näher erläutert:

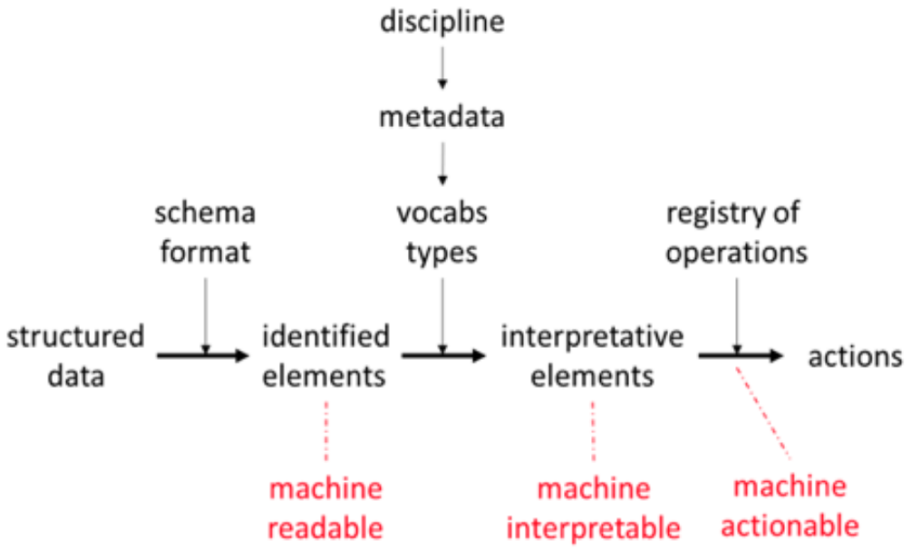

Abbildung 2.3.: Prozess von strukturierten Daten zu ausführbaren Aktionen [2]

Maschinenlesbarkeit (englisch: machine-readability) kann als “Daten in einem vom Computer ohne menschliche Interaktion verarbeitbaren Format, welches keine semantischen Informationen verliert” (vom Autor aus dem Englischen übersetzt) [27] verstanden werden. Solche Formate sind beispielsweise JSON, XML, RDF, welche aktiv zum Austausch von Informationen zwischen Maschinen verwendet werden, aber auch (wenn auch eingeschränkt) semantisch aufbereitetes HTML, welches ebenfalls für Menschen verständlich ist [28]. Abbildung 2.3 fordert für FAIR-DOs jedoch mehr als nur diese “klassischen” strukturierten Daten, indem ein Schema für dieses Format angewendet werden muss und zudem jedes einzelne FAIR-DO persistent und eindeutig identifizierbar ist [2, 1]. Strukturierte Daten können als sogenannter Bitstrom innerhalb eines FAIR-DO als externe Ressource referenziert werden. Diese Vorgaben für die Maschinenlesbarkeit sind (nicht nur für FAIR-DOs) anwendungsunabhängig.

Maschineninterpretierbarkeit (englisch: machine-interpretability) baut auf der Maschinenlesbarkeit auf und erweitert diese um ein "Verständnis" der Semantik. Klassischerweise werden dazu Vokabulare, Taxonomien, Ontologien [29], Technologien des Semantic Web [30], Knowledge-Graphen [31] oder Linked Data [32] verwendet. Dies ist bei FAIR-DOs ebenfalls möglich, jedoch erschließt sich die Semantik vor allem durch die Typisierung, die KIPs und die Verknüpfungen zu anderen FAIR-DOs. Anwendungsfallspezifische Informationen tragen auf dieser Ebene erheblich zum kontextuellen "Verständnis" des FAIR-DO bei und können beispielsweise durch die Spezifizierung von Datentypen oder KIPs ausgedrückt werden [15].

Maschinenprozessierbarkeit (englisch: machine-actionability) baut wiederum auf der Maschineninterpretierbarkeit zum "Verständnis" der (Meta-)Daten auf und erweitert diese um eine Assoziation von ausführbaren Aktionen. Eine Analogie dieser Idee sind die in macOS existierenden "Schnellaktionen", welche zu bestimmten Dateiendungen passende Aktionen anbieten (z. B. Rotation für Bilder, Annotation für PDF-Dokumente,…) [33], aber nicht den Inhalt der jeweiligen Datei evaluieren. Den Nutzenden wird das Gefühl vermittelt, dass die Maschine schon intuitiv "weiß", was sie möglicherweise tun möchten, obwohl lediglich eine Heuristik ohne semantisches Verständnis der Dateien diese Empfehlungen ermittelt. FAIR Digital Object Operations (FAIR-DO-Ops) sollen durch die Typisierung und die semantische Verknüpfung von FAIR-DOs in der Lage sein tatsächlich zu "verstehen", was mit einem FAIR-DO gemacht werden kann. Es muss daher eine Assoziation von Operationen und Datentypen stattfinden, wofür sich mehrere Ansätze in der Konzeptionierung oder Entwicklung befinden [2, 34, 35]. In dieser Arbeit wird auf den Ansatz der typgebundenen Operationen fokussiert, welcher innerhalb eines Operationsregisters (technologieagnostische) Operationen stark an Datentypen knüpft. Außerdem existieren beispielsweise servicegebundene Operationen, welche abhängig vom jeweiligen Service eine vorab bestimmte Menge an Operationen anbieten.

Die Ausführung von Methoden auf verteilten Objekten (z. B. mit RMI-IIOP oder CORBA) ist wohlbekannt [36]. In diesen verteilten Objektarchitekturen werden die Objekte mithilfe eines Namensdienstes (z. B. über JNDI) referenziert und die Methoden stark an die Objekte gebunden und auf diesen ausgeführt [36]. Obwohl diese starke Bindung von Datentypen und Operationen bei FAIR-DOs ebenfalls existiert, unterscheiden sich FAIR-DOs in der Technologieoffenheit und zugrundeliegenden Semantik massiv von verteilten Objektarchitekturen. Aus diesem Grund können bereits existierende Entwicklungen und technische Lösungen (z. B. Namensdienste) aus den vielen Ansätzen für verteilte Objektarchitekturen nicht direkt auf FAIR-DOs übertragen werden, aber sehr wohl als Inspiration dienen.

2.4. Anwendbare Prinzipien der Objektorientierung

ISO/IEC 2382 definiert, dass Objektorientierung "eine Technik oder eine Programmiersprache ist, die Objekte, Klassen und Vererbung unterstützt"(vom Autor aus dem Englischen übersetzt) [37], aber oftmals auch das Verstecken von Daten (auch: Verkapselung), Abstraktion, Nachrichtenübermittlung, dynamische Bindung und Polymorphie als Anforderungen verstanden werden [37]. Diese Mechanismen sind durch objekt-orientierte Programmiersprachen (beispielsweise Java, C# oder C++) in der Informatik wohlbekannt und werden bereits seit einiger Zeit praktiziert. Für die Anwendung auf FAIR-DOs werden im Folgenden ausgewählte Prinzipien der Objektorientierung definiert und im Kontext von FAIR-DOs analysiert.

2.4.1. Verkapselung

Verkapselung (engl. Encapsulation) wird im Sinne von OOP als “Sprachmechanismus zur Verhinderung von Zugriffen auf Komponenten von Objekten” [38] und als “Konstrukt zum Bündeln und Binden von Daten und darauf ausführbaren Methoden” [38] verstanden. In vielen objektorientierten Sprachen werden dafür Klassen verwendet, welches Attribute mit unterschiedlichen Datentypen und Zugangsbeschränkungen, sowie Methoden und Funktionen beinhalten [38, 39]. Durch Zugriffsbeschränkungen (engl. access modifier) können Attribute, Referenzen, Operationen und andere Implementierungsdetails für andere Klassen verborgen werden, auch als Information Hiding oder Data Hiding bekannt [40]. Für FAIR-DOs findet die Verkapselung insofern Anwendung, als innerhalb von KIPs mehrere Datentypen gebündelt referenziert und dementsprechend innerhalb der instanziierenden FAIR-DOs die Werte spezifiziert werden. Zum besseren Verständnis können KIPs mit Klassen aus der objektorientierten Programmierung (OOP) verglichen werden, da die instanziierenden Objekte automatisch eine Identität erhalten. Gleiches gilt auch für PID-InfoTypes, in welchen ebenfalls mehrere Datentypen zusammengefügt werden können, aber mittels eines JSON-Objekts innerhalb eines anderen FAIR-DO instanziiert werden. PID-InfoTypes sind eher mit Records vergleichbar, da lediglich eine typisierte Darstellung von Werten ohne eigene Identität innerhalb eines FAIR-DO-Records vorliegt. Für beide Verkapselungsmodelle findet nicht zwangsläufig Information Hiding statt.

2.4.2. Abstraktion

Bekannte Beispiele für Abstraktion sind abstrakte Klassen, Interfaces, Vererbung oder generell Mittel, die Konzepte von der Umsetzung trennen, Duplikate vermeiden und die Wiederverwendbarkeit verbessern [41, 42]. FAIR-DOs sind meistens eine Abstraktion von Ressourcen aus verschiedenen Ökosystemen (teilweise als Bitstrom bezeichnet) sein, da FAIR-DOs stark generalisiert sind und einen abstrakten, maschineninterpretierbaren und idealerweise maschinenprozessierbaren Zugriff auf diese externen Ressourcen ermöglichen [1, 9]. Die Inhalte eines FAIR-DO werden mithilfe von KIPs und Datentypen spezifiziert [18, 43]. KIPs und PID-InfoTypes spezifizieren den Aufbau von FAIR-DOs und/oder einzelnen Werten. Außerdem können KIPs und PID-InfoTypes weiter durch Vererbung abstrahiert werden. In seinem Paper „Automated Schema Extraction for PID Information Types“ beschreibt Schwardmann diese Zusammenhänge und begründet, warum es wichtig ist, dass PID-BasicInfoTypes nicht weiter abstrahiert werden können [18]. Zusammengefasst, sind PID-BasicInfoTypes die atomaren Typen in der Typisierung von FAIR-DOs und deren Endlichkeit ist zur Verhinderung von Nicht-Terminierung von Auflösung und Referenz (z. B. beim Erstellen von Schemata oder zur Validierung) elementar [18].

2.4.3. Methoden und Funktionen

Abhängig von der Definition oder Interpretation von FAIR-DO-Ops sind diese entweder als Funktion oder Methode im Kontext der Softwareentwicklung einzuordnen. So sind servicegebundene Operationen eher als Funktionen interpretierbar, da die Spezifikation der Funktion im Service geschieht. Die stark typgebundenen Operationen, die in dieser Arbeit behandelt werden, sind durch die starke Bindung (vergleichbar zu Komposition) eher als Methoden zu verstehen, obwohl Änderungen an den jeweiligen Objekten nicht erwünscht sind. Das Verändern von FAIR-DOs nach Persistierung im Handle-System sollte so wenig wie möglich geschehen, damit diese effektiv gecacht werden können und die Persistenz gewahrt bleibt [15, 44]. Von Löschoperationen ist gänzlich abzusehen [44, 15]. Dies ist ein weiterer Unterschied zu objektorientierten Programmierung (OOP), wo das Erstellen, Modifizieren und Löschen von Objekten leichtfertig geschieht.

Durch FAIR-DO-Ops entstehen weitere Anforderungen hinsichtlich der Assoziation von Operationen mit Datentypen und KIPs, dem Ermitteln von möglichen Workflows zwischen zwei Datentypen/KIPs und der Umsetzung von Polymorphie. Diese neuen Konzepte, Anforderungen und Mechanismen müssen aus den gleichen Gründen wie für Datentypen und KIPs in einer DTR umgesetzt werden. Auch, wenn FAIR-DO-Ops mit der ePIC DTR syntaktisch abgebildet werden können, indem einfach ein weiteres JSON-Schema erstellt wird, ist Cordra zur Umsetzung eines Operationsregisters nicht geeignet, da besonders die gegenseitige Assoziation zwischen Datentypen, KIPs und den FAIR-DO-Ops eine zentrale Rolle für die Auffindbarkeit und Nutzbarkeit darstellt.

2.4.4. Vererbung und Polymorphie

In OOP wird Vererbung meistens für die Spezifikation von Subtypen, zur Erweiterung von bestehenden Funktionalitäten (siehe Open-Closed-Prinzip [45, 46] und Liskov-Substitution [42, 46, 47]), zur Unterstützung von allgemeinen Funktionalitäten (cross-cutting concerns z. B. copyable, serializable, …) oder Wiederverwendung existierender Programmbestandteile (z. B. Mixins [48]) verwendet [49]. Zur korrekten Abbildung und Ausführung von Vererbung muss unbedingt eine Überschreibungslogik für Attribute, Referenzen und Methoden existieren und beachtet werden. Diese Logik variiert zwischen den Programmiersprachen mit den unterschiedlichen umgesetzten Prinzipien (z. B. Mehrfachvererbung, Multi-Level-Vererbung, Typkonvertierung, Kapselung, …) und Modellen, sodass abgesehen von der Tatsache, dass die erbende Klasse meistens überschreiben kann, keine generelle Annahme getroffen werden kann. Für FAIR-DOs ist dies genauso der Fall, wobei aufgrund des Entwicklungsstandes mehrere Ansätze existieren.

Im bisherigen Konzept (nach Schwardmann) sind rudimentäre konzeptuelle Ansätze von Vererbung vorhanden und in der ePIC DTR bereits syntaktisch umgesetzt [18]. Die Umsetzung erfolgt mit nicht-maschinenlesbaren Verweisen in den JSON-Schemata von Corda. Alle Funktionalitäten der ePIC-DTR sind in diesen JSON-Schemata in Form von Syntax-Regel abgebildet [50, 12]. Es sind keine funktionalen Erweiterungen, z. B. für spezielle Abfrage- oder Auflösealgorithmen, an Cordra für die Aufgabe als DTR vorgenommen worden. Dadurch sind Mechanismen, wie etwa die Evaluation von Vererbungshierarchien, die Rückwärtsabfrage von Verlinkungen (z. B. zum Ermitteln von KIPs in denen ein Datentyp verwendet wurde) nicht umsetzbar.

In Sektion 4.2 wird näher auf den Ansatz dieser Arbeit eingegangen. Polymorphie basiert auf Vererbung und hat viele Formen von Typ-, Operator- und Funktionsüberladungen über dynamische Bindungen bis hin zu parametrischer Polymorphie (z. B. Java Generics [51]) [52]. Dieses Konzept ist bislang im Umfeld der FAIR-DOs nicht umgesetzt wird aber für diese Arbeit in Sektion 4.2 genutzt.

2.4.5. Komposition und Assoziation

Es existieren eine starke (Komposition) und schwache (Assoziation) Bindung sowohl in OOP als auch in FAIR-DOs. Eine Assoziation zwischen zwei Klassen oder zwei FAIR-DOs ist mit einer Referenz einfach umsetzbar, da beide Objekte unabhängig voneinander existieren können. Bei den FAIR-DOs wird mithilfe von PIDs referenziert.

Eine Komposition wird benötigt, wenn zwei Objekte nicht unabhängig voneinander existieren können und daher stark aneinander gebunden sind. Für FAIR-DOs bietet sich an dieser Stelle die Instanziierung als Wert in einem anderen FAIR-DO an, falls keine Referenzen nötig sind und das Objekt keine eigene Identität besitzen soll. Andernfalls muss, um die Wiederverwendbarkeit zu garantieren, ein eigenes FAIR-DO erstellt und dieses mithilfe einer PIDs referenziert werden.

3. Zielsetzung

Aus Kapitel 2 ergeben sich einige Anforderungen in Hinblick auf die Modellierung von FAIR Digital Object Operations (FAIR-DO-Ops), die Weiterentwicklung des FAIR-DO-Konzepts und die Neuentwicklung einer Data Type Registry (DTR). FAIR-DO-Ops müssen technologieoffen, langlebig und, genau wie Datentypen und KIPs, eindeutig definiert sein [2]. Dies erfordert, auch aus Gründen der Auffindbarkeit, eine Integration in eine DTR, welche FAIR-DO-Ops mit Datentypen und KIPs assoziiert. Das bereits im ePIC System existierende Datenmodell muss daher für diese neu entwickelten Operationen angepasst werden. Vererbung und Polymorphie sind essenziell für die Wiederverwendbarkeit von FAIR-DOs und FAIR-DO-Ops, sowie deren effektive Nutzung. Diese müssen daher ebenfalls im Datenmodell sinnvoll abgebildet, sowie in einer neuen DTR validiert und abgefragt werden können. Es müssen Abfragen der Vererbungshierarchie, der Zuordnung von Operationen zu einzelnen Datentypen und grundlegende Verwaltungsmechanismen für CRUD-Operationen mithilfe einer REST-API ermöglicht werden.

Im Rahmen dieser Arbeit soll zunächst das bisherige Datenmodell an die neuen Anforderungen angepasst werden. Dazu gehört besonders der Entwurf und die Integration eines Datenmodells für FAIR-DO-Ops und die Entwicklung von Vererbungsmechanismen. Im Anschluss soll eine neue prototypische Referenzimplementierung dieses neuen Datenmodells entwickelt werden. Die Integrated Data Type and Operations Registry with Inheritance System (IDORIS) muss mindestens das neue Datenmodell mit einer CRUD-fähigen REST-API verwalten können, aber keine Authentifikationsmechanismen umsetzen. Zusätzlich sollen verschiedene Vererbungsmechanismen umgesetzt werden. Konkret soll die Vererbungshierarchie abgefragt und eine Liste aller vom KIPs geerbten Attribute ausgegeben werden können. Die Assoziation von Operationen mit Datentypen und KIPs muss ebenfalls bidirektional abgefragt werden können. IDORIS ist eine Registry zur Verwaltung von Datentypen und FAIR-DO-Ops, soll aber selbst keine FAIR-DO-Ops ausführen. Zur Ausführung von FAIR-DO-Ops muss zukünftig eine Ausführungskomponente entwickelt werden, die in Kapitel 8.2 näher beschrieben wird.

Eine Abfrage aller Operationen in der Vererbungshierarchie ist optional. Für die Maschinennavigierbarkeit der API ist HATEOAS ein bekanntes Werkzeug, welches auf API-Seite jedoch zusätzlichen Aufwand bedeutet und somit ebenfalls optional ist. Eine Data Type Registry (DTR) sollte die enthaltenen Eingaben validieren können, was für Syntaxüberprüfungen mit etwas Aufwand lösbar ist, aber einen immensen Aufwand für semantische Überprüfungen verursacht. Daher ist die Validierung der Eingabedaten mit sinnvollen Rückgaben bei Fehlern ebenfalls optional.

Die ePIC DTR weist ihren Einträgen automatisch PIDs zu und hinterlegt im Handle-System entsprechende Verweise. Um diese Funktionalität umsetzen zu können, ist ein hohes Maß an Koordination mit entweder Handle direkt oder einem Service, der diese Koordination abstrahiert, wie z. B. dem Typed PID Maker [53], notwendig, was im Rahmen dieser Arbeit nicht erbracht werden kann. Außerdem wird es trotz aller Bemühungen Inkompatibilitäten zum Datenmodell der ePIC DTR geben, sodass für eine produktive Nutzung von IDORIS eine Migration oder Integration erforderlich ist, was ebenfalls aufgrund des Zeitrahmens nicht in dieser Arbeit geschehen kann. Andere fortgeschrittene Funktionalitäten, wie etwa Versionierung, Föderation von mehreren Instanzen oder eine Weboberfläche sind ebenfalls zeitlich nicht realisierbar. Für die Umsetzung dieser Funktionalitäten in der Zukunft ist bei der Entwicklung, wo möglich, auf Erweiterbarkeit zu achten.

4. Modellierung

Das bisherige Datenmodell für die Typisierung von FAIR-DOs ist in der ePIC DTR in Form von JSON-Schemas abgebildet [6]. Entsprechende Schemata können dem Anhang entnommen werden. Die Idee hinter IDORIS sieht vor, dass im Gegensatz zur ePIC DTR mit den gespeicherten Daten im Detail gearbeitet werden soll. Da IDORIS speziell als DTR entwickelt wird, soll das Datenmodell in IDORIS objektorientiert abgebildet werden und den neuen Anforderungen gerecht werden, was eine Chance für weitere Modifikationen schafft. Dazu gehören die Vereinfachung von Datentypen und KIPs (siehe 4.1), die Einführung von Vererbung und Polymorphie im FAIR-DO-Kontext (siehe 4.2) und die Definition von FAIR-DO-Ops (siehe 4.3).

4.1. Vereinfachung von Datentypen und KIPs

Beim Vergleich der Schemata von PID-InfoTypes (siehe Anhang A.3) und KIPs (siehe Anhang A.4) fällt auf, dass diese überwiegend identisch sind. Aufgrund der Vielzahl an Gemeinsamkeiten wird zur Vermeidung von Duplikaten für IDORIS das Konzept der Typprofile (engl. Type Profiles) eingeführt, welches PID-InfoTypes und KIPs vereinen soll. Typprofile spezifizieren typisiert sowohl den Aufbau komplexer Werte in FAIR-DOs (siehe PID-InfoType) als auch den Aufbau von FAIR-DOs selbst (siehe KIP). Dazu werden Metadaten zum Typprofil erhoben, wie beispielsweise einen Namen, eine Beschreibung, eine Liste von erwarteten Einsatzszenarien, eine Liste von Mitwirkenden und ein Datum für die Erstellung und letzte Modifikation. Diese Metadaten waren bereits in den PID-InfoTypes und KIPs enthalten und werden in den Typprofilen beibehalten, jedoch in ihrer Struktur vereinheitlicht. Von zentraler Bedeutung für Typprofile ist die Spezifikation der Attribute, die erfüllt werden sollen. Zusätzlich zu der Referenz auf den jeweiligen Datentyp werden verpflichtend ein menschenlesbarer Name für das Attribut in seinem Kontext, eine Angabe zur Obligation (Wahl zwischen “Mandatory” und “Optional”), optional eine Beschreibung, die Wiederholbarkeit und ein Standardwert spezifiziert. Es existiert außerdem das Feld “subSchemaRelation”, welches beschreibt, wie das Typprofil von einem FAIR-DO oder einem erbenden Typprofilen verwendet und erweitert werden darf [18].

Tabelle 4.1 zeigt die Gemeinsamkeiten und Unterschiede zwischen PID-InfoTypes, KIPs und Typprofilen. Dabei wird deutlich, dass Typprofile viele Kernfunktionalitäten abbilden, aber auch einige Funktionen aufgrund der fehlenden Spezifikation, meist fehlenden Dokumentation und der fehlenden Nutzerakzeptanz (in der Abteilung DEM) derzeit nicht umgesetzt werden. Anstatt beispielsweise nur einen Standardwert für das gesamte FAIR-DO zu spezifizieren, können in IDORIS auch Standardwerte für jeden Wert in einem Typprofil, aber keine konstanten Werte, angegeben werden. Undokumentierte oder abgekündigte Funktionalitäten, wie die Spezifikation von Restriktionen mit nicht in der DTR definierten JSON-Schema ähnlichen Schlüsselwörtern, textuelle Repräsentation von Beziehungen der einzelnen Properties, oder Generierung eines JSON-Schemas wurden nicht übernommen.

	PID-InfoTypes	KIPs	Typprofile
Gemeinsamkeiten	Referenzierung mit PIDs		
	Name und Beschreibung		
	Verweis auf Mitwirkende mit Text oder ORCiD		
	Datum für Erstellung und letzte Modifikation		
	Erwartete Nutzungsszenarien		
	Verweis auf existierende Standards		
	Richtlinien für die Instanzen bezüglich der Umsetzung der spezifizierten Eigenschaften (engl. Feldname: "Relations for Sub-Schemas in Properties-Section") vergleichbar zu Verben aus JSON-Schemas [18]		
	Spezifikation des Aufbaus aus anderen PID-BasicInfoTypes, PID-InfoTypes und Kernel Information Profile mit anwendungsspezifischem Namen und Details, einer booleschen Angabe zu Wiederholbarkeit und Obligation in der Liste "properties"		Ähnlich zu den PID-InfoTypes und KIPs, allerdings wurde die Liste zu "attributes" umbenannt, um der Ähnlichkeit zu Klassen in der Objektorientierung Ausdruck zu verleihen.
Unterschiede	Erzeugung durch Einbetten in FAIR-DO als Value Object ohne Identität	Erzeugung als FAIR-DO	Erzeugung als FAIR-DO, Verweis auf ein FAIR-DO oder durch Einbetten in ein FAIR-DO als Wert
	Nicht wiederverwendbare Instanzen	Wiederverwendbare Instanzen	Abhängig von der Art der Erzeugung
	-	-	Möglichkeit zum Verbot des Einbettens als Wert in FAIR-DOs und somit explizites Gebot zur Erzeugung von FAIR-DOs
	-	-	Angabe einer Lizenz
	Mögliche Spezifikation eines Standardwertes für das gesamte FAIR-DO in Form eines serialisierten JSON-Objekts oder -Arrays		Standardwerte sowohl für jeden Wert als auch für das gesamte Typprofil in Gänze möglich
	Spezifikation eines konstanten Wertes für jede einzelne Property möglich		-
	Restriktionen mit Schlüsselwörtern vergleichbar zu JSON-Schemas [18]		-
	automatisch generiertes JSON-Schema zur Validierung	-	-
	experimentelle, nicht-maschinenlesbare Option zur textuellen Repräsentation von Beziehungen der einzelnen Properties	-	-

Tabelle 4.1: Vergleich der Gemeinsamkeiten und Unterschiede der Datenmodelle, der Konzepte sowie Anwendung von PID-InfoTypes [6, 18], KIPs [6] und Typprofilen

PID-BasicInfoTypes wurden in BasicDataTypes umbenannt und die Struktur der Metadaten vereinfacht. Per Definition zeigt der Schlüssel zu einem Wert in einem FAIR-DO-Record (siehe Abbildung 2.2) auf einen Datentyp. Die Einführung der Oberklasse DataType, die BasicDataTypes und Typprofile als Unterklassen hat, ermöglicht es Werte oder Verweise auf FAIR-DOs, die Typprofile erfüllen, typsicher zu spezifizieren. Typprofile können somit in FAIR-DO-Records als Schlüssel für Werte angegeben werden. Die Werte verweisen entweder, mithilfe einer PID, auf ein FAIR-DO, welches das Typprofil instanziiert oder instanziieren das Typprofil durch ein komplexes JSON-Objekt direkt im jeweiligen FAIR-DO-Record. Mit dieser Methode können dementsprechend Verweise auf andere FAIR-DOs, die einem Typprofil entsprechen müssen, typisiert und validiert werden. Verweise konnten zwar bislang auch schon erstellt werden, jedoch beschrieb der Datentyp für den jeweiligen Verweis im FAIR-DO einen Handle und beinhaltete keine Informationen über das verwiesene FAIR-DO, sodass nicht sichergestellt war, welche Informationen das verwiesene FAIR-DO enthält. Durch typisierte Verweise können nun Beziehungen zwischen FAIR-DOs und deren Inhalt validiert werden. Diese Validierung von typisierten Verweisen auf andere FAIR-DOs, die mit dem jeweiligen Typprofil übereinstimmen, war in der ePIC DTR nicht möglich, da KIPs dort nicht als Datentyp verstanden wurden. Außerdem werden durch diese Abstraktion Referenzen auf Datentypen generalisiert, sodass Doppelstrukturen in IDORIS vermieden werden können. Solche Doppelstrukturen würden beispielsweise bei der Attributlogik auftreten, wenn z.B: zwei separate Felder für einen Verweis auf einen BasicDataType und ein Typprofil erstellt würden. Entsprechend würde eine zusätzliche Validierungslogik benötigt, die sicherstellt, dass immer nur eines der beiden Felder ausgefüllt ist. Vererbung ist daher an dieser Stelle, auch aufgrund der funktionalen Ähnlichkeiten von BasicDataTypes und Typprofilen, die eindeutig einfachere und bessere Variante.

4.2. Vererbung und Polymorphie

Vererbung und Polymorphie sind zentrale Merkmale der Objektorientierung und erhöhen die Wiederverwendbarkeit, Erweiterbarkeit, Aussagekraft und Flexibilität [49, 52]. FAIR-DOs sollen daher ebenfalls mit Vererbung und Polymorphie angereichert werden. In der ePIC DTR konnte keine Vererbung spezifiziert oder evaluiert werden, weshalb stattdessen teilweise im Feld für anwendbare Standards (siehe Anhang A.2, A.3 und A.4) textuelle, nicht maschinenlesbare Referenzen mit einer PID abgelegt wurden oder einfach neue Elemente ohne Verweis auf das "vererbende" Element erzeugt wurden. Im gleichen Feld können jedoch auch textuelle Verweise auf nicht maschinenlesbare Standards, andere URLs oder PIDs abgelegt werden, weshalb dieses Feld mangels Spezifikation und Validierbarkeit nicht aussagekräftig genug ist, um Vererbung zu spezifizieren.

IDORIS führt daher das Feld `inheritsFrom` ein, welches ausschließlich auf das Eltern-Element verweist und eine bidirektionale Assoziation in der Datenbank abbildet. Dadurch kann die Integrität der Verweise sichergestellt werden, da nur existierende Datentypen oder Typprofile als Eltern-Elemente referenziert werden können. Außerdem kann die Vererbungshierarchie bidirektional traversiert werden, um beispielsweise alle Kinder eines Datentyps zu finden oder die Vollständigkeit von erbenden Elementen zu prüfen. In Sektion 4.1 wurden sowohl Typprofile und BasicDataTypes als auch deren Oberklasse DataType eingeführt. Die Vererbungsmechanismen variieren jedoch stark zwischen BasicDataTypes und Typprofilen, weshalb diese getrennt betrachtet werden.

BasicDataTypes spezifizieren die Definition von einfachen Werten in FAIR-DOs. Eine Vererbung von BasicDataTypes soll immer dann verwendet werden, wenn die Definition der Werte des Elternelements weiter spezifiziert und eingeschränkt werden soll. Somit haben Kinder eines BasicDataTypes immer eine Teilmenge der Werte des Eltern-Elements. Ein Anwendungsbeispiel ist die Spezifikation eines BasicDataTypes für eine ORCiD-URL, welches von einem BasicDataType für `https`-URLs erbt, welcher wiederum von URL erbt. Mit jedem Schritt wird die Definition der Werte mithilfe von regulären Ausdrücken weiter spezifiziert und eingeschränkt, sodass die Werte des Kindes immer eine Teilmenge der Werte des Eltern-Elements sind. In einem

Typprofil könnte nun für das Attribut `contact` der BasicDataType URL verwendet werden. Wenn nun in einem FAIR-DO, welches diesem Typprofil entsprechen soll, ORCiD-URL spezifiziert wird, kann die PID für den BasicDataType ORCiD-URL als Referenz verwendet werden. Beim Validieren des FAIR-DO wird überprüft, ob die spezifizierte URL eine gültige ORCiD-URL ist und ob ORCiD-URL von dem BasicDataType URL erbt. Dieses Beispiel macht deutlich, warum erbende BasicDataTypes keine Erweiterungen der Definitionen des Elternelements zulassen dürfen, um die Integrität und Gültigkeit der Vererbung zu gewährleisten. Würde ein erbender BasicDataType beispielsweise den regulären Ausdruck des Eltern-Elements erweitern, sodass die potenzielle Ergebnismenge vergrößert und nicht eingeschränkt wird, kann die Integrität der Vererbungshierarchie nicht mehr gewährleistet werden, da die Werte des Kindes nicht mehr eine Teilmenge der Werte des Eltern-Elements sind. Gleiches gilt für Mehrfachvererbung, da die Mengen der Eltern-Elemente nicht disjunkt sein müssten, um nicht der Definition nach mindestens ein Eltern-Element zu erweitern. Eine Validierung dieses Kriteriums ist mindestens schwierig und würde die Komplexität des Systems für wenige Sonderfälle unnötig erhöhen, weshalb für BasicDataTypes Einfachvererbung verwendet wird.

Typprofile bilden komplexe Strukturen ab, die sowohl den Wert für Attribute in FAIR-DOs (vergleichbar zu PID-InfoTypes) als auch die Struktur von FAIR-DOs selbst (vergleichbar zu KIPs) spezifizieren können. Im Gegensatz zu BasicDataTypes können Typprofile mehrere Attribute spezifizieren, die wiederum auf je einen Datentyp (BasicDataType oder Typprofil) verweisen. Dies ermöglicht sowohl die weitere Spezifikation von Attributen durch striktere (erbende) Datentypen als auch die Erweiterung durch das Hinzufügen neuer Attribute.

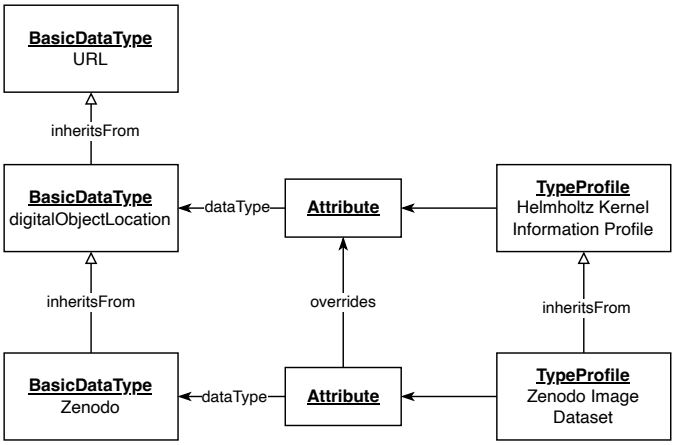

Abbildung 4.1.: Beispielhafte Visualisierung des Attribut-Overrides

Attribut-Overrides sind in Typprofilen erlaubt, um die Definitionen der Attribute des Eltern-Elements zu überschreiben. Dazu ist in der Attribut-Definition ein Feld `overrides` enthalten, welches die Referenz auf das "Eltern"-Attribut enthält. Für ein gültiges Attribute-Override muss das Kind-Attribut auf entweder den gleichen Datentyp, wie das Eltern-Attribut, oder auf einen Datentyp, der von dem Datentyp des Eltern-Attributs erbt, verweisen. Es muss zudem sichergestellt werden, dass Attribute, die überschrieben werden, nicht weniger restriktiv als im Eltern-Element spezifiziert werden. Beispielsweise dürfen verpflichtende Attribute in einem erbenden Typprofil nicht optional sein.

Abbildung 4.1 zeigt ein vereinfachtes Beispiel für die Anwendung von Attribut-Overrides. Zenodo[25] ist eine einschlägig bekanntes Open-Access Datenrepositorium, welches hier nur beispielhaft verwendet wird. Das "Zenodo Image Dataset"-Typprofil erbt vom "Helmholtz-KIP [15]"-Typprofil. Dieses hat ein Attribut, welches auf den BasicDataType "digitalObjectLocation" verweist und eine URL ist. Ein "Zenodo Image Dataset" bekommt diesen BasicDataType standardmäßig vererbt, benötigt aber zum Zwecke dieses Beispiels eine Referenz auf Zenodo. Dazu kann mithilfe des neuen Attribut-Overrides nun einfach ein neues Attribut erstellt werden, welches das vom "Helmholtz-KIP"-Typprofil überschreibt. Dieses überschreibende Attribut muss zwangsläufig einen vom Datentyp des überschriebenen Attributs erbenden Datentypen verweisen. Das ist in diesem Fall der BasicDataType "Zenodo", welcher somit im Typprofil überschrieben wurde, sodass in FAIR-DOs für dieses Attribut nur Werte vom Typ "Zenodo" oder davon erbenden Typen verwendet werden können.

Als Vererbungsmechanismus wurde für Typprofile die Mehrfachvererbung gewählt, um die Wiederverwendbarkeit und Erweiterbarkeit der Typprofile zu erhöhen. In vielen objektorientierten Programmiersprachen ist Mehrfachvererbung aufgrund der hohen Komplexität nicht erlaubt [49]. Java bietet mit Interfaces eine Möglichkeit, Mehrfachvererbung zu umgehen, indem Klassen mehrere Interfaces, die im Gegensatz zu Klassen nur Methoden beinhalten dürfen, implementieren können. Bei der Entwicklung des Datenmodells für IDORIS wurden zunächst Interfaces zusätzlich zu der Mehrfachvererbung modelliert, aber später wegen der Redundanz zur Mehrfachvererbung wieder entfernt. Aufgrund der datenbasierten Natur von FAIR-DOs bietet sich die Mehrfachvererbung als Mittel zur Wiederverwendbarkeit und Erweiterbarkeit an, da zusätzlich zu den FAIR-DO-Ops auch die enthaltenen Attribute vererbt werden.

Die Mehrfachvererbung von Typprofilen bringt einige Herausforderungen mit sich. Beispielsweise kann es zu Konflikten kommen, wenn zwei geerbte Typprofile das gleiche Attribut mit unterschiedlichen (semantischen) Definitionen spezifizieren, da in einem instanziierenden FAIR-DO Attribute mit der PID des jeweiligen Datentyps referenziert werden. Mögliche Ansätze zur Lösung dieses Problems wären das persistente Referenzieren von Attributen in FAIR-DOs, das Erstellen von zusätzlichen Datentypen für die spezifischen Definitionen oder das Verhindern solcher Profile durch Validierung. Das Verhindern durch Validierung scheint aufgrund des minimalen Aufwandes recht sinnvoll und effizient. Ein anderes Problem betrifft das Feld `subSchemaRelation`, welches die Beziehung zwischen einem Typprofil und einem FAIR-DO oder einem erbenden Typprofil spezifiziert. Dort muss sichergestellt werden, dass die im Paper von Schwardmann beschriebenen Verben [18], die die Beziehung zwischen den einzelnen Attributen beschreiben, spezifiziert und eingehalten werden. Dafür muss die Vererbungshierarchie traversiert und die einzelnen `subSchemaRelations` auf Kompatibilität geprüft werden. Außerdem können durch Mehrfachvererbung einfach zirkuläre Abhängigkeiten entstehen, die entdeckt und unbedingt verhindert werden müssen, damit die Einträge in IDORIS und die Validierungsalgorithmen terminieren können. In der ePIC DTR war die Vererbung von PID-BasicInfoTypes (jetzt: BasicDataType) bewusst nicht vorgesehen, damit die Terminierung eines Algorithmus sichergestellt ist, was auch bewiesen wurde [18].

Im ePIC System können Vererbung und Polymorphie aufgrund der fehlenden Möglichkeit zur Validierung sowie Auflösung der Vererbungshierarchie nicht umgesetzt werden. IDORIS implementiert eine umfassende Vererbungslogik und stellt die Integrität der einzelnen Entitäten und der Vererbungshierarchie durch Validierung (siehe Sektion 5.2) sicher. Es existieren zudem REST-API-Endpunkte, um die Vererbungshierarchie zu traversieren und die Vollständigkeit der Vererbungshierarchie zu prüfen.

4.3. Spezifikation von FAIR Digital Object Operations

In Sektion 2.3 wurde der Begriff der Maschinenprozessierbarkeit definiert und der Begriff der FAIR Digital Object Operations (FAIR-DO-Ops) eingeführt. Außerdem wurde zwischen servicegebundenen und typgebundenen Operationen unterschieden, wobei in dieser Arbeit nur typgebundene Operationen betrachtet werden. Typgebundene Operationen sind an einen Datentyp gebunden und können nur auf Instanzen dieses Datentyps ausgeführt werden. Ein Register für diese Art von Operationen muss die bidirektionale Assoziation zwischen Datentyp und Operation abbilden und abfragbar machen.

FAIR Digital Object Operations (FAIR-DO-Ops) beschreiben im Allgemeinen lediglich die Realisierung der Maschinenprozessierbarkeit durch Operationen auf FAIR-DOs. In dieser Arbeit wird diese Definition auf typgebundene FAIR-DO-Ops beschränkt und um die Definition von FAIR Digital Object Operation Type Profiles (FAIR-DO-OpTPs) und FAIR Digital Object Operation Type Profile Adapters (FAIR-DO-OpTPAs) erweitert. Im weiteren Verlauf dieser Arbeit wird zur besseren Lesbarkeit das Wort “Operation” als Synonym für FAIR Digital Object Operation (FAIR-DO-Op) benutzt, sofern aus dem Kontext nichts anderes hervorgeht.

FAIR-DOs zeichnen sich sowohl durch ihre Typisierung als auch durch ihre Fähigkeit zur Abstraktion verschiedener Technologien und Datenstrukturen aus. Beide Aspekte sollen für die Definition von FAIR-DO-Ops mithilfe von typgebundenen Operationen im Kontext dieser Arbeit berücksichtigt werden. Die Typbindung einer FAIR-DO-Op geschieht durch die Spezifikation der Felder `executableOn`, `environment` und `returns`. Das Feld `executableOn` spezifiziert immer genau ein Attribut (für Definition siehe Sektion 4.1), das auf einen Datentyp verweist, auf dem die Operation ausgeführt werden kann. In `environment` können beliebig viele Attribute für Umgebungsvariablen, die nicht notwendigerweise im jeweiligen FAIR-DO oder Wert enthalten sind (z. B. API-Keys, Verweise auf interne Netzwerkressourcen, …), spezifiziert werden. Die Rückgabewerte einer Operation werden im Feld `returns` ebenfalls mit Attributen spezifiziert.

Für die Ausführung einer FAIR-DO-Op müssen die Herausforderungen, die durch die Heterogenität der zu verwendenden Technologien sowie die Anforderungen an Erweiterbarkeit und Integrität entstehen, gelöst werden. Dazu werden, in konzeptueller Anlehnung an die Typisierung von FAIR-DOs, die FAIR-DO-Ops, die zugrundeliegende Technologie sowie die konkrete Ausführung dieser Technologie zur besseren Wiederverwendbarkeit und Erweiterbarkeit voneinander getrennt.

4.3.1. Spezifikation von FAIR Digital Object Operation Type Profiles

Dazu wird die Klasse FAIR-DO-OpTPs (siehe Modellklassendiagram in 4.4) als wiederverwendbares Verbindungsglied zwischen den FAIR-DO-Ops und der jeweiligen konkreten Ausführung einer Technologie eingeführt. FAIR-DO-OpTPs spezifizieren einige Metadaten, darunter den Namen, eine Beschreibung, eine Liste von erwarteten Einsatzszenarien, eine Liste von Mitwirkenden, Datumsangaben für die Erstellung und letzte Modifikation, Verweise auf Standards und eine Lizenz. Zusätzlich enthalten FAIR-DO-OpTPs zwei Mengen von Attributen, die auf Datentypen verweisen und die Eingabe- und Ausgabeparameter der Operation spezifizieren. Dabei wird die gleiche Definition von Attributen wie in Typprofilen verwendet, um die Wiederverwendbarkeit und Erweiterbarkeit, auch innerhalb von IDORIS, zu erhöhen, wodurch auch die Spezifikation von optionalen Attributen, Standardwerten, etc. ermöglicht wird. FAIR Digital Object Operation Type Profiles (FAIR-DO-OpTPs) sind analog zu Typprofilen vererbbar, wobei die Integrität der Vererbungshierarchie durch die Validatoren (siehe Sektion 5.2) sichergestellt wird. FAIR-DO-OpTPs können zusammenfassend als Repräsentation einer Technologie verstanden werden und Verweisen auf die Ausführung dieser in potenziell verschiedenen Umgebungen. Beispiele für Technologien sind Regex oder HTTP, welche in Sektion 5.3.5 erstellt werden. In dieser Arbeit wird zur besseren Lesbarkeit das Wort “OperationTypeProfile” als Synonym für FAIR Digital Object Operation Type Profile (FAIR-DO-OpTP) verwendet, sofern aus dem Kontext der Anwendung nichts anderes hervorgeht.

4.3.2. Spezifikation von FAIR Digital Object Operation Type Profile Adapters

Die konkrete Ausführung eines FAIR-DO-OpTP wird durch einen oder mehrere FAIR-DO-OpTPAs realisiert, die die konkrete Technologie und Implementierung spezifizieren. FAIR Digital Object Operation Type Profile Adapters (FAIR-DO-OpTPAs) werden außerhalb von IDORIS spezifiziert, da sie ausschließlich aus Richtung des FAIR-DO-OpTP abgefragt werden und keine Logik außerhalb der regulären Validierungslogik für FAIR-DOs benötigen. Somit werden FAIR-DO-OpTPAs als FAIR-DO repräsentiert und im Handle.net-System verwaltet. Eine Definition für die Inhalte eines FAIR-DO-OpTPA enthält die üblichen Metadaten (z. B. aus dem Helmholtz KIP [15]) und Informationen zur Ausführung eines FAIR-DO-OpTP. Diese Informationen umfassen die zu nutzende Technologie (z. B. Docker, Python, JavaScript), einen Einstiegspunkt (z. B. Docker-Image oder Python-Skript mit Argumenten), möglicherweise einen Verweis auf den Sourcecode und weitere technologiespezifische Details. FAIR-DO-OpTPAs sollen vom jeweiligen ausführenden System zur Laufzeit

geladen, interpretiert und auf Basis der Spezifikation des FAIR-DO-OpTP ausgeführt werden können. Durch die Angabe der Technologie können passende FAIR-DO-OpTPAs für die Fähigkeiten der jeweiligen Umgebung ausgewählt werden. Diese Zusammenhänge werden abstrakt in Abbildung 4.2 abgebildet.

Ein Beispiel dafür ist ein FAIR-DO-OpTP für eine HTTP-Anfrage, die einen Adapter für die Ausführung in Python, eine für die Ausführung in JavaScript und eine für die Ausführung in einem Docker-Container hat. Gleichermaßen könnten komplexe und rechenintensive Algorithmen in Java, C++, Rust, für Web Assembly oder in einer anderen Technologie implementiert und bereitgestellt werden. Die Auswahl des Adapters erfolgt anhand der verfügbaren Technologien sowie der Sicherheits- und Kapselungsbedürfnisse des ausführenden Systems. Die Ein- und Ausgabe der im FAIR-DO-OpTP spezifizierten Attribute und Rückgabewerte erfolgt technologieabhängig (z. B. für Docker Environment Variablen, für Python-Skripte) durch das ausführende System. Zur technischen Umsetzung von FAIR-DO-OpTPAs ist ein hochgradig modularer und erweiterbarer Ansatz notwendig, der die Integration verschiedener Technologien und Implementierungen sicher ermöglicht. Außerdem muss ein Toolkit (vergleiche GitHub-Actions [54]) bereitgestellt werden, welches das Erstellen und Ausführen von FAIR-DO-OpTPAs erleichtert, indem beispielsweise die Ein- und Ausgabe der Attributwerte durch JSON-Objekte ermöglicht wird. Die Entwicklung einer (hochskalierbaren) Ausführungsumgebung für die, in dieser Arbeit beschriebenen, Konzepte ist funktional verschieden von der Aufgabe IDORIS als zentraler Infrastrukturkomponente und muss daher im Rahmen zukünftiger Arbeiten betrachtet werden (siehe Ausblick 8.2).

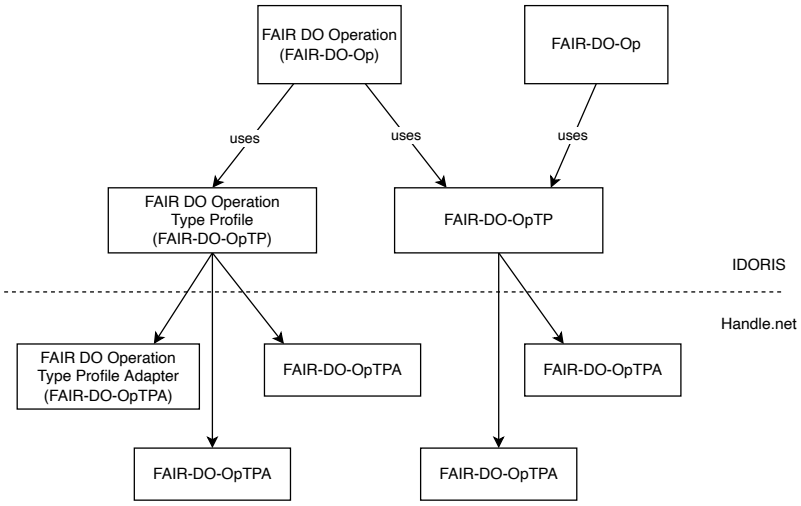

Abbildung 4.2.: Visualisierung der Zusammenhänge zwischen FAIR-DO-Ops, FAIR-DO-OpTPs und FAIR-DO-OpTPAs

4.3.3. Aufbau von FAIR Digital Object Operations

In Analogie zu der Typisierung von FAIR-DOs spezifizieren FAIR-DO-OpTPAs die konkreten Implementierungen einer Technologie (siehe BasicDataTypes), während FAIR-DO-OpTPs die Struktur und die Schnittstelle zur Wiederverwendbaren Nutzung dieser Technologien (siehe Typprofile) spezifizieren. Die Vererbung von FAIR-DO-OpTPs verbessert die Wiederverwendbarkeit und Erweiterbarkeit dieser Spezifikationen. FAIR-DO-Ops können nun mithilfe von FAIR-DO-OpTPs und FAIR-DO-OpTPAs typsicher spezifiziert und ausgeführt werden, wodurch die Maschinenprozessierbarkeit und die Wiederverwendbarkeit von Operationen auf FAIR-DOs sichergestellt wird. Die Spezifikation von einzelnen oder mehreren Ausführungsschritten erfolgt im Feld `execution` der jeweiligen FAIR-DO-Ops durch eine Liste von Operationsschritten (engl. OperationSteps). Jeder Operationsschritt kann entweder ein FAIR-DO-OpTP ausführen, eine FAIR Digital Object Operation (FAIR-DO-Op) ausführen oder selbst wiederrum eine Liste von Operationsschritten enthalten. Die Ausführung der Operationsschritte erfolgt standardmäßig sequenziell, kann jedoch auch asynchron erfolgen. Operationsschritte haben Zugriff auf alle Attribute der FAIR-DO-Ops und alle Eingabe- und Rückgabeattribute der Operationsschritte auf der gleichen oder einer höheren Ebene. Falls diese Attribute nicht mit den spezifizierten Attributen für das in dem jeweiligen OperationStep zu nutzende FAIR-DO-OpTP oder FAIR-DO-Op übereinstimmen, wird ein Attributmapping durchgeführt. Ein Attributmapping akzeptiert ein

Eingabeattribut und gibt ein Ausgabeattribut zurück. Alternativ kann ein Attributmapping einen statischen Wert zurückgeben, die Eingabe in einen statischen Wert einbetten oder ein spezielles Element aus einem wiederholbaren Attribut anhand eines Index extrahieren. Dadurch werden alle Attribute innerhalb einer FAIR-DO-Op über alle Ebenen hinweg zu den Attributen der einzelnen FAIR-DO-OpTP in einem Graphen miteinander verknüpft, wodurch komplette Arbeitsabläufe parallelisiert und validiert werden können. Es ist somit für Menschen und Maschinen nachvollziehbar, welcher Operationsschritt welches Attribut benötigt, welche Attribute zurückgegeben werden, welche Attribute spezifiziert werden müssen, aber nicht spezifiziert wurden und, wie die Daten bei der Ausführung der Operationsschritte verarbeitet werden.

4.4. Modellklassendiagramm

Die linke Hälfte des Modellklassendiagramms beschreibt die Metadaten, die alle von der abstrakten `GenericIDORISEntity`-Klasse vererbt werden. Im Anwendungsbeispiel (in Sektion 5.3) wird dies deutlich. Auf der rechten Seite sind die in den vorhergehenden Sektionen eingeführten Konzepte in Form von Klassen abgebildet. Im Folgenden werden die wichtigsten Klassen zusammengefasst:

- **`GenericIDORISEntity`:** Diese Klasse sorgt für konsistente Metadaten in allen erbenden Klassen und ermöglicht einfache Erweiterung durch Vermeidung von Redundanzen.
- **`DataType`:** Diese Klasse vereinheitlicht den Zugriff auf BasicDataTypes und Typprofile, was von den Attributen genutzt wird. Genaueres wird in Sektion 4.1 beschrieben.
- **`BasicDataType`:** BasicDataTypes spezifizieren die elementare Syntax von Werten durch reguläre Ausdrücke und wurden größtenteils von den PID-BasicInfoTypes (siehe 2.2) übernommen. Die wenigen Änderungen sind in Sektion 4.1 beschrieben.
- **`TypeProfile`:** Mithilfe von Typprofilen können komplexe Strukturen sowohl von FAIR-DOs als auch von einzelnen Werten in FAIR-DOs spezifiziert werden. Das ausführliche Modell ist in 4.1 näher beschrieben.
- **`Attribute`:** Attribute werden zur eindeutigen Referenzierung von Datentypen verwendet und reichern diese um kontextspezifische semantische Informationen an. Diese Klasse wurde in Sektion 4.1 eingeführt und wird in Sektion 4.3 verwendet.
- **`Operation`:** Diese Klasse spezifiziert FAIR-DO-Ops. Dazu wird auf genau ein Attribut, auf dem die jeweilige FAIR-DO-Op ausführbar sein soll, verwiesen. Außerdem werden mögliche Umgebungsvariablen und die Ausgaben sowie die einzelnen Operationsschritte verwiesen. Für mehr Details siehe 4.3.
- **`OperationStep`:** Operationsschritte bilden jeweils einen von möglicherweise vielen Operationsschritten einer Operation ab. In einem Operationsschritt wird daher auf die zu nutzende Technologie (FAIR-DO-OpTP), eine auszuführende Operation verwiesen oder gar mehrere Operationsschritte verschachtelt. Außerdem werden Eingabeattribute und Ausgabeattribute mithilfe von `AttributeMappings` spezifiziert. Genaueres ist in Sektion 4.3.3 beschrieben.
- **`AttributeMapping`:** Ein Attribut-Mapping wandelt ein Eingabeattribut und/oder einen statischen Wert in ein Ausgabeattribut um. Dies wird zur Spezifikation von `OperationSteps` verwendet (siehe 4.3.3)
- **`OperationTypeProfile`:** Wie in Sektion 4.3.1 beschrieben wird, spezifizieren FAIR-DO-OpTPs die Verwendung einer Technologie. Es werden Ein- und Ausgabeattribute sowie die FAIR-DO-OpTPA spezifiziert. FAIR-DO-OpTPs sind FAIR-DOs, sodass diese lediglich mit einer PID referenziert werden und somit keine eigene Klasse in IDORIS benötigen. Weitere Informationen über FAIR-DO-OpTPAs sind in Abschnitt 4.3.2 vorhanden.

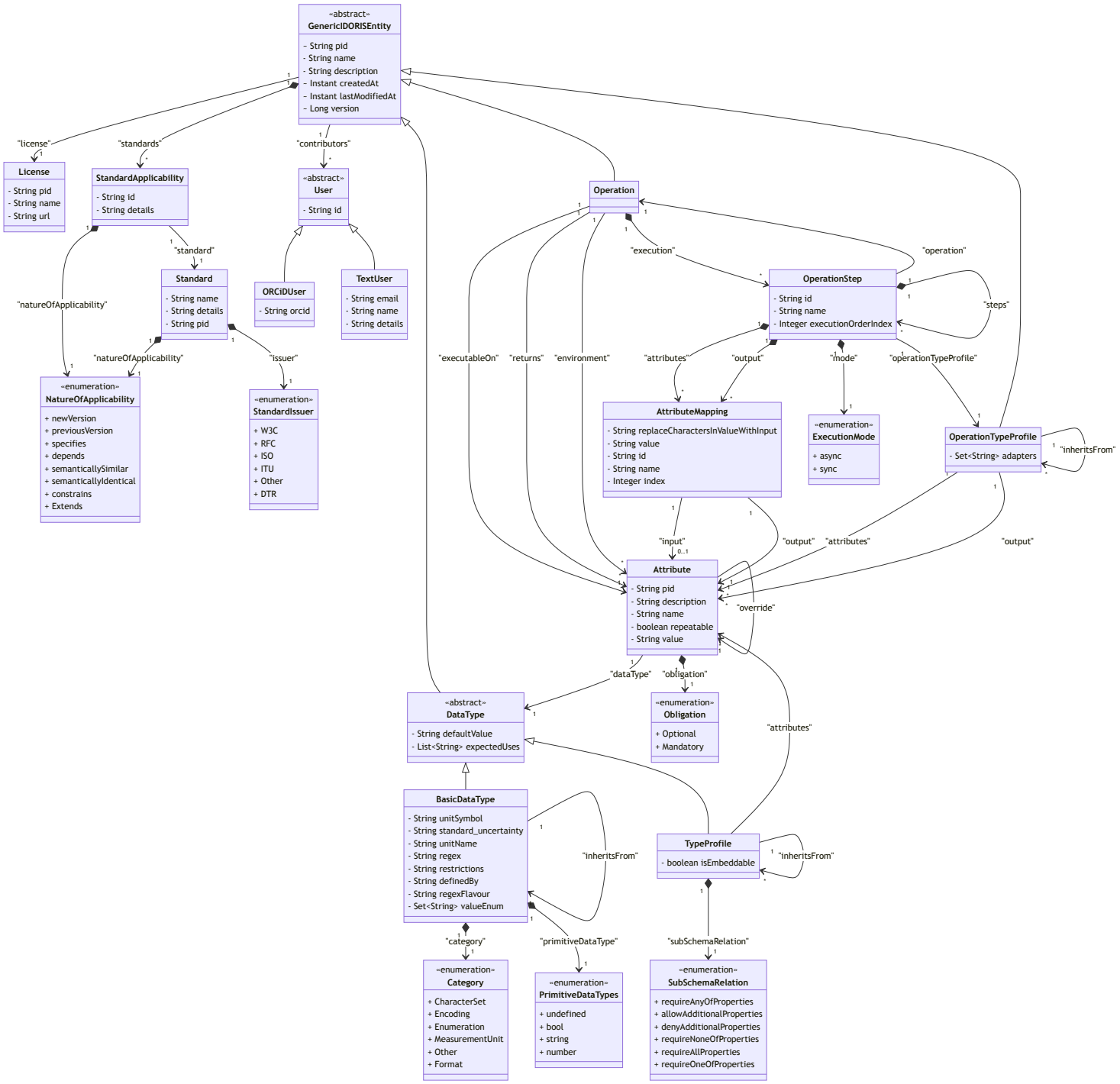

Abbildung 4.3.: Modellklassendiagramm des neuen Datenmodells

4.5. Software-Architektur

Für die Implementation von IDORIS wurde die stark und statisch typisierte Programmiersprache Java in der LTS-Version 21 gewählt [55, 56]. Java ist eine der am weitesten verbreiteten Programmiersprachen und wird dementsprechend in vielen Projekten und Organisationen (darunter die Abteilung DEM am SCC des KIT) verwendet. Aufgrund der starken Ausprägung objektorientierter Prinzipien, der Ausgereiftheit der Sprache und existierenden Vorkenntnissen in Java und dem Spring-Framework wurde Java als Programmiersprache gewählt.

Das Spring-Framework [57, 58] ist eine umfassende Sammlung von Werkzeugen zur Entwicklung von Java-Anwendungen und bietet viele Features, die die Entwicklung von Microservices wie IDORIS erleichtern. Darunter sind verschiedene Sammlungen für Web-APIs, ORM, Dependency-Injection, verschiedene Sicherheitsmechanismen (z. B. Authentifizierung und Autorisierung) und viele mehr. Spring wurde wegen des großen und zuverlässigen verfügbaren Funktionsumfangs, der Integration in viele andere Technologien, der Unterstützung von Java als Programmiersprache und der Vorkenntnisse gewählt. Für IDORIS wurden Spring Boot [59], Spring MVC [60], Spring Data REST [61, 62], Spring HATEOAS [63], Spring Data Neo4j [64] sowie der HAL REST Explorer [65] und Lombok [66] verwendet.

In den vorhergehenden Sektionen und im Modellklassendiagramm (siehe Sektion 4.4) ist die hohe Anzahl von Beziehungen zwischen den Entitäten zu erkennen. Gerade bei der Auflösung der Mehrfachvererbungen für Typprofile und FAIR-DO-OpTPs, der Validierung dieser Vererbungshierarchien und der Validierung von FAIR-DO-Ops mit den hoch verschachtelten Strukturen ist eine hohe Anzahl von Anfragen notwendig. Zukünftig soll es zudem möglich sein, automatisiert Pfade zwischen zwei Datentypen zu erkennen, Workflows mit FAIR-DO-Ops zu erstellen und die Umsetzbarkeit dieses Workflows zu prüfen. Die Datenstruktur, die hohe Anzahl von Beziehungen und die Vielzahl von Graph-Operationen sind mit speziell dafür optimierten Graphdatenbanken einfacher lösbar.

Neo4j [67] ist eine native und hochperformante Graphdatenbank, die mit der eigenen Abfragesprache "Cypher" [68] intuitiv zu bedienen ist und eine Spring Data Integration bietet [64]. Diese Eigenschaften unterscheiden Neo4j von anderen etablierten Graphdatenbanken, wie beispielsweise Apache Jena [69], die komplexe SPARQL Abfragen auf einen RDF oder OWL-Graphen in einem weniger performanten Triplestore [70, 71, 72, 73, 74] speichern. Das Labeled-Property-Graph-Modell von Neo4j ist im Vergleich zu RDF-Graphen flexibler, da es mehrere (auch gleiche) Relationen zwischen zwei Entitäten ermöglicht, Relationen mit Parametern versehen werden können und Entitäten mit verschiedenen Labels intuitiv Java-Klassen zugeordnet werden können. Spring Data Neo4j [64] nutzt diese Eigenschaften aus und bietet eine einfache Integration von Neo4j in Spring-Anwendungen. Für besonders komplexe Abfragen wurden in der Neo4j-Datenbank die Plugins APOC [75] und Graph Data Science [76] installiert. Die Dokumentation von Neo4j [77, 78] und Spring Data Neo4j [64, 79] ist sowohl umfangreich als auch gut verständlich, sodass das nötige Wissen schnell erarbeitet werden konnte. Aus diesen Gründen wurde Neo4j als Datenbank für IDORIS gewählt. Relationale Datenbanken kamen aufgrund der hohen Anzahl von Beziehungen und der komplexen Struktur der Daten nicht in Frage, da die Umsetzung der Beziehungen in Tabellen und die Abfragen sowie die Umsetzung möglicher Erweiterungen auf diesen Tabellen sehr aufwändig, ineffizient und um einiges komplizierter gewesen wären.

5. Implementierung

IDORIS ist eine Spring-Boot-Anwendung und nutzt die vom Spring-Framework bereitgestellte komponentenbasierte Architektur. Spring Boot übernimmt als komponentenorientierte Middleware die Konfiguration und Verwaltung der Anwendung, das Bean-Lifecycle-Management, die Dependency-Injection und die Integration der Spring Data Repositories. Daher gliedert sich die Implementierung in die Pakete `domain` für das Datenmodell, `dao` für die Spring Data REST Repositories, `validators` für die Validatoren sowie das Visitor-Pattern und `web` für Web-MVC-Endpunkte. Konfigurationen sind im Paket `configuration` zu finden.

5.1. Spring Data Rest

Der Aufbau des Datenmodells für IDORIS wurde bereits in Kapitel 4 beschrieben und als Modellklassendiagramm (siehe Abbildung 4.3) visualisiert. Für die Implementierung dieses Datenmodells wurde Spring Data REST [62] mit Spring Data Neo4j als Datenbanktreiber verwendet. Spring Data REST ermöglicht die einfache Erstellung von REST-APIs, die automatisch aus den Spring Data Repositories generiert werden. Dazu müssen, wie bei jedem Spring Data Projekt, zunächst die Modellklassen mit den entsprechenden Annotationen des Datenbanktreibers versehen werden. Spring Data Neo4j verwendet hierfür die Annotationen `@Node("Labelname")`, `@Relationship(value, direction)` und `@Id` für die Modellklassen mit den entsprechenden Labels für den Graphen, die Beziehungen mit Namen sowie Richtung und Identifikation. Besonders an dem Labeled-Property-Graph-Modell von Neo4j ist, dass die Beziehungen zwischen zwei Knoten Parameter haben können, die in Java als separates Objekt modelliert werden müssen. Dazu wird die Annotation `@RelationshipProperties` verwendet, die auf eine eigene Modellklasse verweist (siehe Klasse `StandardApplicability` in Anhang A.1).

```
1 @NoRepositoryBean
2 public interface IAbstractRepo<T, ID> extends
3     Neo4jRepository<T, ID>,
4     ListCrudRepository<T, ID>,
5     PagingAndSortingRepository<T, ID> {}
```

Listing 5.1: Sourcecode für IAbstractRepo

Zur Abfrage dieses Datenmodells werden Spring Data Repositories mit den entsprechenden Methoden in Interfaces deklariert und von Spring Data Neo4j automatisch implementiert. Dazu wurde in IDORIS zunächst ein abstraktes Repository-Interface `IAbstractRepo` erstellt, welches zur Reduktion von Redundanzen alle wichtigen Interfaces zusammenführt (siehe Listing 5.1). `IAbstractRepo` kann als Generic nicht selbst instanziiert werden, was durch die Annotation `@NoRepositoryBean` für Spring sichtbar gemacht wird. `Neo4jRepository` ist das Interface von Spring Data Neo4j, welches die Standardmethoden für die Datenbankabfragen an Neo4j-Datenbanken bereitstellt [64]. `ListCrudRepository` und `PagingAndSortingRepository` sind Interfaces von Spring Data, die grundlegende CRUD-, Sortier- und Pagingfähigkeiten ermöglichen. Dadurch sind bereits

viele Standardmethoden für den Umgang mit dem Datenmodell und der Datenbank mit sehr wenig Aufwand implementiert.

```
1 @RepositoryRestResource(collectionResourceRel = "typeProfiles", path = "typeProfiles")
2 public interface ITypeProfileDao extends IAbstractRepo<TypeProfile, String> {
3     @Query("MATCH (d:TypeProfile {pid: $pid})-[:inheritsFrom*]->(typeProfile:TypeProfile)
      return typeProfile")
4     Iterable<TypeProfile> findAllTypeProfilesInInheritanceChain(String pid);
5 }
```

Listing 5.2: Sourcecode für ITypeProfileDao

Listing 5.2 zeigt ein Beispiel für ein Repository, das `IAbstractRepo` nutzt. Durch die Annotation `@RepositoryRestResource` wird Spring Data Rest angewiesen, die Methoden des Interfaces als REST-API bereitzustellen. Der erzeugte REST-API Endpunkt kann, allein durch die Erweiterung von `IAbstractRepo`, bereits CRUD-Operationen mit Paging und Sorting auf den `TypeProfile`-Objekten anbieten und durchführen. Für spezielle Abfragen, die nicht durch die Standardmethoden abgedeckt sind, kann mithilfe der `@Query`-Annotation eine eigene Cypher-Query spezifiziert werden, welche Parameter und Ergebnisse anhand der Methodensignatur typisiert. Alle von Spring-Data-REST generierten Endpunkte sind HATEOAS-konform [80, 63], liefern ALPS-Informationen [81, 63] und HTTP-OPTIONS für eine bessere Navigierbarkeit. HATEOAS unterstützt sowohl Mensch als auch Maschine bei der Nutzung und Navigierung durch die API indem mögliche Aktionen basierend auf den angezeigten Daten ausgegeben werden [80]. Mit ALPS können weitere semantische Informationen über eine API und deren Datenschemata bereitgestellt werden [81].

```
1  @RepositoryRestController
2  public class OperationController {
3      @Autowired
4      IOperationDao operationDao;
5
6      @GetMapping("v1/operations/{pid}/validate")
7      public ResponseEntity<?> validate(@PathVariable("pid") String pid) {
8          Operation operation = operationDao.findById(pid).orElseThrow();
9          SubSchemaRelationValidator validator = new SubSchemaRelationValidator();
10         ValidationResult result = operation.execute(validator);
11         return ResponseEntity.ok(result);
12     }
13 }
```

Listing 5.3: Sourcecode für OperationController

Falls zusätzliche Logik für die Abfragen benötigt wird, können eigene Controller auf Basis von Spring Web MVC erstellt werden, welche einzelne Pfade aus den Repositorien überschreiben oder hinzufügen können. Ein solcher Controller ist in Listing 5.3 abgebildet. Im Gegensatz zu herkömmlichen Spring MVC Controllern werden diese mit der Annotation `@RepositoryRestController` anstelle von `@RestController` versehen, um die Integration mit Spring Data Rest zu ermöglichen. Ansonsten können alle Funktionalitäten von Spring MVC genutzt und die benötigten Ressourcen (z. B. Repositorien) über Dependency-Injection eingebunden (`@Autowired`) werden. Ein Nachteil, gegenüber den von Spring Data REST automatisch generierten Endpunkten, ist die fehlende automatische Bereitstellung von HATEOAS-Informationen und ALPS-Dokumentation. Mithilfe von Spring HATEOAS können diese Informationen manuell hinzugefügt werden [63].

```java
@RepositoryRestController
public class TypeProfileController {
    @Autowired
    ITypeProfileDao typeProfileDao;
    [...]
    @GetMapping("typeProfiles/{pid}/inheritedAttributes")
    public ResponseEntity<?> getInheritedAttributes(@PathVariable("pid") String pid) {
        Iterable<TypeProfile> inheritanceChain =
            typeProfileDao.findAllTypeProfilesInInheritanceChain(pid);
        List<EntityModel<Attribute>> attributes = new ArrayList<>();
        inheritanceChain.forEach(typeProfile -> {
            typeProfileDao
            .findById(typeProfile.getPid())
            .orElseThrow()
            .getAttributes()
            .forEach(profileAttribute -> {
                EntityModel<Attribute> attribute = EntityModel.of(profileAttribute);
                attribute.add(
                    linkTo(IDataTypeDao.class)
                        .slash("api")
                        .slash("dataTypes")
                        .slash(profileAttribute.getDataType().getPid())
                        .withRel("dataType"));
                attributes.add(attribute);
            });
        });

        CollectionModel<EntityModel<Attribute>> resources=CollectionModel.of(attributes);
        resources.add(
            linkTo(TypeProfileController.class)
            .slash("api")
            .slash("typeProfiles")
            .slash(pid)
            .slash("inheritedAttributes")
            .withSelfRel());
        resources.add(
            linkTo(ITypeProfileDao.class)
            .slash("api")
            .slash("typeProfiles")
            .slash(pid)
            .withRel("typeProfile"));
        return ResponseEntity.ok(resources);
    }
    [...]
}
```

Listing 5.4: Ausschnitt des Quellcodes des `TypeProfileController`

Dies wird in der Methode `getInheritedAttributes` in Listing 5.4 demonstriert. Zunächst wird mithilfe des `TypeProfileDao` eine Liste aller Attribute in der Vererbungshierarchie erstellt (siehe Zeilen 8 bis 16). Jedem Attribut wird ein HATEOAS-Link zu dem zugehörigen Datentyp hinzugefügt (siehe Zeilen 13 bis 15). Diese Liste wird in ein `CollectionModel` verpackt, welchem wiederum Verweise zu der eigenen Abfrage (die sogenannte `self`-Relation) und dem Typprofil, auf dem diese Abfrage durchgeführt wurde, hinzugefügt werden (siehe Zeilen 18 bis 22). Das Verpacken in `EntityModel` und `CollectionModel` ist für eine korrekte Darstellung (z. B. im HAL-Explorer) notwendig. Listing 5.5 zeigt ein beispielhaftes Ergebnis dieser Abfrage mit den entsprechenden HATEOAS-Verweisen auf die Datentypen der Attribute, das Typprofil (in diesem Fall ein Key-Value-Paar) und die eigene Abfrage. Diese Attribute werden dem abgefragten Typprofil von einem Eltern-Typprofil vererbt.

```
1  {
2      "_embedded": {
3          "attributes": [
4              {
5                  "name": "Value",
6                  "description": "The value of a key-value pair. This field is mandatory.",
7                  "repeatable": false,
8                  "obligation": "Mandatory",
9                  "value": null,
10                 "_links": {
11                     "dataType": {
12                         "href": "http://localhost:8095/api/dataTypes/3f6852ac"
13                     }
14                 }
15             }, {
16                 "name": "Key",
17                 "description": "The key of a key-value pair. This field is mandatory.",
18                 "repeatable": false,
19                 "obligation": "Mandatory",
20                 "value": null,
21                 "_links": {
22                     "dataType": {
23                         "href": "http://localhost:8095/api/dataTypes/3f6852ac"
24                     }
25                 }
26             }
27         ]
28     },
29     "_links": {
30         "self": {
31             "href": "http://localhost:8095/api/typeProfiles/0c7f6adf/inheritedAttributes"
32         },
33         "typeProfile": {
34             "href": "http://localhost:8095/api/typeProfiles/0c7f6adf"
35         }
36     }
37 }
```

Listing 5.5: Beispielergebnis des Codes in Listing 5.4 mit HATEOAS Verweisen

5.2. Validierungssystem

Das Visitor-Pattern [84, 83] ist ein im Compilerbau häufig verwendetes Verhaltensmuster zur Trennung von Modellklassen und Logik. Es ermöglicht Änderungen und Erweiterungen der Logik ohne Modifikationen der Modellklassen, sodass das Open-Closed Prinzip, die Dependency-Inversion Regel und Regeln der Clean Architecture erfüllt werden [46]. Im Compilerbau wird das Visitor-Pattern für Syntax- und Semantikanalysen, Optimierungen sowie die Übersetzung in Bytecode verwendet [85, 86, 87]. Eine Visitorklasse setzt dabei jeweils eine Regel um und traversiert rekursiv durch den Syntaxbaum, dabei wird mithilfe des Double-Dispatch [88] (siehe Listing 5.6) die für die jeweilige Klasse passende Methode ausgeführt. Ein entsprechendes Klassendiagramm ist in Abbildung 5.1 auffindbar.

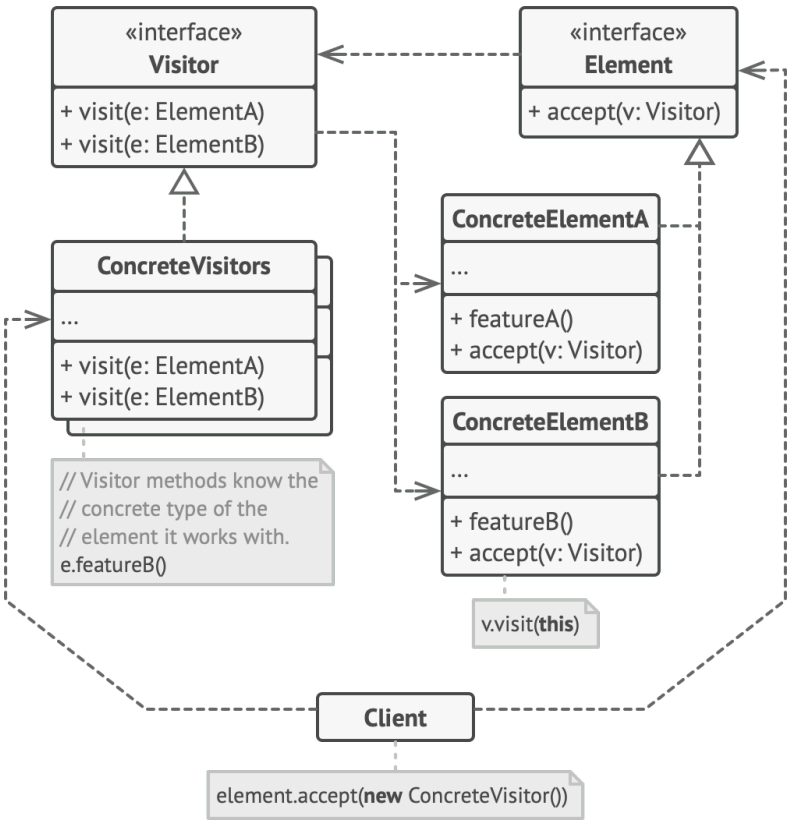

Abbildung 5.1.: Abstraktes Klassendiagramm für das Visitor-Pattern [82, 83]

In IDORIS wird das Visitor-Pattern für die Validierung von Datentypen, Attributen, Attribut-Mappings, FAIR-DO-Ops, FAIR-DO-OpTPs und Operationsschritten verwendet. Die Vielzahl an Konzepten, Regeln und Abfragen in Kapitel 4 zeigt deutlich die Notwendigkeit für eine klare Trennung von Modell und Logik. Bislang wird das Visitor-Pattern ausschließlich zur Validierung verwendet, jedoch ist eine Erweiterung auf andere Anwendungsfälle aufgrund der Modularität einfach umsetzbar. Jeder konkrete Visitor (bislang `InheritanceValidator`, `SubSchemaRelationValidator`, `SyntaxValidator`) repräsentiert einen Teil der Regeln und wird von einer abstrakten Visitorklasse abgeleitet. Die abstrakte Visitorklasse enthält eine abstrakte Methode für jede Modellklasse, die von den konkreten Visitorklassen überschrieben werden muss (siehe Klassendiagramm in Abbildung 5.1). Außerdem ist in der `Visitor`-Klasse ein temporäres Caching sowie eine Kreiserkennung implementiert, um Endlosschleifen zu verhindern und Mehrfachabfragen zu vermeiden. Diese Architektur ermöglicht eine einfache Erweiterung neuer Regeln, ohne die bestehenden Klassen zu verändern.

```
1 [...]
2 public class Attribute extends VisitableElement {
3     [...]
4     @Override
5     protected <T> T accept(Visitor<T> visitor, Object... args) {
6         return visitor.visit(this, args);
7     }
8 }
```

Listing 5.6: Ausschnitt aus der Attributklasse mit dem Double-Dispatch-Aufruf

Alle vom Visitor-Pattern besuchbaren Klassen müssen die abstrakte Klasse `VisitableElement` erweitern und die Methode `accept` implementieren. Diese Methode erhält einen generischen Visitor und die Argumente für diesen und ruft auf diesem, gemäß dem Double-Dispatch-Prinzip [88], die zugehörige Methode auf. Listing 5.6 stellt dies beispielhaft anhand der Klasse `Attribute` dar, deren `accept`-Methode auf der Implementierung des Visitors die generische Methode `T visit(Attribute attribute, Object... args)` aufruft.

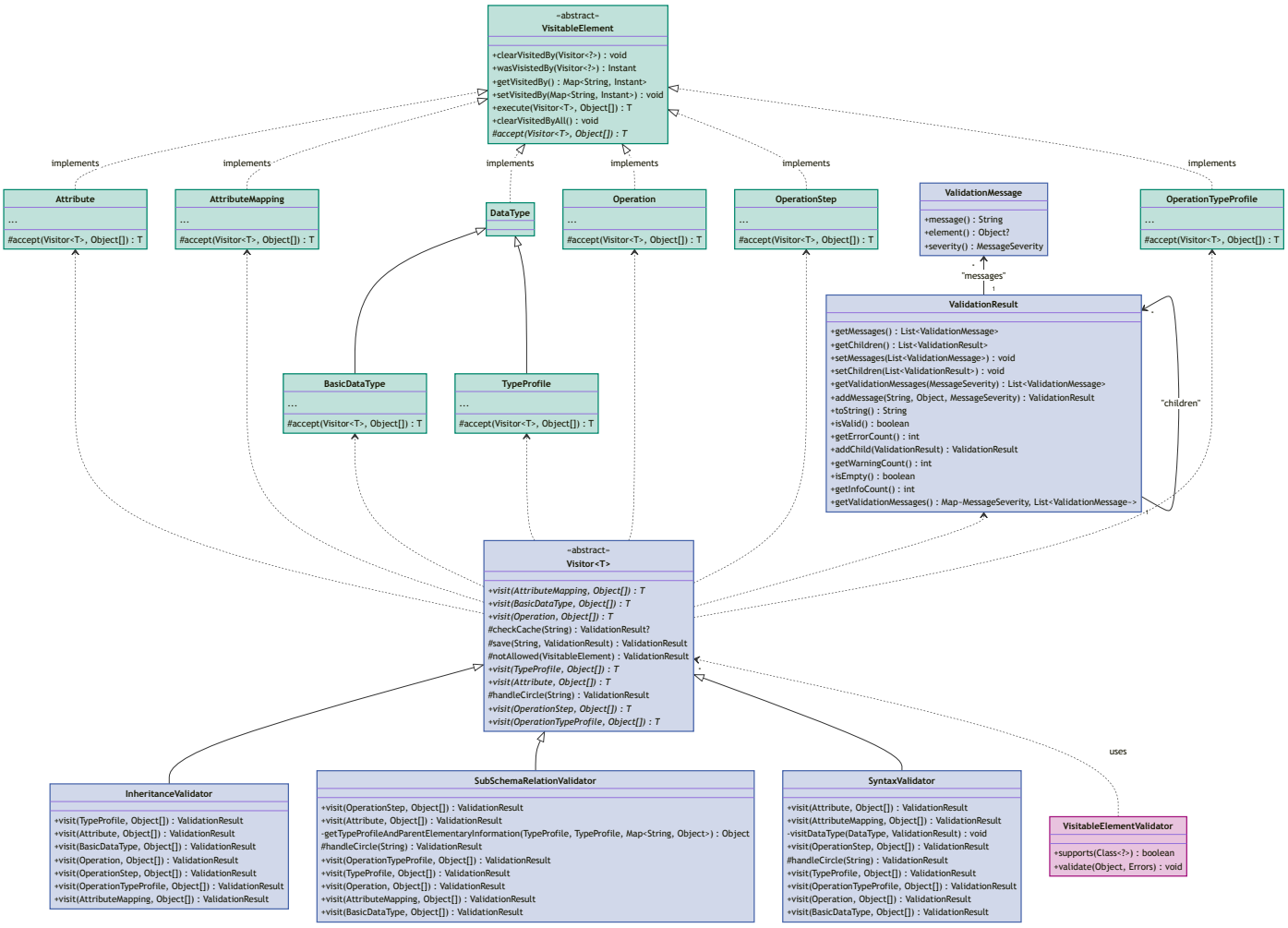

Abbildung 5.2.: Umsetzung des Visitor-Patterns in IDORIS

Das Klassendiagramm in Abbildung 5.2 zeigt die Umsetzung des Visitor-Patterns in IDORIS mit allen für die Validatoren wichtigen Komponenten. Hierbei wird die Trennung von Datenmodell (grün), Visitor-Struktur (blau) und der Nutzung der Visitoren (lila) deutlich. Die Dependency-Regel, nach der Modellklassen nicht von der Logik abhängen dürfen, wird eingehalten, was auf Basis der gepunkteten Abhängigkeitspfeile zu erkennen ist. Die einzige Abhängigkeit des Modells gegenüber der Logik besteht in der Implementierung der `accept`-Methode (siehe Listing 5.6) und der Kenntnis der generischen und abstrakten Visitorklasse. Jeder Visitor muss für jede Modellklasse, die von der abstrakten `VisitableElement`-Klasse erbt, eine Methode implementieren, welche stark auf diese Modellklasse zugeschnitten ist. Dadurch wird die Abhängigkeit der Logik von den Modellklassen auf ein Minimum reduziert und die Erweiterbarkeit und Wartbarkeit durch die Visitoren gewährleistet.

Listing 5.7 zeigt einen Ausschnitt des Syntaxvalidators, welcher die Korrektheit und Vollständigkeit der einzelnen Objekte überprüft. Für jede Klasse, die von `VisitableElement` erbt, existiert eine Methode, die die spezifischen Regeln für diese Klasse überprüft. So wird beispielsweise für die Klasse `Attribute` überprüft, ob

ein Name, eine Beschreibung, ein Datentyp und eine Obligation vorhanden sind. Erkannte Fehler werden zusammen mit ihrem Schweregrad (Auswahl aus `ERROR`, `WARNING`, `INFO`), einer Nachricht und dem fehlerhaften Element in eine `ValidationMessage` verpackt und in einem `ValidationResult` gespeichert. Sowohl der Datentyp, auf den das Attribut verweist, als auch ein mögliches überschriebenes Attribut, werden durch rekursive Aufrufe validiert und dem Validierungsergebnis als Kind-Elemente übergeben. Zur Vermeidung von Mehrfachauswertungen und Endlosschleifen wird ein temporäres Caching verwendet, welches die bereits validierten Elemente speichert und bei erneuter Abfrage direkt zurückgibt. Die Validierungsergebnisse werden in einem JSON-Array über die API zurückgegeben und sind durch die menschenlesbaren Fehlertexte sowie dem Einfügen des jeweiligen betroffenen Elements einfach verständlich (siehe Listing 5.8).

```
1  public class SyntaxValidator extends Visitor<ValidationResult> {
2      @Override
3      public ValidationResult visit(Attribute attribute, Object... args) {
4          ValidationResult result;
5          if ((result = checkCache(attribute.getPid())) != null) return result;
6          else result = new ValidationResult();
7
8          if (attribute.getName() == null || attribute.getName().isEmpty()) {
9              result.addMessage("For better human readability and understanding, " +
10                 "you MUST provide a name for the attribute.", attribute, ERROR);
11         }
12
13         if (attribute.getDescription() == null || attribute.getDescription().isEmpty()) {
14             result.addMessage("For better human readability and understanding, " +
15                 "you SHOULD provide a description for the attribute.", attribute,WARNING);
16         }
17
18         if (attribute.getDataType() == null) {
19             result.addMessage("You MUST provide a data type for the attribute.",
20                 attribute, ERROR);
21         } else {
22             result.addChild(attribute.getDataType().execute(this, args));
23         }
24
25         if (attribute.getObligation() == null) {
26             result.addMessage("You MUST provide an obligation for the attribute. " +
27                 "The default value is 'Mandatory'.", attribute, ERROR);
28         }
29
30         if (attribute.getOverride() != null) {
31             result.addChild(attribute.getOverride().execute(this, args));
32         }
33
34         return save(attribute.getPid(), result);
35     } [...]
36 }
```

Listing 5.7: Ausschnitt des Quellcodes des `SyntaxValidator`

```
[
  {
    "codes": [
      "SyntaxValidator.BasicDataType",
      "SyntaxValidator"
    ],
    "arguments": [
      {
        "ERROR": [
          {
            "message": "A format data type MUST have a regex.",
            "element": {
              "type": "BasicDataType",
              "name": "TEST",
              "primitiveDataType": "string",
              [...]
            },
            "severity": "ERROR"
          }
        ]
      },
      {
        "WARNING": [
          {
            "message": "For better human readability and understanding, you SHOULD provide a description for the data type.",
            "element": {[...]},
            "severity": "WARNING"
          },
          {
            "message": "For better human readability and understanding, you SHOULD provide a list of expected uses for the data type.",
            "element": {[...]},
            "severity": "WARNING"
          }
        ]
      }
    ],
    "defaultMessage": "Validation with SyntaxValidator failed.",
    "objectName": "BasicDataType",
    "code": "SyntaxValidator"
  }
]
```


Listing 5.8: Ausschnitt einer beispielhaften Fehlermeldung bei der Validierung

```java
@AllArgsConstructor
public class VisitableElementValidator implements Validator {
    private ApplicationProperties applicationProperties;

    @Override
    public boolean supports(Class<?> clazz) {
        return VisitableElement.class.isAssignableFrom(clazz);
    }

    @Override
    public void validate(Object target, Errors errors) {
        VisitableElement visitableElement = (VisitableElement) target;
        Set<Visitor<ValidationResult>> validators = Set.of(
                new SubSchemaRelationValidator(),
                new SyntaxValidator(),
                new InheritanceValidator()
        );

        validators.forEach(validator -> {
            ValidationResult validationResult = visitableElement.execute(validator);
            var validationMessages = validationResult.getValidationMessages()
                    .entrySet()
                    .stream()
                    .filter(e -> e.getKey()
                        .isHigherOrEqualTo(applicationProperties.getValidationLevel()))
                    .filter(e -> !e.getValue().isEmpty())
                    .toArray();
            if (validationMessages.length == 0) return;

            switch (applicationProperties.getValidationPolicy()) {
                case STRICT -> {
                    errors.rejectValue(null, validator.getClass().getSimpleName(),
                        validationMessages, "Validation with " +
                        validator.getClass().getSimpleName() + " failed.");
                }
                case LAX -> {
                    if (!validationResult.isValid()) {
                        errors.rejectValue(null, validator.getClass().getSimpleName(),
                            validationMessages, "Validation with " +
                            validator.getClass().getSimpleName() + " failed.");
                    }
                }
            }
        });
    }
}
```

Listing 5.9: Ausschnitt des Quellcodes des `VisitableElementValidator`

In Abbildung 5.2 ist der `VisitableElementValidator` (in lila) vorhanden. Diese Klasse bildet die Schnittstelle zu Spring Data REST, indem es das `Validator`-Interface aus Spring-Boot [59] implementiert. Die Methode `supports` prüft, ob eine Klasse von diesem Validator unterstützt wird, indem sie prüft, ob die Klasse von `VisitableElement` erbt. In der Methode `validate` wird zunächst das Element zu einem `VisitableElement` gecastet (siehe Listing 5.9 Zeile 12). Anschließend wird es mit einer Menge von Validatoren sequenziell validiert. (siehe Listing 5.9 Zeilen 13–19) Jeder Validator bewegt sich eigenständig rekursiv durch die Objektstruktur und prüft die Regeln, die für das jeweilige Element gelten. Dabei werden die Ergebnisse in einer Baumstruktur rekursiv zurückgegeben, ausgewertet und aggregiert, damit bei Bedarf Fehlermeldungen generiert werden können. Fehler können verschiedene Schweregrade haben (`ERROR`, `WARNING`, `INFO`), haben immer eine menschenlesbare Fehlermeldung und können das fehlerhafte Element enthalten. In der `application.properties`-Konfigurationsdatei können das niedrigste Fehlermeldungsniveau und die Fehlerbehandlungspolitik (STRICT oder LAX) festgelegt werden. Die Fehlerbehandlungspolitik bestimmt, ob bei einem Fehler die gesamte Validierung fehlschlägt oder ob nur eine Warnung ausgegeben wird. Nachdem ein Element von einem Validator validiert wurde (siehe Listing 5.9 Zeile 24) werden Nachrichten gefiltert, die unter dem eingestellten Validierungsniveau liegen oder leere Nachrichten enthalten (Zeile 24–29). Falls keine Nachrichten vorhanden sind, wird die Validierung für diesen Validator durch das `return`-Statement in der Lambda-Expression beendet, sodass die anderen Validatoren ausgeführt werden können. Andernfalls wird basierend auf der Fehlerbehandlungspolitik entschieden, ob ein Fehler ausgegeben wird (siehe Listing 5.9 Zeile 30–43). Die Fehlermeldung enthält den Namen des Validators, die Nachrichten und eine allgemeine Fehlermeldung. Für die Fehlerbehandlungsstrategie `STRICT` schlägt bei jedem Fehler auf beliebigem Niveau die gesamte Validierung fehl und alle Nachrichten werden ausgegeben. Für die Fehlerbehandlungsstrategie `LAX` schlägt die Validierung nur fehl, wenn mindestens ein Fehler des Niveaus `ERROR` auftritt. Der Algorithmus terminiert erst nach der vollständigen Validierung des Elementes mit allen Validatoren. Das Ausgeben von Warnungen und Informationen bei gleichzeitigem Speichern des Objektes ist mit dem Validierungsmechanismus von Spring nicht möglich.

```
1  @Component
2  public class RepositoryRestConfig implements RepositoryRestConfigurer {
3      [...]
4      @Override
5      public void configureValidatingRepositoryEventListener(
       ValidatingRepositoryEventListener v) {
6          v.addValidator("beforeSave",
7              new VisitableElementValidator(applicationProperties));
8          v.addValidator("beforeCreate",
9              new VisitableElementValidator(applicationProperties));
10         v.addValidator("beforeLinkSave",
11             new VisitableElementValidator(applicationProperties));
12     } [...]
13 }
```

Listing 5.10: Ausschnitt der `RepositoryRestConfig` zur Validator-Konfiguration

Die Konfiguration der Validatoren erfolgt in der Klasse `RepositoryRestConfig` (siehe Listing 5.10). Durch die Implementierung des `RepositoryRestConfigurer`-Interfaces kann die Methode `configureValidatingRepositoryEventListener` überschrieben werden, um die Validatoren für die verschiedenen Ereignisse zu konfigurieren. In diesem Fall wird der `VisitableElementValidator` für das Speichern, Erstellen und Verknüpfen von Objekten konfiguriert. Falls eine Validierung Fehler produziert, wird das Objekt nicht gespeichert

und alle Fehlermeldungen aller Validatoren ausgegeben. Die Fehlermeldungen (siehe Listing 5.8) enthalten den Fehlercode, die Fehlermeldung, das Objekt, das den Fehler verursacht hat, und die Schwere des Fehlers.

5.3. Beispiel für die Verwendung von IDORIS

Zur Demonstration der Funktionalität von IDORIS werden im Folgenden auszugsweise alle nötigen Schritte zur Erstellung einer FAIR-DO-Op für die Abfrage eines ORCiD-Profils beschrieben. Es soll aus einer ORCiD-URL (beispielsweise `https://orcid.org/0009-0005-2800-4833`) die ORCiD-Nummer extrahiert und in eine URL für die ORCiD-API eingefügt werden, sodass das Ergebnis zurückgegeben werden kann. An dieser Stelle ist nicht das IDORIS-interne Metadatenattribut `ORCiDUser`, sondern ein eigenständiger Datentyp für die Verwendung in FAIR-DOs gemeint. Dazu müssen zunächst die benötigten Datentypen, Attribute, OperationTypeProfiles, Operationen und OperationSteps erstellt werden. In der Datenbank werden am Ende etwa 60 Knoten und 100 Beziehungen erstellt. An dieser Stelle wird das grobe Vorgehen beschrieben und auf einige interessante HTTP-Anfragen und -Antworten im Anhang (siehe A.3) verwiesen.

5.3.1. Erstellen von Nutzern, Lizenzen und Standards

Bereits in der ePIC-DTR wird die Provenienz der Inhalte durch die Speicherung von Nutzern und Standards dokumentiert. IDORIS verbessert diese Funktionalität durch eine strengere Typisierung von Nutzern und Standards und die Einführung von Lizenzen.

Bei der Erstellung eines Benutzers muss zwischen einem `ORCiDUser` (enthält nur die ORCiD) und einem `TextUser` (enthält den Namen, eine E-Mail-Adresse und ein Feld für Details) unterschieden werden. Zukünftig sollen Nutzer automatisch beim Einloggen über ORCiD erstellt werden, sodass Informationen über die mitwirkenden Personen automatisch erfasst werden. Momentan müssen die Nutzer jedoch manuell unter Eingabe des Typen und der entsprechenden Parameter erstellt werden (siehe Listing A.5). Die Erstellung eines Nutzers erfolgt über einen POST-Request an den Endpunkt `/api/users` mit dem entsprechenden Nutzertyp und den Parametern. Die Antwort enthält den erstellten Nutzer mit einer Datenbank-internen ID und HATEOAS-Links (siehe Listing A.6).

Eine Lizenz besteht aus einem Namen und einer URL, die idealerweise auf SPDX zeigen sollte (siehe A.7). Die Antwort auf einen POST-Request an den Endpunkt `/api/licenses` enthält die erstellte Lizenz mit einer Datenbank-internen ID und HATEOAS-Links (siehe A.8).

Standards bestehen aus einem Namen, einem Issuer (der Organisation, die den Standard herausgibt, z. B. ISO. IEEE, …) und einem Feld für Details (siehe A.11). Die Antwort auf einen POST-Request an den Endpunkt `/api/standards` enthält den erstellten Standard mit einer Datenbank-internen ID und HATEOAS-Links (siehe A.12).

5.3.2. Erstellen von BasicDataTypes

BasicDataTypes können sowohl über den `/api/basicDataTypes` als auch `/api/dataTypes`-Endpunkt erstellt werden, da sie von der Klasse `DataType` erben. Aus diesem Grund muss, ähnlich zu den Nutzern, der Typ des Datentyps angegeben werden, um das Element entweder den Typprofilen oder BasicDataTypes zuzuweisen. Die Erstellung von BasicDataTypes wird von den Validatoren überprüft, um sicherzustellen, dass alle notwendigen Felder korrekt ausgefüllt sind. Falls dies nicht der Fall ist, antwortet IDORIS mit `406 - Not Acceptable` und erzeugt die Entität nicht (siehe A.16). Die Antwort auf eine gültige Anfrage (siehe A.3.2.2)

enthält den erstellten BasicDataType mit einer Datenbank-internen ID und HATEOAS-Links. Verweise auf andere Entitäten, wie Nutzer, Lizenzen und Standards, werden in Form HTTP-URL auf das jeweilige Element in der API abgebildet. Gleiches gilt auch für den Verweis auf einen anderen BasicDataType, von dem geerbt werden soll (siehe A.3.2.2). Das Feld `category` wird standardmäßig auf `Format` gesetzt, muss für eine Enumeration jedoch auf `Enumeration` geändert werden, damit entsprechende Regeln in den Validatoren wirksam werden (siehe A.3.2.3). Aus dem Schema der ePIC-DTR für PID-BasicInfoTypes (siehe A.2) wurden einige andere Kategorien und Felder für BasicDataTypes übernommen, die jedoch, mangels Anwendungsfall und Dokumentation, nicht weiter betrachtet werden.

5.3.3. Erstellen von Attributen

Bevor ein Typprofil oder FAIR-DO-OpTP erstellt werden kann, muss für jedes Feld ein Attribut erzeugt werden. Dazu müssen `POST`-Anfragen an den Endpunkt `/api/attributes` mit mindestens einem Namen, einer Referenz auf einen Datentyp, die Wiederholbarkeit und Obligation spezifiziert werden. Für die Wiederholbarkeit wird standardmäßig `false` gesetzt und für die Obligation `Mandatory`, sodass diese Felder bei der Erstellung für diese Werte nicht noch einmal explizit spezifiziert werden müssen. Idealerweise sollte außerdem eine Beschreibung hinzugefügt werden, um das spätere Verständnis des Attributs zu erleichtern. Zudem kann ein anderes Attribut mithilfe einer Referenz im `override`-Feld überschrieben werden, sofern der Datentyp von dem des zu überschreibenden Attributs erbt und das erbende Attribut gleich strikt oder strikter als das vererbende Attribut ist. Der `SyntaxValidator` überprüft die Vollständigkeit und syntaktische Korrektheit der Attribute, bevor sie in der Datenbank gespeichert werden. Falls ein Attribut nicht den Anforderungen entspricht, wird, wie bei den BasicDataTypes, die Anfrage mit einem `406 - Not Acceptable` einschließlich detaillierter Fehlermeldung beantwortet und das Attribut nicht erstellt. Die Antwort der `POST`-Anfrage enthält das erstellte Attribut, mit der automatisch generierten PID und verschiedenen HATEOAS-Verweisen unter anderem auf den Datentyp und das überschriebene Attribut (siehe A.3.3).

5.3.4. Erstellen von Typprofilen

Typprofile können sowohl über den `/api/typeProfiles` als auch `/api/dataTypes`-Endpunkt mithilfe der `POST`-Methode erstellt werden. Das Feld `type` muss den Wert `TypeProfile` enthalten, damit Spring Data Rest das Element als Typprofil erkennt. Außerdem müssen die Felder `name` sowie `subSchemaRelation` spezifiziert werden und die Liste `attributes` existieren, auch wenn keine Attribute spezifiziert werden. Attribute werden durch Referenzen auf die entsprechenden Attribut-Entitäten hinzugefügt, wobei die Reihenfolge der Attribute in der Liste die Reihenfolge der Attribute im Typprofil bestimmt. Das Konzept der Subschemarelationen wurde in Sektion 4.1 beschrieben. Die Multivererbung wird durch die Referenz auf die Eltern-Typprofile in der Liste `inheritsFrom` realisiert. Für die bessere Menschenlesbarkeit sollten zudem die Felder für die Beschreibung und erwartete Nutzungsszenarien spezifiziert werden, ansonsten wird eine Warnung ausgegeben. Mithilfe des Feldes `isEmbeddable` wird festgelegt, ob das Typprofil als Wert in FAIR-DOs und somit als Attribut in anderen Typprofilen verwendet werden kann. In Anhang A.3.4 sind Beispiele für Typprofile mit und ohne Vererbung sowie mit und ohne zusätzliche Attribute abgedruckt.

5.3.5. Erstellen von OperationTypeProfiles

Die Erstellung von FAIR-DO-OpTPs erfolgt über den `/api/operationTypeProfiles`-Endpunkt mithilfe der `POST`-Methode. Das Feld `name` ist, wie immer, verpflichtend, eine Beschreibung ist erwünscht. Ein FAIR-DO-OpTPs spezifiziert in der Liste `inheritsFrom` die Eltern-FAIR-DO-OpTP, von denen geerbt wird. Die Liste

`attributes` referenziert alle Eingabeargumente, während die Liste `outputs` alle Ausgangsattribute referenziert. Auf die FAIR-DO-OpTPAs werden in der Liste `adapters` mithilfe von PIDs verwiesen, da diese nicht in IDORIS verwaltet werden. Im Anhang A.3.5 sind zwei FAIR-DO-OpTPs für HTTP-Anfragen und Regex eingefügt.

5.3.6. Erstellen von Operationen

Erst nachdem alle nötigen BasicDataTypes, Attribute, Typeprofiles und FAIR-DO-OpTPs erstellt wurden, können FAIR-DO-Ops mit dem `/api/operations`-Endpunkt und der `POST`-Methode erstellt werden. Die Erstellung von FAIR-DO-Ops ist am komplexesten, da sie alle vorherigen Entitäten referenzieren und miteinander verknüpfen (siehe Anhang A.3.6). Eine gültige FAIR-DO-Op enthält einen Namen, einen Verweis auf den Datentypen, auf diesem die jeweilige Operation ausführbar ist, im Feld `executableOn` und mindestens einen Operationschritt in der Liste `execution`. Außerdem sollten eine Beschreibung und eine Liste von Rückgabewerten (`returns`) spezifiziert werden. In der Liste `execution` werden alle Ausführungsschritte einer FAIR-DO beschrieben.

Es kann eine beliebige Anzahl von Operationsschritten hinzugefügt werden, die vom Ausführungssystem basierend auf ihrem `executionOrderIndex` ausgeführt werden. Daher muss jeder Operationsschritt einen `executionOrderIndex` und einen Namen enthalten. Zudem muss entweder ein FAIR-DO-OpTP referenziert (Feld `operationTypeProfile`), eine FAIR-DO-Ops referenziert (Feld `operation`) oder untergeordnete Operationsschritte in der Liste `steps` enthalten sein. Davon darf gleichzeitig immer nur genau eines vorhanden sein, damit die Ausführung klar spezifiziert ist. In den Feldern `attributes` und `outputs` werden die, innerhalb des Operationskontextes verfügbaren, Attribute zu denen, die die jeweilige Ausführung benötigt, gemappt. Ein Attribut-Mapping muss immer ein Ausgabeattribut (`output`) spezifizieren, dem auf drei verschiedene Arten eine Eingabe zugeordnet werden kann:

- Nur mit einem Eingabeattribut `input`
- Nur mit einem festen Wert `value`
- Durch Einsetzen des Eingabeattributs in den festen Wert durch Ersetzen einer, in `replaceCharactersInValueWithInput` spezifizierten, Zeichenkette (standardmäßig `{{input}}`) im `value`-Feld (siehe A.38)

Falls das Eingabeattribut wiederholbar ist und das Ausgabeattribut nicht, muss das Feld `index` spezifiziert werden. Dieses Mapping ist derzeit für alle zu spezifizierende Attribute notwendig, auch, wenn Ein- und Ausgabeattribute identisch sind. Zur besseren Lesbarkeit sollte zudem ein Name für das Attribute-Mapping spezifiziert werden. In Anhang A.3.6 sind zwei Operationen abgedruckt, die eine ORCiD-Nummer aus einer ORCiD-URL extrahieren A.36, in eine URL für die ORCiD-API einfügen und damit einen Ausschnitt eines ORCiD-Profil zurückgeben können A.38.

Dieses konstruierte Beispiel demonstriert den Aufbau von FAIR-DO-Ops in IDORIS und die verbundene Komplexität durch die Vielzahl der erstellten Entitäten. Die hier demonstrierten Verfahren sind allerdings so flexibel, dass auch komplexe Arbeitsabläufe durchgehend typisiert, modularisiert und wiederverwendbar abbildbar sind. Ein Beispiel für so einen komplexen Arbeitsablauf, welches nicht weiter ausgeführt wird, könnte beispielsweise die Interaktion mit einem API-Endpunkt sein, bei dem die Authentifizierung, die Anfrage und die Verarbeitung der Antwort in verschiedenen Operationsschritten oder Teiloperationen abgebildet werden. In dieser Operation könnte auch ein neuer Dienst mithilfe eines Docker-Containers gestartet werden, notwendige Konfigurationsdateien und die zu verarbeitenden Daten parallelisiert heruntergeladen und von dem soeben gestarteten Dienst verarbeitet und ausgegeben werden. Die Anwendungsmöglichkeiten sind auch nicht auf die Verarbeitung von Daten beschränkt, sondern könnten theoretisch auch manuelle Laborarbeiten oder die Steuerung von Robotern abbilden.

5.3.7. Modifikationen von Entitäten

Nachdem alle Entitäten erstellt wurden, können diese mithilfe der API mithilfe der `PUT`- und `PATCH`-Methoden modifiziert werden. Mithilfe von `PUT` können alle Felder einer Entität geändert werden, während mit `PATCH` nur einzelne Felder geändert werden können. Jede Änderung wird von den Validatoren überprüft, um sicherzustellen, dass die Änderung korrekt ist. Mithilfe der `DELETE`-Methode können Entitäten gelöscht werden, wobei Entitäten mit anderen Beziehungen oder einem eigenen Repository nicht gelöscht werden (z. B. Attribute).

5.3.8. Verwalten von Beziehungen

Die Verwaltung von Beziehungen erfolgt parallel zu der von Entitäten. Bei jeder `GET`-Anfrage an eine Entität werden HATEOAS-Verweise zu den Beziehungen der Entität (z. B. `/api/typeProfiles/pid123456789/attributes`) zurückgegeben. Durch eine `PUT`-Request mit dem Header `Content-Type: text/uri-list` an die URL aus dem HATEOAS-Verweis kann eine neue Beziehung zu dem im Body referenzierten Element hergestellt werden [89]. Zum Bearbeiten oder Löschen eines einzelnen Verweises wird die `PUT`- oder `DELETE`-Methode auf ein einzelnes Element in der Liste der Beziehungen angewendet. Diese einzelnen Elemente können mit einem Index hinter dem HATEOAS-Link referenziert werden (z. B. `/api/typeProfiles/pid123456789/attributes/1`) [89]. Alternativ können Beziehungen auch mithilfe von `PUT` oder `PATCH`-Anfragen an die Entität selbst bearbeitet werden, indem das jeweilige Feld neu spezifiziert wird.

5.3.9. Suche nach ausführbaren Operationen

```
1  @Bean
2  public RepresentationModelProcessor<EntityModel<DataType>> dataTypeProcessor() {
3      return new RepresentationModelProcessor<EntityModel<DataType>>() {
4          @Override
5          public EntityModel<DataType> process(EntityModel<DataType> model) {
6              String pid = Objects.requireNonNull(model.getContent()).getPid();
7              model.add(Link.of(linkTo(IOperationDao.class)
8                      .slash("api")
9                      .slash("operations")
10                     .slash("search")
11                     .slash("getOperationsForDataType")
12                     .toUri() + "?pid=" + pid, "operations"));
13             return model;
14         }
15     };
16 }
```

Listing 5.11: Ausschnitt des Quellcodes der `RepositoryRestConfig` für das Hinzufügen von HATEOAS-Links

Listing 5.11 zeigt den Code für das Hinzufügen eines HATEOAS-Links für Datentypen. Die ersten zwei Zeilen definieren eine Methode, die ein `RepresentationModelProcessor` für `EntityModel<DataType>` zurückgibt. Gemäß Spring Data REST-Dokumentation [62] können diese Prozessoren verwendet werden, um Rückgaben zu manipulieren. An dieser Stelle werden nicht die enthaltenen Daten modifiziert, sondern lediglich ein

neuer Verweis zum HATEOAS-Modell hinzugefügt. Dieser Link wird mithilfe des Builder-Patterns von Spring HATEOAS [63] spezifiziert und führt zu einer Suchfunktion des `IOperationDao`. Auf Anfrage gibt dieser Endpunkt alle auf einem, mit der angegebenen PID identifizierten, Datentypen verfügbaren FAIR-DO-Ops zurück. In Anhang A.3.7 wird dies mit dem Datentypen `ORCiD-URL` demonstriert. Die auftretenden Fehler sind der Tatsache geschuldet, dass die Cypher-Query [68] in Listing A.40 die Attribute und OperationSteps noch nicht zurückgibt.

5.3.10. Auflösung der Vererbungshierarchie

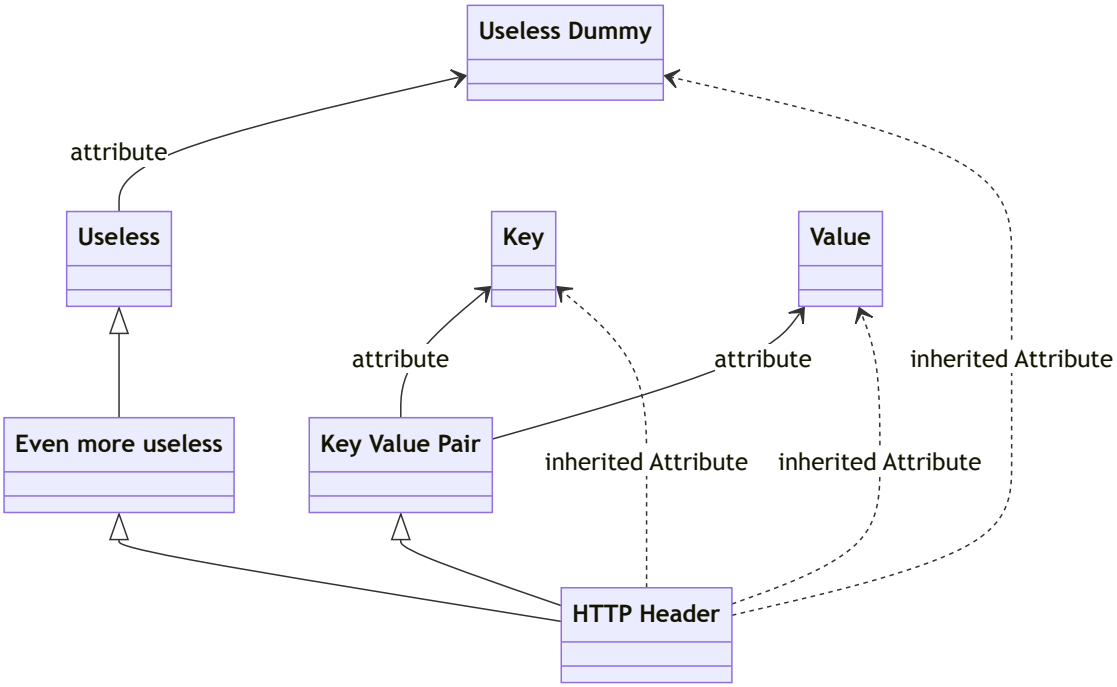

Abbildung 5.3.: Beispiel für die Mehrfachvererbung anhand von `HTTP-Header`

Im Folgenden werden einige Vererbungsfunktionalitäten anhand des Typprofils für HTTP-Header in Listing A.29 demonstriert und in Abbildung 5.3 dargestellt. Die umfassenden Vererbungsfunktionalitäten können am besten anhand von Typprofilen demonstriert werden, da diese Mehrfachvererbung umsetzen. Dieses Profil demonstriert die Mehrfachvererbung, indem es einerseits vom Typprofil `Key-Value-Pair` in Listing A.26 und andererseits von einem Dummy-Typprofil (`Even more useless`) für diese Demonstration erbt, welches wiederum von dem `Useless`-Typprofil (siehe Listing A.27) erbt. Es existieren derzeit drei verschiedene Möglichkeiten zur Auflösung der Vererbungshierarchie:

- **`/api/typeProfiles/PID-EINFÜGEN/inheritsFrom`:** Dieser Endpunkt gibt die direkten Eltern-Typprofile zurück, die bei der Erzeugung oder Modifikation des Typprofils vom Nutzer spezifiziert wurden. Siehe Ergebnis einer beispielhaften Anfrage in Listing A.43.
- **`/api/typeProfiles/PID-EINFÜGEN/inheritedAttributes`:** Dieser Endpunkt gibt eine Liste aller Attribute zurück, die von den Eltern-Typprofilen geerbt wurden. Siehe Ergebnis einer beispielhaften Anfrage in Listing A.44.
- **`/api/typeProfiles/PID-EINFÜGEN/inheritanceTree`:** Dieser Endpunkt gibt einen Vererbungsbaum mit allen Eltern-Typprofilen, deren Eltern-Typprofilen, usw. zurück. Siehe Ergebnis einer beispielhaften Anfrage in Listing A.45.

6. Evaluation

Um IDORIS als Prototyp einer neuartigen DTR evaluieren zu können, müssen auch die Änderungen und Erweiterungen am FAIR-DO-Modell betrachtet werden. Bei der Konzeptionierung der Idee für diese Arbeit war anfangs lediglich geplant FAIR-DO-Ops zu dem, in Kapitel 2 beschriebenen, existierenden FAIR-DO-Modell hinzufügen, um Änderungen am bestehenden System möglichst gering zu halten. Die ursprünglichen Gründe gegen eine komplette Neuentwicklung waren vielfältig, beispielsweise die hohe Akzeptanz der ePIC-DTR im wissenschaftlichen Diskurs, die Notwendigkeit eines stabilen und skalierbaren Betriebs, der Aufwand für eine Umstellung aufbauender Softwaresysteme sowie der hohe Aufwand für eine Neuentwicklung. Es müsste schon ein (hoher) Mehrwert geschaffen werden, um den Aufwand der Entwicklung zu rechtfertigen sowie die Nutzung der Neuentwicklung attraktiv zu gestalten. Dementsprechend wurden Ideen für neue Funktionalitäten und Verbesserungen, die einen solchen Mehrwert schaffen würden, zusammengestellt und eine Auswahl als Ziele für diese Arbeit definiert, deren Erfüllung im Folgenden evaluiert wird.

6.1. Intuitives Datenmodell mit REST-API

Das Datenmodell der ePIC-DTR ist komplex und in Teilen redundant aufgebaut, was in Teilen auf die Limitierung der zugrundeliegenden JSON-Dokumente zurückzuführen ist. Außerdem sind viele Felder nicht dokumentiert oder intuitiv genug gestaltet, dass der Sinn und die erwarteten Werte aus dem Kontext erschlossen werden könnten. Aus diesen Gründen wurde für IDORIS das Datenmodell vereinfacht (siehe Sektion 4.1) und (hoffentlich) intuitiver gestaltet. Zunächst wurden PID-InfoTypes und KIP, aufgrund des hohen Grades an schematischer Übereinstimmung, in Typprofile zusammengeführt. BasicDataTypes und Typprofile werden in der Oberklasse `DataType` verwaltet, um Doppelstrukturen im Rest des Modells (z. B. bei den Attributen) zu vermeiden sowie deren Einsatz in FAIR-DOs flexibler zu gestalten. Für BasicDataTypes könnte zukünftig evaluiert werden, ob Funktionen, wie etwa die Definition von Maßeinheiten oder die Kategorien nicht anderweitig umsetzbar sind, sodass BasicDataTypes ausschließlich durch reguläre Ausdrücke und Wertenumerationen definiert werden.

Das Ziel eine zum Datenmodell passende REST-API bereitzustellen, die optional HATEOAS unterstützt, wurde mithilfe von Spring Data REST [61] effizient umgesetzt (siehe Sektion 5.1). Spring Data REST ermöglicht es zudem einfache Cypher-Queries direkt via REST-API zugänglich zu machen. Für komplexere Anfragen wurden zusätzliche Spring Web Model-View-Controller (MVC)-Endpunkte implementiert. Der Einsatz von Spring Data REST wird sich dem Test der Zeit stellen müssen, da bereits jetzt die Grenzen der Flexibilität dieses Werkzeugs aufgezeigt werden. Diese Grenzen wurden besonders bei der Umsetzung der Validatoren recht früh sichtbar. Ein Wechsel auf eine klassische Model-Service-Controller Architektur mit Spring MVC wurde erwägt, aber wegen des hohen Aufwands zur Replizierung aller Funktionalitäten, die mit Spring Data REST sehr einfach umgesetzt werden können, verworfen. Ein weiterer Grund gegen einen Wechsel war der HAL-Explorer, mit welchem die API in einem UI einfach erkundet werden kann und dafür ALPS-Profile benötigt, welche Spring Data REST automatisch generiert.

6.2. Spezifikation von FAIR-DO-Ops

Die ePIC-DTR wurde entwickelt, um die Maschineninterpretierbarkeit für FAIR-DOs durch ein globales Typisierungssystem zu ermöglichen. Wie in Abschnitt 2.3 erläutert wird, ist die automatische Verarbeitung von maschinenprozessierbaren FAIR-DOs der nächste Evolutionsschritt. Im wissenschaftlichen Diskurs werden zum Veröffentlichungszeitpunkt dieser Arbeit verschiedene Ansätze für diese Maschinenprozessierbarkeit entwickelt. Am weitesten entwickelt war zu diesem Punkt die Spezifikation und Bereitstellung von Service-Endpunkten mittels DOIP [34, 35], einem Protokoll für den Umgang mit FAIR-DOs. Dieser Ansatz ist entfernt mit der Dokumentation von HTTP-API mithilfe von OpenAPI [90] vergleichbar und sehr stark an den jeweiligen Dienst gebunden, da dieser den Endpunkt bereitstellt. FAIR-DO-Ops können jedoch auch als Mittel für eine technologieoffene Interaktion mit FAIR-DOs verstanden werden, wie auch FAIR-DOs als einheitliche Abstraktionsebene für (Meta-)Daten verstanden werden können. Eine lose Betrachtung dieser Technologie-agnostischen FAIR-DO-Ops ist aufgrund der fehlenden maschineninterpretierbaren Definition der Ein- und Ausgaben nicht sinnvoll. Erst durch die Verknüpfung mit dem bestehenden Typisierungssystem der FAIR-DOs können automatische Entscheidungen über die Interaktionsmöglichkeiten mit FAIR-DOs getroffen werden. In Abgrenzung zu den DOIP-basierten FAIR-DO-Ops werden das in dieser Arbeit entwickelte Operations-Konzept durch ihre starke Bindung an Datentypen als typgebundene FAIR-DO-Ops bezeichnet. Servicegebundene FAIR-DO-Ops (siehe 2.3) sind in der Anzahl von Operationen sowie in ihrer Flexibilität den typgebundenen Operationen mit deren maschineninterpretierbaren Ausführung unterlegen.

IDORIS ist in der Lage typgebundene FAIR-DO-Ops zu verwalten und mit den Datentypen zu verknüpfen sowie diese Verknüpfung bidirektional abzufragen. In Abschnitt 4.3 wurde ein technologieoffenes Konzept für typgebundene FAIR-DO-Ops entwickelt. Die Ziele technologieoffene FAIR-DO-Ops abzubilden, mit Datentypen zu verknüpfen und diese Verknüpfung in beide Richtungen abfragen zu können sind somit erfüllt. Zusätzlich wurde die Wiederverwendbarkeit von Technologien durch die Aufteilung in FAIR-DO-Ops, FAIR-DO-OpTPs und FAIR-DO-OpTPAs ermöglicht. Mithilfe einer umfassenden Attributlogik und den OperationSteps können außerdem komplexe Operationen mit mehreren Teilschritten (aka. Workflows) abgebildet und maschineninterpretierbar nachvollzogen werden (siehe Sektion 6.6). Diese zusätzlichen Funktionen verbessern durch die Trennung von Technologien (FAIR-DO-OpTPs), deren Ausführung (FAIR-DO-OpTPAs) und deren Verwendung (FAIR-DO-Ops) die Wiederverwendbarkeit, Erweiterbarkeit, Nachvollziehbarkeit und Nutzbarkeit sowie Entwicklung von neuen FAIR-DO-Ops. Besonders die Attributlogik mit den Attributmappings und Overrides ist ein Beispiel für die hohe, aus diesen Funktionen resultierende, (nicht nur konzeptionelle) Komplexität. Dies ist am Anwendungsbeispiel (Sektion 5.3 und Anhang A.3) sowie dem Graph (siehe 6.2) gut nachvollziehbar, wo für zwei, zugegebenermaßen verhältnismäßig einfache, Operationen knapp 13 Datentypen, 22 Attribute und 12 Attributmappings erzeugt wurden. Es muss daher zukünftig evaluiert werden, inwiefern die Komplexität bei gleichem Funktionsumfang reduziert werden kann, ohne spätere Erweiterungen zu verhindern.

6.3. Vererbung und Polymorphie

Vererbung ist in der ePIC-DTR nicht möglich, wobei allerdings ein nicht maschinenlesbarer Mechanismus zum Verweis auf Standards existiert und von manchen als semantische Vererbung bezeichnet wird. Ein funktionierender Vererbungsmechanismus zeichnet sich durch mehr als eine (maschinenlesbare) Verknüpfung zwischen Eltern- und Kind Element aus (siehe Sektion 2.4.4). Polymorphie, Override-Logik, implizite Werte und die Erweiterung oder Spezifizierung der vererbenden Datentypen sind nur eine Auswahl von Vererbungsmechanismen. Die bisherige DTR kann dies weder konzeptionell abbilden, im Datenmodell repräsentieren noch durch Vererbungslogik ergänzen.

Integrated Data Type and Operations Registry with Inheritance System (IDORIS) enthält das Wort Vererbungssystem bereits im Namen. Es kann Vererbungsverknüpfungen maschinenlesbar und -navigierbar abbilden, Vererbungshierarchien auflösen, Attribut-Overrides abspeichern und unterstützt (konzeptionell) Polymorphie. In den Zielen ist gefordert, dass Vererbung im Datenmodell abgebildet wird, die Vererbungshierarchie abgefragt werden kann und eine Liste aller geerbten Attribute für ein Typprofil ausgegeben werden kann. Dies ist erfüllt, wie in den Kapiteln 4 und 5 gezeigt wird. Es können zudem alle Operationen innerhalb der Vererbungshierarchie eines Datentyps ausgegeben werden, was ein optionales Ziel erfüllt. Dieser polymorphe Ansatz wurde etwas ausgeweitet, sodass alle Operationen für ein Typprofil selbst, aber auch für jedes (evtl. geerbte) Attribut ausgegeben werden können. Vererbungsmechanismen verursachen enormen Validierungsbedarf und können gerade bei rekursiven Algorithmen Terminierungsprobleme verursachen, weshalb für eine robuste produktionsreife DTR mit hochqualitativen Daten an dieser Stelle zukünftig mehr Arbeit investiert werden sollte.

6.4. Validierung von Einträgen und Beziehungen

Die ePIC-DTR unterstützt die syntaktische Validierung der Einträge mit JSON-Schemas. Interne Beziehungen innerhalb der DTR (z. B. von einem KIP zu den Datentypen der Felder) werden auf den korrekten Typen validiert. Für IDORIS wurde aufgrund der tiefen Strukturen ein rekursiver Ansatz mit dem Visitor-Pattern (siehe Sektion 5.2) gewählt. Mit diesem Verhaltensmuster werden, in Anlehnung zur semantischen Analyse im Compilerbau, die syntaktische Validierung einzelner Entitäten und die Validierung der Beziehungen in einer rekursiven Tiefensuche durchgeführt. Gleichzeitig werden mögliche Fehler und Warnungen mitsamt Nachricht und dem jeweiligen betroffenen Element für den Nutzer in einer Baumstruktur gesammelt, sodass Fehler einfach zu finden sind. Dies erfüllt ein weiteres optionales Ziel. Durch die Einführung der Vererbungsprinzipien und der hohen Anzahl von Verknüpfungen entstehen Zykluse im Graphen, die ohne Behandlung zu einer Endlosschleife der rekursiv arbeitenden Algorithmen führen können. Aus diesem Grund wurde ein primitives Abbruchkriterium in Form eines erweiterten Cache erstellt, welches bei bestimmten Validatoren unpräzise ist. Um zukünftig eine effiziente parallele Verarbeitung der Ergebnisse ermöglichen zu können, sollten daher andere Strategien wie etwa der Einsatz von Symboltabellen oder die Verwendung von Graphfunktionen evaluiert werden. Außerdem muss besonders die Vererbungshierarchie noch stärker hinsichtlich rekursiver Abhängigkeiten evaluiert werden.

6.5. Erweiterbarkeit

IDORIS ist durch die Modularisierung von Spring Boot einfach erweiterbar. Das Visitor-Pattern ermöglicht eine einfache Ergänzung und Optimierung von Validatoren sowie eine generische Lösung für Operationen auf den Entitäten. Diese Erweiterungen können zur bisherigen Validierungslogik hinzugefügt werden, oder über neue Web MVC-Endpunkte direkt angesprochen werden. Neue Queries (durch Ableitung aus dem Methodennamen oder mithilfe von Cypher) können einfach in dem betreffenden Repository hinzugefügt und über dieses direkt HATEOAS-kompatibel über die REST-API zugänglich gemacht werden. Alternativ können die Repository-Methoden auch in den Web MVC-Endpunkten verarbeitet und anschließend ausgegeben werden. Für schwierig zu typisierende Query-Ergebnisse kann der Neo4jClient [91] aus dem Spring Data Neo4j-Framework beispielsweise in Web MVC-Endpunkten oder Visitoren verwendet werden, muss aber eigenständig konfiguriert werden.

6.6. Verwendung von Neo4j

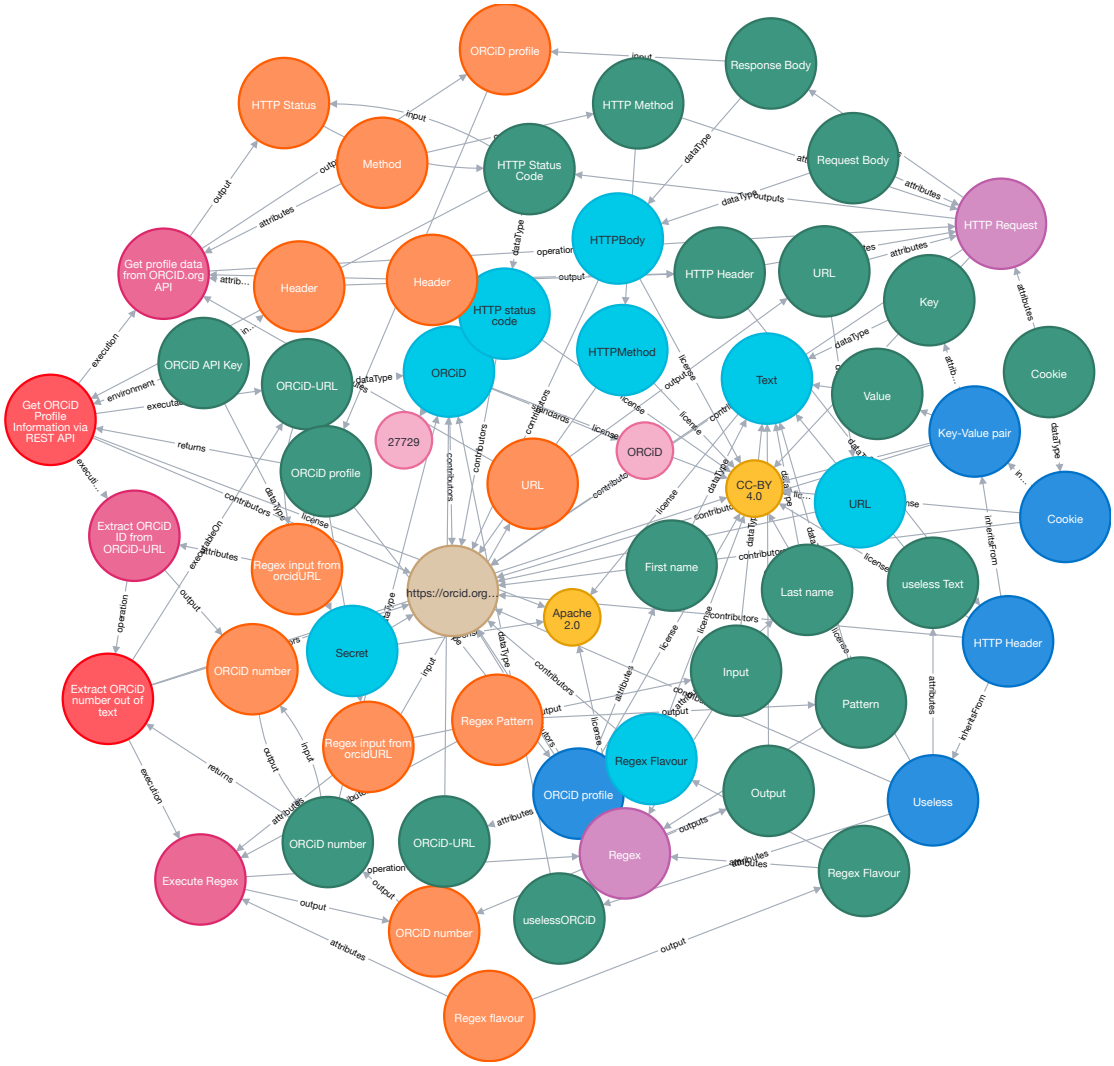

Abbildung 6.1.: Visualisierung des Graphen aus Sektion 5.3 auf der Neo4j-Weboberfläche
Query: `MATCH (n) RETURN n`

In Abschnitt 4.5 wird die Wahl Neo4j als Graphdatenbank für die Speicherung der Daten mit dem stark verknüpften Datenmodell von IDORIS begründet. Abbildung 6.1 ist ein Graph (generiert von der Neo4j-Weboberfläche) mit allen Daten aus dem Anwendungsbeispiel in Sektion 5.3, welche in 59 Knoten und 123 Kanten abgebildet werden. Die Farben der Knoten entsprechen dem jeweiligen Label in Neo4j bzw. der jeweiligen Modellklasse in IDORIS und in der Legende (siehe Abbildung 6.2) definiert. Zur einfachen Darstellung der Zusammenhänge zwischen den einzelnen Labels wurde ein Metagraph erstellt, welcher in Abbildung 6.3 dargestellt ist. Ein Metagraph ist jedoch nicht mit den Datenbankschemas in relationalen Datenbanken vergleichbar, da es zwar eine abstrakte Darstellung der Datenbankinhalte ist, aber in Neo4j keine Spezifikation des Datenbankschemas stattfindet.

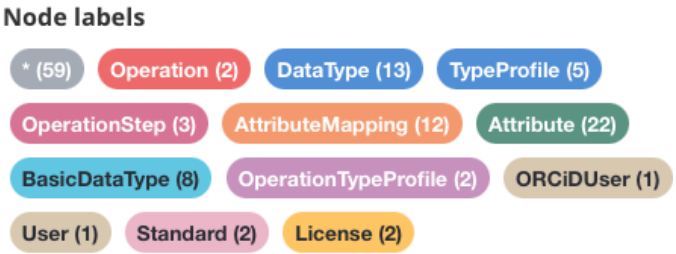

Abbildung 6.2.: Legende für die Farben der Knoten

Dieser Metagraph enthält im Gegensatz zum Modellklassendiagramm (siehe 4.3) nur die Labels und Relationen, die in der Datenbank vorhanden sind, und zeigt in dieser Konfiguration zur besseren Lesbarkeit nicht die Klassen `DataType`, `TextUser` und `ORCiDUser` an.

Abbildung 6.3.: Metagraph für das IDORIS-Modell in Neo4j
Query: `CALL apoc.meta.subGraph({includeLabels:['Attribute', 'AttributeMapping', 'BasicDataType', 'TypeProfile', 'Operation', 'OperationTypeProfile', 'OperationStep', 'User','License']})`

Abbildung 6.4 zeigt den Aufbau der FAIR-DO-Op aus dem Anwendungsbeispiel (siehe 5.3) mit allen Attributen, Attribut-Mappings, OperationSteps und FAIR-DO-OpTPs. Dieser Graph soll die Abbildung von FAIR-DO-Ops als zentralen Bestandteil dieser Arbeit dokumentieren, weshalb aus Gründen der besseren Lesbarkeit einige Knoten nicht abgebildet werden, auch wenn diese für FAIR-DO-Ops von großer Bedeutung sind. Einstiegspunkt für diese Sektion ist die Operation `Get ORCiD Profile Information via REST API` (rot markiert, links oben). Es sind gleich mehrere interessante Aspekte im Graphen zu erkennen. Zum einen wird auf der Horizontalen die Unabhängigkeit der FAIR-DO-OpTPs von den FAIR-DO-Ops sichtbar, da von den `OperationTypeProfiles` keine ausgehenden Pfade zu den Operationen existieren können (siehe auch Modellklassendiagramm in Abbildung 4.3). Die Attribute der FAIR-DO-OpTPs (rechts in dunkelgrün) werden ausschließlich mithilfe der AttributeMappings (mittig in orange) von den OperationSteps verarbeitet (links in pink). Außerdem wird die Abhängigkeit des oberen OperationSteps (`Get profile data from ORCiD.org API`) vom Ausgabeattribut `ORCiD number` des vorhergehenden OperationSteps, welches vom AttributeMapping `URL` des nachfolgenden OperationSteps benötigt wird, deutlich. Dadurch ist ersichtlich, dass diese OperationSteps zwangsläufig nacheinander ausgeführt werden müssen. Ansonsten bestehen zwischen den beiden OperationSteps keine direkten Verbindungen und somit keine weiteren Abhängigkeiten.

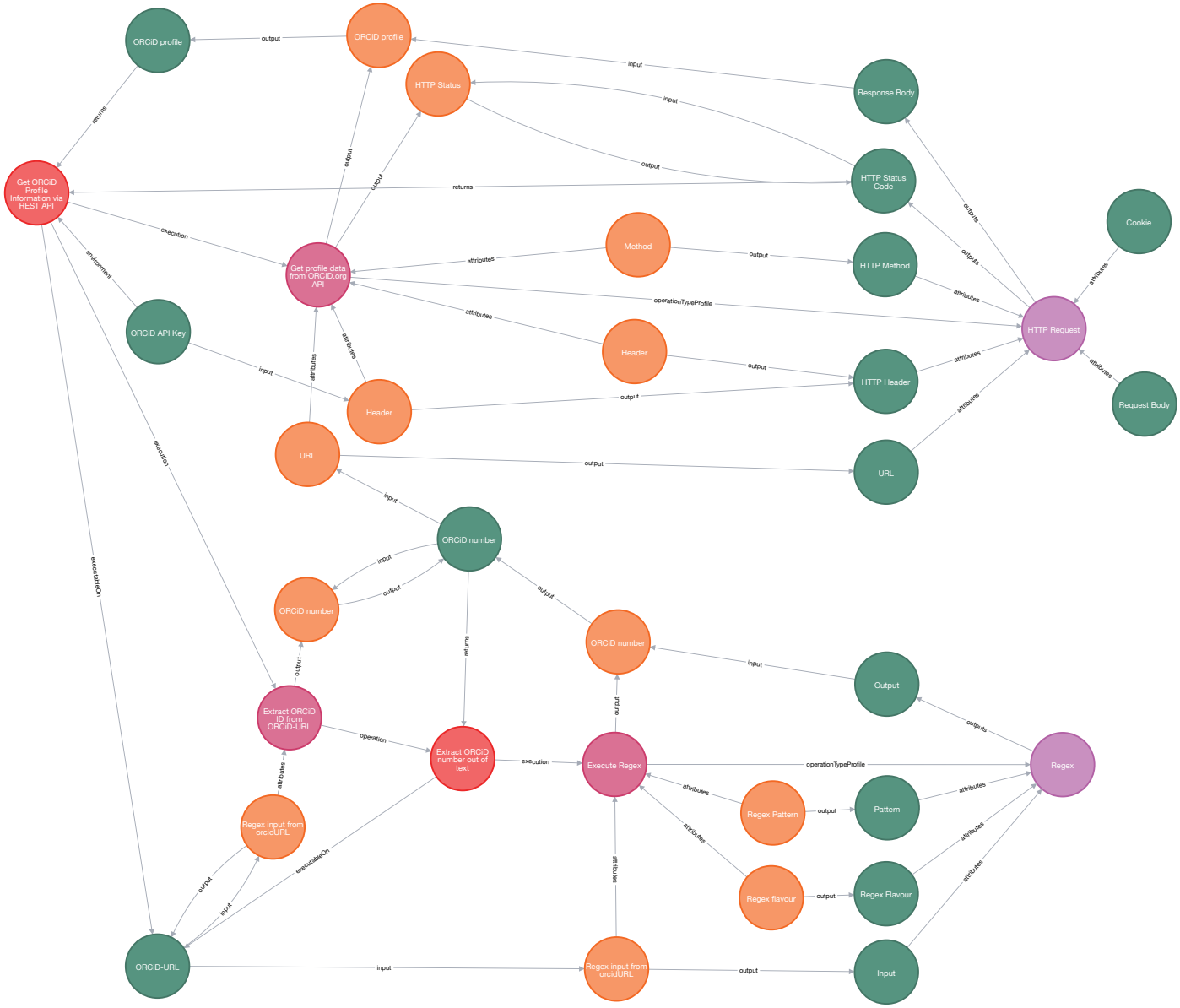

Abbildung 6.4.: Aufbau einer Operation im Graph
Query: `MATCH (n: Operation | AttributeMapping | OperationStep | OperationTypeProfile | Attribute) RETURN n)`

Abbildung 6.5 fügt zu dem Graphen aus Abbildung 6.4 Annotationen für mögliche Abstraktionsniveaus hinzu. Diese Annotationen sollen die Betrachtung von FAIR-DO-Ops mit verschiedenen Detailtiefen visuell darstellen und die Fähigkeiten der visuellen Repräsentation sowie Abfrage von Informationen in Graphen verdeutlichen. Die disjunkte Darstellung der “Abstraktionsniveaus” dient ausschließlich der besseren Lesbarkeit, da der Informationsgehalt durch Schnittmengen (besonders bei der Referenz von Attributen) massiv erhöht werden kann.

Zum einen könnten nur die FAIR-DO-Ops mit ihren jeweiligen Ein- und Ausgabeattributen (siehe lila Rechtecke) betrachtet werden. Diese Darstellung eignet sich zur schnellen Einordnung der Nützlichkeit einer FAIR-DO-Op ohne notwendige Kenntnisse über den internen Ablauf, beispielsweise bei der automatischen Suche von möglichen Pfaden zwischen zwei bestimmten DataTypes, da jedem Attribut definitionsgemäß ein Datentyp zugeordnet ist.

Eine andere Darstellung (grüne Rechtecke) enthält die einzelnen OperationSteps sowie die AttributeMappings, welche die entsprechenden Attribute referenzieren, und kann beispielsweise zur Bestimmung der Ausführungsreihenfolge basierend auf den Attributabhängigkeiten genutzt werden. Ein OperationStep kann erst ausgeführt werden, wenn alle notwendigen Attribute spezifiziert sind. Dies kann durch Rückverfolgung der AttributeMappings überprüft werden. Bei genauer Betrachtung fällt auf, dass einige AttributeMappings in keinem Rechteck enthalten sind. Das liegt daran, dass diese keine Eingangsatttribute besitzen und die Attributwerte stattdessen statisch spezifizieren, sodass diese für die Abstraktionsebenen de facto unsichtbar sind.

Die detaillierteste Ebene enthält die Attribute und FAIR-DO-OpTPs (siehe blaue Rechtecke). Diese Details werden für die konkrete Ausführung einer Operation mit den, vorab durch die AttributeMappings eingesetzten Werten verwendet. FAIR-DO-OpTPs sowie ihre Attribute sind streng genommen nicht mehr Teil der Operationsspezifikation, sondern repräsentieren wiederverwendbar und erweiterbar jeweils eine Technologie, die von potenziell unbegrenzt vielen FAIR-DO-Ops genutzt werden kann.

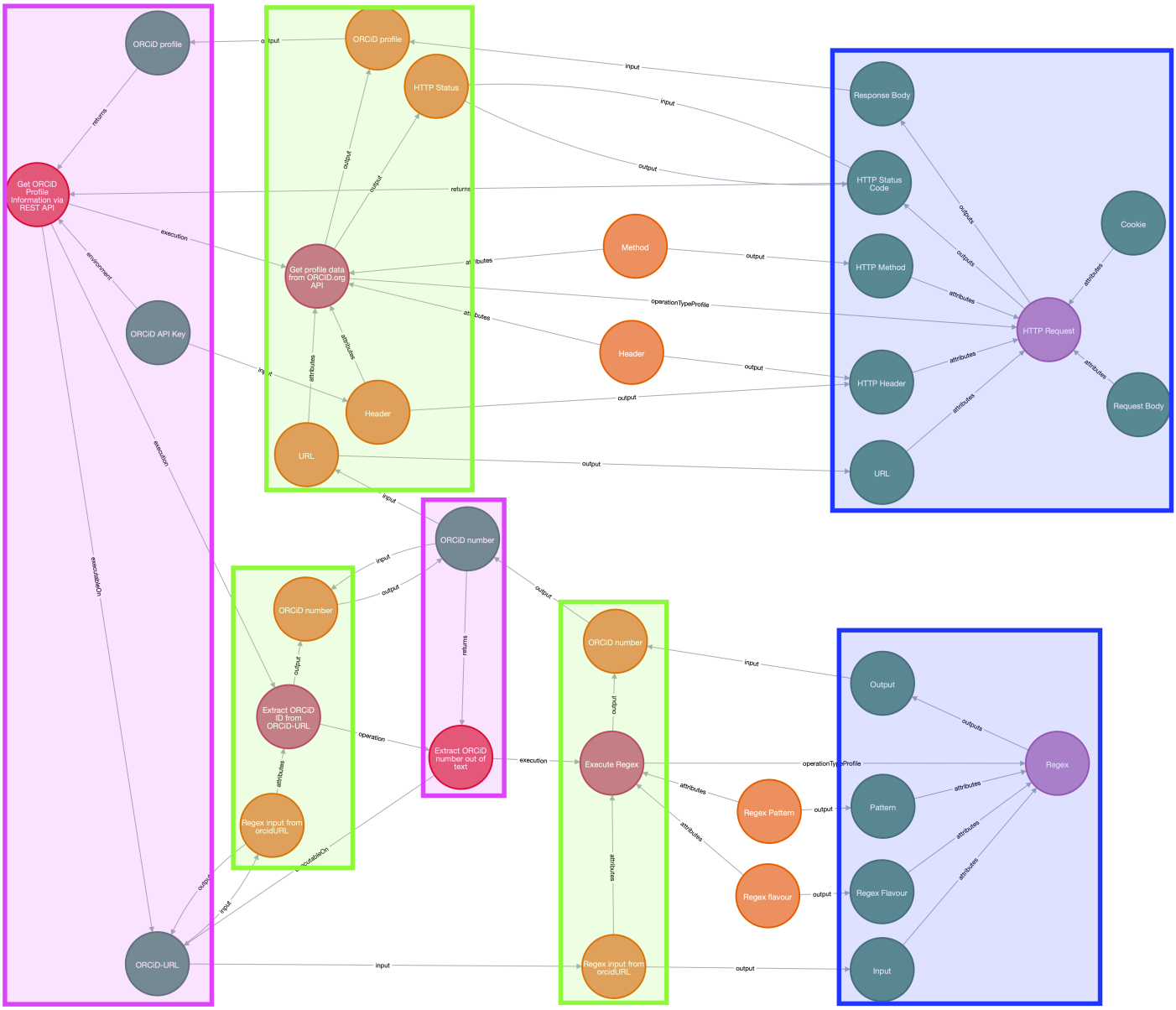

Abbildung 6.5.: Mögliche Abstraktionsniveaus in einer Operation

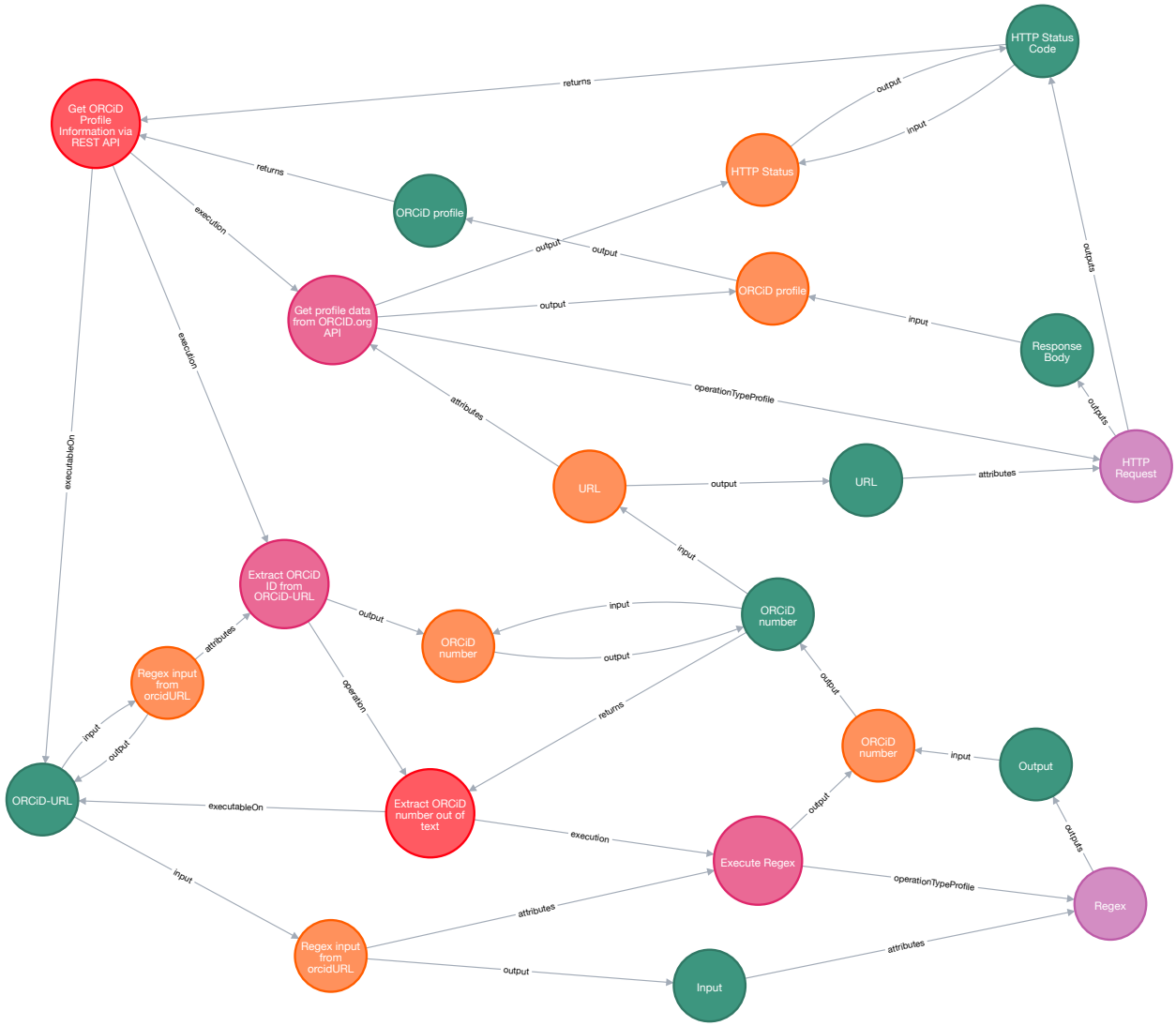

Abbildung 6.6.: Zirkuläre Abhängigkeiten im Graph
Query: `MATCH (m1) WITH collect(m1) as nodes CALL apoc.nodes.cycles(nodes) YIELD path RETURN path)`

Speicherung aller anfallenden Information in der Neo4j-Graphdatenbank [67] bringt neben effizienten Abfragen von Beziehungen, der anschaulichen Visualisierung und dem Finden von Pfaden zwischen Datentypen noch weitere Möglichkeiten mit sich. Es können zirkuläre Zusammenhänge, wie in Abbildung 6.6 gezeigt wird, einfach erkannt werden. Aus diesem Grund zeigen Graph 6.6 und 6.7 nur in den Zyklen vorhandene Elemente an, sodass beispielsweise einige AttributeMappings ohne Eingangsattribut nicht angezeigt werden. Für bestimmte Relationen, wie beispielsweise der Vererbung, der Assoziation verpflichtender Attribute innerhalb eines TypeProfiles, und die Validierung sind zirkuläre Abhängigkeiten zu vermeiden. Diese können, anstelle der Query von Abbildung 6.6, auch mit einer Tiefensuche unter Zuhilfenahme einer Symboltabelle in der rekursiven Validator-Architektur erkannt werden. In Neo4j sind diese Abfragen jedoch um einiges effizienter umgesetzt, sodass hier keine Doppelentwicklung betrieben werden muss.

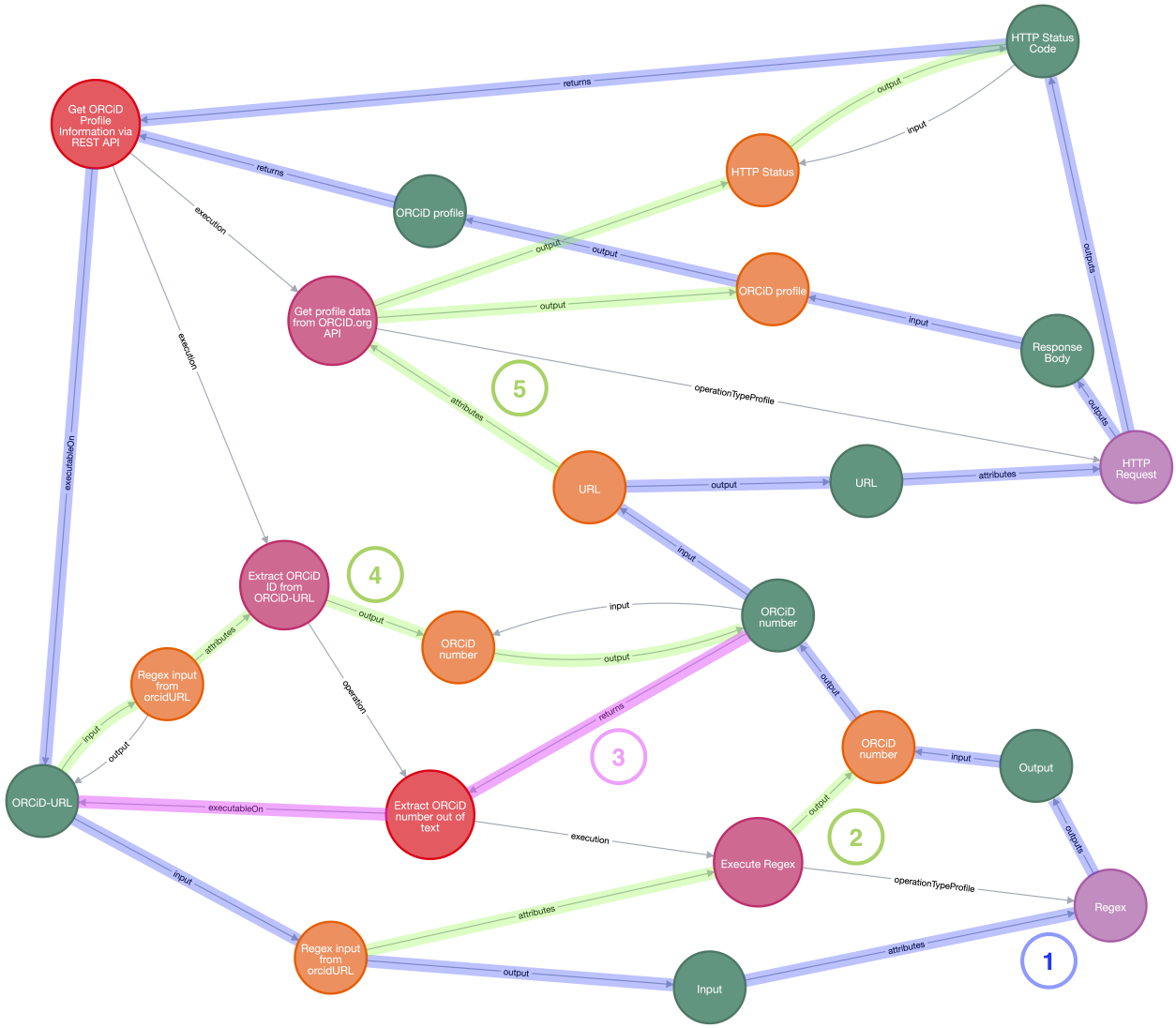

Abbildung 6.7.: Nachverfolgung des Attributdatenflusses in einer Operation

Der aus dem Anwendungsbeispiel resultierende Graph zeigt einige zirkuläre Abhängigkeiten auf, was in Abbildung 6.6 visualisiert wird. Vorab wurde auf zirkuläre Abhängigkeiten sowie deren Folgen für rekursive Validierungsalgorithmen hingewiesen. Mit einer angemessenen Erkennungslogik sind die in Abbildung 6.6 sichtbaren Zyklen unproblematisch und tatsächlich nützlich, wie im Folgenden anhand von Abbildung 6.7 erläutert wird. Zur besseren Orientierung sind ausgewählte Zyklen in Abbildung 6.7 markiert und durchnummeriert. Die Farben der Markierungen entsprechen den Farben der Abstraktionsniveaus in Abbildung 6.5 und sind als Einordnung in den Gesamtkontext der FAIR-DO-Op zu verstehen. Im Folgenden werden die einzelnen Zyklen anhand ihrer Nummer kurz erläutert:

1. Dieser Zyklus zeigt den gesamten Datenfluss der Operation `Get ORCiD Profile via REST-API` von der Eingabe der `ORCiD-URL` bis zur Ausgabe des `ORCiD-Profils` und `HTTP Status Code`. Daher kann auch von einer Nachverfolgung des Attributdatenflusses innerhalb einer FAIR-DO-Op gesprochen werden. Alle Attribute, die nicht mit einem statischen AttributeMapping gemappt werden, sind in diesem Zyklus enthalten. Beispiele für statische AttributeMappings (siehe 4.3.3), die den Wert für ein Attribut statisch festlegen, sind `Regex-Pattern` und `Regex flavour` aus Abbildung 6.4. Alle folgenden Zyklen zweigen von diesem Datenfluss ab und münden in diesen zurück.

2. An dieser Stelle kann der Datenfluss 1 mithilfe des Abstraktionsniveaus für OperationSteps (siehe Abbildung 6.5) abgekürzt werden. Dies ermöglicht eine Übersicht über die Aufgabe des OperationSteps, ohne das FAIR-DO-OpTP und seine Attribute näher betrachten zu müssen. Wenn in der Operation `Extract ORCiD number out of Text` mehrere OperationSteps vorhanden wären, würden auf diesem Abstraktionsniveau die Abhängigkeiten zwischen den Attributen der einzelnen OperationSteps evaluiert, um Deadlocks zu vermeiden.

3. Diese "Abkürzung" im Datenfluss abstrahiert die OperationSteps und zeigt nur die Ein- und Ausgabeattribute der Operation. Diese Abstraktion wurde bereits in Abbildung 6.5 erläutert und dient zur schnellen Einordnung der Nützlichkeit einer FAIR-DO-Op.

4, 5. Zyklen 4 und 5 verbergen die Details der einzelnen OperationSteps und zeigen die, in Punkt 2 angesprochene, Abhängigkeit der beiden OperationSteps zueinander. Diese Abhängigkeit besteht durch das Ausgabeattribut `ORCiD number` als Ergebnis des unteren OperationSteps, welches als Eingabeattribut für den oberen OperationStep dient.

6. Es existiert kein Zyklus für die oberste Abstraktionsebene, da diese lediglich die FAIR-DO-Op und ihre Ein- und Ausgabeattribute enthält und keinen Zyklus bilden kann.

Diese Sektion zeigt die vielfältigen Möglichkeiten, die sich durch die Verwendung einer Graphdatenbank ergeben. Es können komplexe Inhalte anschaulich dargestellt und ebenso anschaulich abgefragt werden. Zudem können typische Graph-Operationen, wie das Finden von Pfaden, das Erkennen von Zyklen oder das Traversieren des Graphen, effizient umgesetzt werden. Neo4j ist zudem eine native Graphdatenbank und kann daher die Beziehungen zwischen den Entitäten effizienter speichern und abfragen als relationale Datenbanken. Der Graph konnte durch das Labeled-Property-Graph-Modell von Neo4j bewusst möglichst einfach und nah am Datenmodell abgebildet werden. Dies vereinfacht die Benutzung und Weiterentwicklung von IDORIS, da kaum künstliche Umwege, wie etwa Join-Tabellen, berücksichtigt werden müssen. Ein Nachteil von Neo4j und Graphdatenbanken allgemein ist die schwierige Skalierbarkeit. Trotzdem ist der Einsatz von Neo4j für IDORIS aufgrund der hohen Anzahl von Beziehungen und der komplexen Struktur des Datenmodells gerechtfertigt.

7. Fazit

In dieser Arbeit wurde das FAIR-DO-Datenmodell weiterentwickelt und prototypisch in einer neuartigen DTR, namens IDORIS, umgesetzt. Aufgrund der neuen Möglichkeiten durch die Verwendung einer Graphdatenbank wurde das Datenmodell aus der ePIC-DTR [6] angepasst und etwas vereinfacht. Die Idee von Technologie-agnostischen typgebundenen FAIR-DO-Ops für maschinenprozessierbare FAIR-DOs wurde weiterentwickelt, in dieses Datenmodell integriert und von IDORIS verwaltet. Außerdem wurde ein Konzept für Vererbung sowie Polymorphie entwickelt und in IDORIS implementiert. IDORIS zeichnet sich im Vergleich zu der bestehenden ePIC-DTR [6] durch das graphbasierte Datenmodell mit FAIR-DO-Ops und Vererbung sowie die umfangreiche und erweiterbare Logik für die Validierung und Abfrage von sowohl Elementen als auch Beziehungen aus. Zur Validierung wurde eine regelbasierte Logik auf Basis des Visitor-Patterns implementiert, welche einfach um neue Regeln für syntaktische und/oder semantische Validierung erweitert werden kann. Die Verwaltung aller Elemente und Beziehungen erfolgt über eine HATEOAS-kompatible REST-API, welche bei Änderungen eine Validierung initiiert und nachvollziehbare Fehlermeldungen ausgibt. Für die Implementierung von IDORIS wurden Spring Boot [59] und Spring Data Neo4j [64] gewählt, da sie sowohl eine effiziente Umsetzung von komplexen Konzepten als auch eine einfache Integration von Neo4j [67] ermöglichen und eine REST-API mit wenig Aufwand bereitstellen.

Das Ziel dieser Arbeit war die Entwicklung eines Prototyps für eine DTR mit Vererbungsmechanismen, welche FAIR-DO-Ops abbilden und mit Datentypen assoziieren kann. Dieses Ziel wurde durch die Entwicklung von IDORIS erfüllt und durch die Entwicklung eines regelbasierten Validierungsmechanismus sowie die Bereitstellung einer HATEOAS-kompatiblen API robust gestaltet (siehe Kapitel 6). Im Laufe dieser Arbeit sind einige Herausforderungen aufgetreten. Die Spezifikation von FAIR-DO-Ops musste von Grund auf neu entwickelt werden, da es keine etablierten Konzepte oder Standards für diese gibt. Dabei musste besonders darauf geachtet werden, dass diese Konzepte möglichst generisch und technologieunabhängig sind, um zusammen mit FAIR-DOs als einheitliche Abstraktionsschicht für maschinenprozessierbare Daten dienen zu können. In Anlehnung an die Typisierung von FAIR-DOs wurde dazu eine Trennung von Operationen, der Technologie und der Ausführung vorgenommen, was außerdem die Wiederverwendbarkeit und Erweiterbarkeit von FAIR-DO-Ops ermöglicht. Bei der Realisierung der verschiedenen Vererbungsmechanismen wurde die Bedeutung eines regelbasierten Validierungssystems deutlich. Eine weitere Herausforderung waren die Limitationen von Spring Data REST [61] bei der Integration dieses Validierungssystems, da das Abfangen einzelner Abfragen nicht ohne weiteres möglich ist und die eingebaute Validierungsfunktionalität limitiert ist.

Das weiterentwickelte FAIR-DO-Konzept und dazugehörige Datenmodell mit FAIR-DO-Ops und Vererbung sowie IDORIS als Prototyp für eine DTR schaffen bereits jetzt einen Mehrwert im Vergleich zu bestehenden Systemen (siehe Kapitel 6) und werden zukünftig weiterentwickelt. IDORIS schafft eine Umgebung zur weiteren Forschung an automatisiert maschinenprozessierbaren Daten und den nötigen Infrastrukturen. Es ist nun möglich Arbeitsabläufe mit FAIR-DO-Ops maschinenprozessierbar, nachvollziehbar und typisiert abzubilden, was die Reproduzierbarkeit dieser Abläufe verbessert. Das Konzept typgebundener FAIR-DO-Ops ist bewusst abstrakt und minimalistisch gehalten, um optimalen Raum für zukünftige Entwicklungen bieten zu können. Daher bieten die innovativen Entwicklungen dieser Arbeit eine Grundlage für die weitere wissenschaftliche Forschung an maschinenprozessierbaren FAIR-DOs für nachhaltige Forschungsdateninfrastrukturen.

8. Ausblick

IDORIS ist bislang ein experimenteller Prototyp für eine neuartige DTR und bietet eine solide Grundlage für zukünftige Forschung und Entwicklung. Es existiert eine Vielzahl von Ideen für die zukünftige wissenschaftliche Weiterentwicklung von IDORIS und den zugrundeliegenden Konzepten, welche im Folgenden näher erläutert werden.

8.1. Mögliche Optimierungen für den Produktiveinsatz

Ziel dieser Arbeit war kein schlüsselfertiger Dienst, der direkt verwendet werden kann. Stattdessen wurden neue Konzepte und Ideen getestet, evaluiert und schließlich prototypisch implementiert. Wie aus der Evaluation in Kapitel 6 hervorgeht, ist dies gelungen. Aus der Evaluierung gehen bereits einige Ideen hervor, wie IDORIS in einen produktionsreifen Dienst überführt werden kann.

Das Datenmodell sollte im wissenschaftlichen Diskurs weiter optimiert werden, damit es langfristig möglichst konsistent, zukunftssicher und intuitiv ist. Durch die Verwendung von Spring Data REST gewinnt ein konsistentes Datenmodells an Bedeutung, da Änderungen unmittelbar auf die API-Schnittstellen abgebildet werden. Konkret bietet sich daher eine wissenschaftliche Evaluation des, mit IDORIS eingeführten, Datenmodells an. Bei dieser Evaluation könnten vollständigere Anwendungsszenarien mit bereits existierenden FAIR-DOs betrachtet werden. Außerdem wäre eine Bedarfsermittlung hinsichtlich der gewünschten Funktionalitäten des Datenmodells und von IDORIS auch im Vergleich zur ePIC-DTR für die zukünftige Ausrichtung hilfreich. Es erscheint zudem sinnvoll die Minimalität des Datenmodells zu verifizieren, damit dieses für Nutzende und Entwickelnde intuitiv verständlich und nicht zu komplex in der Anwendung ist. Ein Beispiel für so eine Komplexitätsreduktion wäre das Entfernen der Messeinheitsspezifikationen aus den BasicDataTypes, da dies auch über Typprofile realisiert werden kann, was möglicherweise Vorteile (z. B. Konvertierung zwischen Einheiten) bietet. Ebenso könnte in der Attributlogik das Format der Obligationen, Wiederholbarkeit und bei Typprofilen die Subschema-Relation hinsichtlich der Flexibilität evaluiert werden. Für eine robustere Vererbung können in IDORIS weitere Validatoren einfach hinzugefügt werden und mehr Fehlerfälle erkennen. Außerdem könnten mithilfe von Spring Data REST Events bei Löschungen zusätzliche Maßnahmen zur Optimierung möglicherweise verbliebener Datenbankentitäten implementiert werden.

Zusätzlich zu diesen konzeptuellen und funktionalen Verbesserungsmaßnahmen kann die Software-Qualität von IDORIS selbst weiter verbessert werden. Gerade in Hinblick auf das komplexe Datenmodell, die Validatorstruktur und die Vielzahl von konzeptuellen Regeln würde eine umfangreiche Testabdeckung eine einfache automatisierte Überprüfung der genannten Mechanismen ermöglichen. Durch eine Integration von IDORIS in ein Continuous-Integration-System, um diese automatisierten Tests durchzuführen und Artefakte (z. B. Docker-Images) bereitzustellen, kann die Entwicklung vereinfachen. Für die Nutzenden sollte eine umfassende technische Dokumentation von IDORIS, der konzeptuellen Ideen und der REST-API in einem modernen und verständlichen Format bereitgestellt werden. Hierfür eignen sich zur Code-Dokumentation Javadoc [92] und zur API-Dokumentation Swagger [93]. Die Installation und Handhabung von IDORIS könnte in einem Benutzerhandbuch in Form einer durchsuchbaren Webseite beschrieben werden. Außerdem könnten für eine

einfache Installation und Konfiguration von IDORIS mit der Neo4j-Datenbank Docker-Compose-Dateien und Helm-Charts für Nutzende bereitgestellt werden.

8.2. Entwicklung einer Ausführungskomponente

IDORIS selbst ist nicht dazu gedacht FAIR-DO-Ops auszuführen, sondern nur die Definitionen und Assoziationen von FAIR-DO-Ops und Datentypen zu verwalten. Die Ausführung der FAIR-DO-Ops auf Basis von FAIR-DOs oder einzelnen Werten ist rechenintensiv und erfordert eine Umgebung, die die sichere Ausführung von möglicherweise unbekanntem Code ermöglicht. Daher wurde bereits bei der Konzeptionierung dieser Arbeit bewusst auf die Entwicklung einer solchen Ausführungskomponente innerhalb von IDORIS verzichtet und stattdessen eine Idee für den Aufbau einer solchen Komponente und ihre Interaktion mit IDORIS entwickelt. Ein FAIR-DO-Ops-Executor koordiniert die Ausführung von FAIR-DO-Ops primär auf 5 Abstraktionsebenen:

1. **Daten aus FAIR-DOs:** Die Menge der möglichen Operationen eines FAIR-DO bildet sich aus der Vereinigung der Operationen für die einzelnen Datentypen im FAIR-DO und der Operationen des Typprofils, welches die Struktur des FAIR-DO spezifiziert. Zur Auswahl der Operationen wird IDORIS nach jedem Datentyp und Typprofil gefragt und welche Operationen sie anbieten. Zukünftig könnte dafür eine spezielle API-Route in IDORIS implementiert werden, die alle Operationen eines FAIR-DO zurückgibt. Auf Basis der Operation kann der Nutzer die Ausführung eines FAIR-DO-Op anfordern. Für diese Operation wird ein Datenfluss in IDORIS berechnet (vergleichbar zu Abbildung 6.7), der alle beteiligten Operationen, Operationsschritte, FAIR-DO-OpTPs, AttributeMappings, Attribute und zugeordnete Datentypen enthält. Der Executor setzt die Werte aus dem FAIR-DO und mögliche Umgebungsvariablen (z. B. API-Schlüssel) ein, validiert die syntaktische Korrektheit sowie Vollständigkeit der Eingaben und berechnet daraus die Ausführungsreihenfolge mit den notwendigen FAIR-DO-OpTPs, Attributen und Attributwerten. Durch den Datenfluss wird sichergestellt, dass die Ausführung der Operationen in der richtigen Reihenfolge und mit den richtigen Werten erfolgt, sodass mögliche Abhängigkeiten zwischen einzelnen Operationsschritten berücksichtigt werden können.

2. **FAIR-DO-Op-Attribute:** Jedem Ausführungsschritt wird ein FAIR-DO-OpTP als auszuführende Technologie mit allen Attributen und den eingesetzten Werten zugeordnet. Ab diesem Schritt können durch die Validierung nach der Operationsberechnung im vorherigen Schritt keine falschen oder fehlenden Werte mehr auftreten. Nun muss die Ausführungsumgebung für das FAIR-DO-OpTP mithilfe der verfügbaren FAIR-DO-OpTPAs bereitgestellt werden. Eine Liste der verfügbaren FAIR-DO-OpTPAs wird von IDORIS zusammen mit den FAIR-DO-OpTPs zurückgegeben.

3. **FAIR Digital Object Operation Type Profile Adapters (FAIR-DO-OpTPAs):** In Abschnitt 4.3 wurden FAIR-DO-OpTPAs als Trennung der Ausführung von der Technologie eines FAIR-DO-OpTP vorgestellt. Ein FAIR-DO-OpTPA ist eine spezifische Implementierung eines FAIR-DO-OpTP für eine bestimmte Runtime. FAIR-DO-OpTPAs werden in Form eines FAIR-DO, die je nach Runtime ein unterschiedliches Typprofil instanziieren, verwaltet und sind somit maschineninterpretierbar. Inhaltlich spezifiziert ein FAIR-DO-OpTPA wie eine Technologie in einer bestimmten Umgebung ausgeführt wird, welche Attribute benötigt werden und wie die Ausführung gestartet wird. Beispielsweise werden für das FAIR-DO-OpTP `HTTP` die Attribute `URL`, `Method`, `Headers` und `Body` benötigt und mehrere FAIR-DO-OpTPAs referenziert. So existiert ein FAIR-DO-OpTPA für die Ausführung von Hypertext Transfer Protocol (HTTP)-Anfragen mit der `fetch`-API in einer JavaScript-Browserumgebung. Ein anderer FAIR-DO-OpTPA könnte die Ausführung von HTTP-Anfragen mit der `requests`-Bibliothek in einer Python-Umgebung ermöglichen. Die Auswahl eines passenden FAIR-DO-OpTPA erfolgt durch den Executor anhand der verfügbaren

Runtime-Umgebungen. Für besondere Sicherheitsanforderungen könnten Black- und Whitelisting von einzelnen FAIR-DO-OpTPAs und Herkunftsnachweise durch kryptografische Signaturen eingeführt werden.

4. **Runtime:** Die Runtime ist die Grundlage, auf der die FAIR-DO-OpTPAs im vorherigen Schritt gewählt wurden und nun ausgeführt werden. Beispielsweise könnte dies innerhalb eines Browsers JavaScript oder Web Assembly sein, für Containerumgebungen wären Umgebungen in beliebigen Sprachen (Java, Python, …) denkbar.
5. **Basisinfrastruktur:** Die Basisinfrastruktur stellt die notwendigen Ressourcen für die Ausführung von FAIR-DO-Ops bereit, wie z. B. Rechenleistung, Speicher und Netzwerk. Beispielsweise könnte dies ein Browser, eine Docker-Umgebung oder ein Kubernetes-Cluster sein. Auf der Basisinfrastruktur läuft die Runtime und wird bedarfsgerecht skaliert.

Diese grobe Struktur zeigt die Komplexität der Ausführung von FAIR-DO-Ops und die Notwendigkeit einer speziellen Komponente, die diese Ausführung koordiniert. Es ist zu erwarten, dass die Entwicklung einer solchen Komponente aufgrund der Vielzahl von Technologien und Runtimes, die für die Ausführung von FAIR-DO-Ops benötigt werden, ein ambitioniertes Projekt darstellt. Viele der Funktionen können durch spezielle Endpunkte in IDORIS vereinfacht werden, für deren Spezifikation und Implementierung jedoch ein tieferes Verständnis der Anforderungen des FAIR-DO-Ops-Executor an IDORIS notwendig sind. Dank der Erweiterbarkeit von IDORIS können diese API-Endpunkte jedoch einfach hinzugefügt werden.

8.3. Funktionserweiterungen

Es existieren bereits einige Ideen für Funktionserweiterungen von IDORIS, die den Rahmen dieser Arbeit gesprengt hätten, aber zukünftig umgesetzt werden können:

Die Entwicklung einer Web-Oberfläche für IDORIS vereinfacht die Nutzung für Endnutzer und erleichtert die Verwaltung von FAIR-DO-Ops und FAIR-DOs. Dabei sollte die Web-Oberfläche die gleichen Funktionen wie die API bereitstellen, um eine konsistente Nutzung zu ermöglichen. Validierungsergebnisse und Fehlermeldungen wurden bei der Implementierung von IDORIS bereits so gestaltet, dass sie in einer Web-Oberfläche einfach dargestellt werden können. Eine nette Funktionalität wäre die Anzeige von Graphen, die die Vererbungshierarchie und die Assoziationen von FAIR-DO-Ops und Datentypen visualisieren, sodass die komplexen Beziehungen zwischen den Elementen besser verstanden werden können. Außerdem könnte eine GraphQL-API [94, 95] bereitgestellt werden, um graphbasierte Anfragen von Nutzern effizienter zu bearbeiten und die Abfrage von komplexen Beziehungen zu vereinfachen. Die bestehende REST-API könnte an manchen Stellen etwas vereinfacht werden, sodass beispielsweise die Erstellung von Attributen direkt beim Erstellen eines Typprofils oder FAIR-DO-OpTP möglich ist.

In Sektion 8.1 wurde bereits das Referenzieren von IDORIS-Einträgen mit PIDs angesprochen. Der Typed PID Maker [53] ist ein Dienst, der typisierte FAIR-DOs im Handle-System erstellen kann, die Verwaltung von PIDs übernimmt und in der Service-Infrastruktur der Abteilung DEM ein zentraler Dienst für den Umgang mit FAIR-DOs ist. Einerseits können mit dem Typed PID Maker IDORIS-Einträge zusätzlich als typisierte FAIR-DOs veröffentlicht werden, um diese kritischen Informationen langfristig zugänglich zu machen. Dazu würden KIPs und Typprofile sowohl in der ePIC-DTR als auch IDORIS für die jeweiligen Eintragstypen erstellt, sodass eine Vorwärtskompatibilität ermöglicht wird. Im Handle-System sind schnelle Änderungen nicht vorgesehen, sodass vor der Veröffentlichung der Einträge in FAIR-DOs womöglich ein Versionierung- und Review-Prozess, beispielsweise wie bei der Veröffentlichung von Vokabularen in EVOKS [96], sinnvoll ist. Zudem könnten verschiedene Versionen der Entitäten in IDORIS vorgehalten und mit jeweils einer eigenen

PID referenziert werden, sodass große Änderungen nicht zwangsläufig zur Ungültigkeit früher erstellter FAIR-DOs führen. Andererseits kann ein IDORIS-Zugriff im Typed PID Maker hinzugefügt werden, sodass das neue vereinfachte Modell von IDORIS einfach mit FAIR-DOs genutzt werden kann. Diese Integration ermöglicht einen Zugriff des Typed PID Makers auf die Datentypen von IDORIS, welche zur Validierung der verwalteten FAIR-DOs genutzt werden. Ein Zugriff auf FAIR-DO-Ops-Executor könnte ebenfalls integriert werden, um die Ausführung von FAIR-DO-Ops auf Basis von FAIR-DOs zu vereinfachen und möglicherweise Ergebnisse in neue FAIR-DOs zu speichern.

Gemäß den Richtlinien der RDA sollen DTRs als verteilte Infrastruktur betrieben werden können [12], um die Verfügbarkeit und Skalierbarkeit zu erhöhen. Die ePIC-DTR ermöglicht dies, da kaum Verknüpfungen zwischen den Einträgen bestehen, die Einträge somit unabhängig voneinander sind und keine bedeutende Abfragelogik existiert. Für IDORIS ist dies, auch aufgrund der stark graphbasierten Struktur, nicht ohne weiteres möglich. Graphpartitionierung ist ein komplexes und aufwändiges Verfahren, um die Datenbank in mehrere Teile zu unterteilen, die unabhängig voneinander betrieben werden können. Eine einfachere Lösung ist die bilaterale Kommunikation und Kooperation zwischen mehreren Instanzen von IDORIS (z. B. zwischen Partnereinrichtungen oder verschiedenen Projektgruppen), die einander die Suche ermöglichen und bei Bedarf Einträge austauschen können. Für die Kommunikation von Änderungen können hochverfügbare Message-Broker, wie Apache Kafka [97] oder RabbitMQ [98], sowie Webhooks [99] genutzt werden, um die Änderungen zu publizieren und zu abonnieren. Dazu müssten in IDORIS zum einen zusätzliche Endpunkte und Logik realisiert werden und die Verknüpfung von entfernten Einträgen in der Datenbank umgesetzt werden. Aus Gründen der Technologieoffenheit für mögliche zukünftige DTRs, die auf dem Datenmodell von IDORIS aufbauen, sollte die Implementierung so gestaltet sein, dass sie auch in anderen Systemen genutzt werden kann. Daher ist die Verwendung von RMI-IIOP wahrscheinlich nicht sinnvoll.

In jedem Fall müssen die Datenbankeinträge für eine gute und effiziente Suche in Neo4j indexiert werden. Neo4j bietet entsprechende Möglichkeiten. Eine interessante Idee wäre die Integration der Neo4j-Datenbank in ein Large Language Model (LLM) mittels LangChain [100], was bereits für einfachere Graphen von Neo4j in einem interaktiven Kurs demonstriert wurde [101, 102, 103]. Dadurch könnten die Einträge in IDORIS in einem LLM genutzt werden, um die Suche und Verknüpfung von Einträgen zu erleichtern oder möglicherweise einfach neue Einträge zu erstellen. Gleichermaßen besteht die Möglichkeit, dass mithilfe eines LLM Daten verarbeitet, Operationen textbasiert über den FAIR-DO-Ops-Executor ausgeführt und die Ergebnisse evaluiert werden. Der Nutzen dieser Integration müsste genauestens unter Berücksichtigung der Technikfolgenabschätzung und Kosten evaluiert werden.

Ein Herausforderung ist die Einstiegshöhe für Anwender der ePIC-DTR. Dazu könnte ein Migrationsassistent entwickelt werden, der die Daten aus der ePIC-DTR in IDORIS importiert und dabei die Daten in das neue Datenmodell transformiert. Außerdem könnte ein designierter API-Endpunkt in IDORIS implementiert werden, der die Datentypen im ePIC-Format zurückgibt, um eine einfache Migration von bestehenden Systemen zu ermöglichen. Neue Funktionalitäten, wie beispielsweise die Zusammenführung von PID-InfoTypes und KIPs in Typprofile, FAIR-DO-Ops sowie die Vererbungslogik wären für diese Systeme nicht ohne weiteres möglich, sodass langfristig trotzdem eine Migration oder Integration notwendig ist. Die Entscheidung, ob eine derartige Migration ermöglicht werden soll, muss wirtschaftlich unter anderem anhand der Anzahl der Nutzer und der Anzahl der Einträge in der ePIC-DTR evaluiert werden.

Zusammengefasst, ermöglichen das konzeptuelle Datenmodell für typgebundene FAIR Digital Object Operations und dessen Realisierung mit IDORIS bereits jetzt maschinenprozessierbare FAIR-DOs. Durch diese Arbeit wurde eine solide Grundlage für die weitere Forschung an FAIR-DOs und Maschinenprozessierbarkeit geschaffen. IDORIS wird zukünftig im FDO-Forum [10] vorgestellt. Die Ergebnisse werden zu der gemeinsamen internationalen Forschung und Entwicklung von FAIR-DOs mit FAIR-DO-Ops beitragen.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

ALPS Application-Level Profile Semantics
API Application Programming Interface
APOC Awesome Procedures on Cypher
CORBA Common Object Request Broker Architecture
CRUD Create, Read, Update, Delete
DEM Data Exploitation Methods
DOI Digital Object Identifier
DOIP Digital Object Interface Protocol
DTR Data Type Registry
ePIC European Persistent Identifier Consortium
FAIR **F**indable, **A**ccessible, **I**nteroperable und **R**eusable
FAIR-DO FAIR Digital Object
FAIR-DO-Op FAIR Digital Object Operation
FAIR-DO-OpTP FAIR Digital Object Operation Type Profile
FAIR-DO-OpTPA FAIR Digital Object Operation Type Profile Adapter
FDO-Forum FAIR Digital Object Forum
HAL Hypertext Application Language
HATEOAS Hypermedia As The Engine Of Application State
HMC Helmholtz Metadata Collaboration Platform
HTML Hyper-Text Markup Language
HTTP Hypertext Transfer Protocol
IDORIS Integrated Data Type and Operations Registry with Inheritance System
IEEE Institute of Electrical and Electronics Engineers
ISBN Internationale Standardbuchnummer
ISO International Standards Organization
JNDI Java Naming and Directory Interface
JSON JavaScript Object Notation
KIP Kernel Information Profile
KIT Karlsruher Institut für Technologie
LLM Large Language Model
LTS Long Term Support
MVC Model-View-Controller
NFDI Nationale Forschungsdaten-Infrastruktur
OOP Objektorientierte Programmierung
ORCiD Open Researcher and Contributor ID
ORM Objekt-Relationales Mapping
OWL Web Ontology Language
PID Persistent Identifiers
PoC Proof-of-Concept
RDA Research Data Alliance
RDF Resource Description Framework

REST Representational State Transfer
RFC Request for Comment
RMI-IIOP Remote Method Invocation over Internet Inter-ORB Protocol
SCC Scientific Computing Center
SPARQL SPARQL Protocol and RDF Query Language
SPDX Software Package Data Exchange
SYS Systeme und Server
UI User Interface
URL Uniform Resource Locator
XML Extensible Markup Language

Literaturverzeichnis

[1] Directorate-General for Research and Innovation (European Commission). *Turning FAIR into Reality: Final Report and Action Plan from the European Commission Expert Group on FAIR Data*. Publications Office of the European Union, 2018. ISBN: 978-92-79-96546-3. URL: https://data.europa.eu/doi/10.2777/1524 (besucht am 02. 07. 2024).

[2] Claus Weiland, Sharif Islam, Daan Broder, Ivonne Anders und Peter Wittenburg. „FDO Machine Actionability". In: (19. Nov. 2022). DOI: 10.5281/zenodo.7825650. URL: https://zenodo.org/records/7825650 (besucht am 01. 07. 2024).

[3] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J. G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A. C. 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao und Barend Mons. „The FAIR Guiding Principles for Scientific Data Management and Stewardship". In: *Sci Data* 3.1 (15. März 2016), S. 160018. ISSN: 2052-4463. DOI: 10.1038/sdata.2016.18. URL: https://www.nature.com/articles/sdata201618 (besucht am 01. 07. 2024).

[4] Robert Kahn und Robert Wilensky. „A Framework for Distributed Digital Object Services". In: *Int J Digit Libr* 6.2 (1. Apr. 2006), S. 115–123. ISSN: 1432-1300. DOI: 10.1007/s00799-005-0128-x. URL: https://doi.org/10.1007/s00799-005-0128-x (besucht am 29. 05. 2024).

[5] Erik Schultes und Peter Wittenburg. „FAIR Principles and Digital Objects: Accelerating Convergence on a Data Infrastructure". In: *Data Analytics and Management in Data Intensive Domains*. Hrsg. von Yannis Manolopoulos und Sergey Stupnikov. Communications in Computer and Information Science. Cham: Springer International Publishing, 2019, S. 3–16. ISBN: 978-3-030-23584-0. DOI: 10.1007/978-3-030-23584-0_1.

[6] *ePIC Data Type Registry*. 10. Dez. 2021. URL: http://dtr-pit.pidconsortium.net/#urls/intro.html (besucht am 19. 07. 2024).

[7] Corporation for National Research Initiatives CNRI. *Cordra*. Version v2.5.2. 29. Aug. 2023. URL: https://gitlab.com/cnri/cordra/cordra.git (besucht am 02. 07. 2024).

[8] *FAIR Principles*. GO FAIR. URL: https://www.go-fair.org/fair-principles/ (besucht am 19. 07. 2024).

[9] Peter Wittenburg. *Digital Objects as Drivers towards Convergence in Data Infrastructures*. 2019. DOI: 10.23728/B2SHARE.B605D85809CA45679B110719B6C6CB11. URL: https://b2share.eudat.eu/records/b605d85809ca45679b110719b6c6cb11 (besucht am 01. 07. 2024).

[10] *FAIR Digital Objects Forum*. URL: https://fairdo.org/ (besucht am 18. 07. 2024).

[11] *Research Data Alliance | Home*. URL: https://www.rd-alliance.org/ (besucht am 18. 07. 2024).

[12] Larry Lannom, Daan Broeder und Giridhar Manepalli. „RDA Data Type Registries Working Group Output“. In: (30. Apr. 2015). DOI: 10.15497/A5BCD108-ECC4-41BE-91A7-20112FF77458. URL: https://zenodo.org/records/1406127 (besucht am 01. 07. 2024).

[13] *FAIR Digital Object Fabric IG Home*. URL: https://www.rd-alliance.org/groups/group-fair-digital-object-fabric-ig-1942598258/ (besucht am 18. 07. 2024).

[14] Jason Scott. *Helmholtz Metadata Collaboration (HMC)*. URL: https://helmholtz-metadaten.de/en (besucht am 18. 07. 2024).

[15] Constanze Curdt, Gerrit Günther, Thomas Jejkal, Christian Koch, Florian Krebs, Andreas Pfeil, Anton Pirogov, Jan Schweikert, Pedro Videgain Barranco und Martin Weinelt. *Helmholtz Metadata Collaboration, Helmholtz Kernel Information Profile*. Report. Kiel, Germany: HMC Office, GEOMAR Helmholtz Centre for Ocean Research, Dez. 2022. 35 S. DOI: 10.3289/HMC_publ_03. URL: https://oceanrep.geomar.de/id/eprint/57942/ (besucht am 15. 07. 2024).

[16] Herbert Van de Sompel. „FAIR Digital Objects and FAIR Signposting“. 27. Mai 2023. DOI: 10.5281/zenodo.7977333. URL: https://zenodo.org/records/7977333 (besucht am 18. 07. 2024).

[17] *Handle.Net Registry*. 22. März 2024. URL: https://handle.net/ (besucht am 18. 07. 2024).

[18] Ulrich Schwardmann. „Automated Schema Extraction for PID Information Types“. In: *2016 IEEE International Conference on Big Data (Big Data)*. 2016 IEEE International Conference on Big Data (Big Data). Dez. 2016, S. 3036–3044. DOI: 10.1109/BigData.2016.7840957. URL: https://ieeexplore.ieee.org/document/7840957 (besucht am 23. 07. 2024).

[19] *ORCID*. 8. Juli 2024. URL: https://orcid.org/ (besucht am 18. 07. 2024).

[20] *FDO elevator pitch(es)!* Google Docs. URL: https://docs.google.com/document/d/1iToZx9oIlYZDQcGyFcSQ9e3QsacAJ33gHuLPJeugUjA/edit?usp=drive_web&ouid=116917362010927683702&usp=embed_facebook (besucht am 19. 07. 2024).

[21] *ePIC PID Information Type Registries*. URL: http://www.pidconsortium.eu/ (besucht am 22. 07. 2024).

[22] *ePIC FAQs – Persistent Identifiers for eResearch*. URL: https://www.pidconsortium.net/?page_id=1060 (besucht am 22. 07. 2024).

[23] *Cordra*. 1. Sep. 2023. URL: https://www.cordra.org/ (besucht am 22. 07. 2024).

[24] Volker Hartmann, Thomas Jejkal und Sabrine Chelbi. *MetaStore*. Version 1.2.2. Feb. 2023. DOI: 10.5281/zenodo.7685007. URL: https://github.com/kit-data-manager/metastore2 (besucht am 22. 07. 2024).

[25] *Zenodo*. URL: https://zenodo.org/ (besucht am 30. 08. 2024).

[26] Luciano Floridi. *The Fourth Revolution How the Infosphere Is Reshaping Human Reality*. Oxford: Oxford university press, 2014. ISBN: 978-0-19-960672-6.

[27] Paul D. [R-WI-1 Rep. Ryan. *H.R.4174 - 115th Congress (2017-2018): An Act to Amend Titles 5 and 44, United States Code, to Require Federal Evaluation Activities, Improve Federal Data Management, and for Other Purposes*. 14. Jan. 2019. URL: https://www.congress.gov/bill/115th-congress/house-bill/4174 (besucht am 19. 07. 2024).

[28] *Machine-Readable Medium and Data*. In: *Wikipedia*. 21. Dez. 2023. URL: https://en.wikipedia.org/w/index.php?title=Machine-readable_medium_and_data&oldid=1191119094#Data (besucht am 19. 07. 2024).

[29] *Ontology (Information Science)*. In: *Wikipedia*. 11. Juli 2024. URL: https://en.wikipedia.org/w/index.php?title=Ontology_(information_science)&oldid=1233975174 (besucht am 19. 07. 2024).

[30] *Semantic Web*. In: *Wikipedia*. 23. Juli 2024. URL: https://en.wikipedia.org/w/index.php?title=Semantic_Web&oldid=1236248103 (besucht am 27. 08. 2024).

[31] *Knowledge Graph*. In: *Wikipedia*. 24. Juli 2024. URL: https://en.wikipedia.org/w/index.php?title=Knowledge_graph&oldid=1236424221 (besucht am 27. 08. 2024).

[32] *Linked Data*. In: *Wikipedia*. 6. Juni 2024. URL: https://en.wikipedia.org/w/index.php?title=Linked_data&oldid=1227615841 (besucht am 19. 07. 2024).

[33] Rainer Stotzka. „Introducing FAIR Digital Objects“ (Karlsruhe Institute of Technology). URL: https://docs.google.com/presentation/d/1zBhvHmxxgr9LtNjdEh_HyzHHGbXh5wTP (besucht am 19. 07. 2024).

[34] Peter Wittenburg. „Commenting on “Digital Object” Aspects“. In: (2019), 1 file, 1.3 MB. DOI: 10.23728/B2SHARE.2317B12321764F669C92EBBCF7518164. URL: https://b2share.eudat.eu/records/2317b12321764f669c92ebbcf7518164 (besucht am 23. 08. 2024).

[35] Koenraad de Smedt, Dimitris Koureas und Peter Wittenburg. „An Analysis of Scientific Practice towards FAIR Digital Objects“. In: ().

[36] Clemens Düpmeier. „Vorlesung "Verteilte Systeme"“. Vorlesung (Duale Hochschule Baden-Württemberg Karlsruhe). 2024.

[37] *ISO/IEC 2382:2015*. Information technology — Vocabulary. Version 2-10. Mai 2015. URL: https://www.iso.org/standard/63598.html (besucht am 22. 07. 2024).

[38] *Encapsulation (Computer Programming)*. In: *Wikipedia*. 2. Mai 2024. URL: https://en.wikipedia.org/w/index.php?title=Encapsulation_(computer_programming)&oldid=1221885910 (besucht am 25. 07. 2024).

[39] Alan Snyder. „Encapsulation and Inheritance in Object-Oriented Programming Languages“. In: *SIGPLAN Not.* 21.11 (1. Juni 1986), S. 38–45. ISSN: 0362-1340. DOI: 10.1145/960112.28702. URL: https://doi.org/10.1145/960112.28702 (besucht am 25. 07. 2024).

[40] Dr. Richard Lutz. „Software Engineering“. Vorlesung (Duale Hochschule Baden-Württemberg Karlsruhe). 2022.

[41] *Abstraction (Computer Science)*. In: *Wikipedia*. 6. Juni 2024. URL: https://en.wikipedia.org/w/index.php?title=Abstraction_(computer_science)&oldid=1227536907 (besucht am 25. 07. 2024).

[42] Barbara Liskov und Stephen Zilles. „Programming with Abstract Data Types“. In: *Proceedings of the ACM SIGPLAN Symposium on Very High Level Languages*. New York, NY, USA: Association for Computing Machinery, 28. März 1974, S. 50–59. ISBN: 978-1-4503-7884-0. DOI: 10.1145/800233.807045. URL: https://doi.org/10.1145/800233.807045 (besucht am 19. 07. 2024).

[43] Ulrich Schwardmann und Tibor Kálmán. „How FDO Attributes Can Support Machine- and Human-Readability? - a Description along Three Examples“. In: *Research Ideas and Outcomes* 9 (30. Okt. 2023), e108737. ISSN: 2367-7163. DOI: 10.3897/rio.9.e108737. URL: https://riojournal.com/article/108737/ (besucht am 23. 07. 2024).

[44] Koenraad De Smedt, Dimitris Koureas und Peter Wittenburg. „FAIR Digital Objects for Science: From Data Pieces to Actionable Knowledge Units“. In: *Publications* 8.2 (2 Juni 2020), S. 21. ISSN: 2304-6775. DOI: 10.3390/publications8020021. URL: https://www.mdpi.com/2304-6775/8/2/21 (besucht am 13. 11. 2023).

[45] *Open–Closed Principle*. In: *Wikipedia*. 23. Mai 2024. URL: https://en.wikipedia.org/w/index.php?title=Open%E2%80%93closed_principle&oldid=1225257299 (besucht am 16. 08. 2024).

[46] Daniel Lindner. „Advanced Software Engineering“. Vorlesung (Duale Hochschule Baden-Württemberg Karlsruhe). 2024.

[47] *Liskov Substitution Principle*. In: *Wikipedia*. 8. Juli 2024. URL: https://en.wikipedia.org/w/index.php?title=Liskov_substitution_principle&oldid=1233350358 (besucht am 16. 08. 2024).

[48] *Mixin*. In: *Wikipedia*. 7. Aug. 2024. URL: https://en.wikipedia.org/w/index.php?title=Mixin&oldid=1239089086 (besucht am 27. 08. 2024).

[49] *Vererbung (Programmierung)*. In: *Wikipedia*. 28. Apr. 2024. URL: https://de.wikipedia.org/w/index.php?title=Vererbung_(Programmierung)&oldid=244466276 (besucht am 25. 07. 2024).

[50] Larry Lannom, Ulrich Schwardmann, Christophe Blanchi und Peter Wittenburg. „Typing FAIR Digital Objects". In: (8. Juni 2022). DOI: 10.5281/zenodo.7825599. URL: https://zenodo.org/records/7825599 (besucht am 01. 07. 2024).

[51] *Generics in Java*. In: *Wikipedia*. 29. Juni 2024. URL: https://en.wikipedia.org/w/index.php?title=Generics_in_Java&oldid=1231696469 (besucht am 16. 08. 2024).

[52] *Polymorphie (Programmierung)*. In: *Wikipedia*. 24. Juli 2024. URL: https://de.wikipedia.org/w/index.php?title=Polymorphie_(Programmierung)&oldid=247064639 (besucht am 25. 07. 2024).

[53] Andreas Pfeil und Thomas Jejkal. *Typed PID Maker*. Okt. 2020. URL: https://github.com/kit-data-manager/pit-service (besucht am 07. 08. 2024).

[54] *Creating a JavaScript Action*. GitHub Docs. URL: https://docs.github.com/_next/data/HvnWGpT9_08fkDH2Fj8MP/en/free-pro-team@latest/actions/sharing-automations/creating-actions/creating-a-javascript-action.json?versionId=free-pro-team%40latest&productId=actions&restPage=sharing-automations&restPage=creating-actions&restPage=creating-a-javascript-action (besucht am 15. 08. 2024).

[55] *Java (Programmiersprache)*. In: *Wikipedia*. 27. Juni 2024. URL: https://de.wikipedia.org/w/index.php?title=Java_(Programmiersprache)&oldid=246266715 (besucht am 15. 08. 2024).

[56] James Gosling, Hrsg. *The Java Language Specification*. 3rd ed. Java Series. Upper Saddle River, NJ: Addison-Wesley, 2005. 651 S. ISBN: 978-0-321-24678-3.

[57] *Spring (Framework)*. In: *Wikipedia*. 28. Dez. 2023. URL: https://de.wikipedia.org/w/index.php?title=Spring_(Framework)&oldid=240593052 (besucht am 15. 08. 2024).

[58] *Home*. Spring.io Home. URL: https://spring.io/ (besucht am 15. 08. 2024).

[59] *Spring Boot*. Spring Boot. URL: https://spring.io/projects/spring-boot (besucht am 16. 08. 2024).

[60] *Spring Web MVC :: Spring Framework*. URL: https://docs.spring.io/spring-framework/reference/web/webmvc.html (besucht am 16. 08. 2024).

[61] *Spring Data REST*. Spring Data REST. URL: https://spring.io/projects/spring-data-rest (besucht am 16. 08. 2024).

[62] *Spring Data REST :: Spring Data REST*. URL: https://docs.spring.io/spring-data/rest/reference/index.html (besucht am 16. 08. 2024).

[63] *Spring HATEOAS*. Spring HATEOAS. URL: https://spring.io/projects/spring-hateoas (besucht am 16. 08. 2024).

[64] *Spring Data Neo4j*. Spring Data Neo4j. URL: https://spring.io/projects/spring-data-neo4j (besucht am 16. 08. 2024).

[65] *Tools :: Spring Data REST*. URL: https://docs.spring.io/spring-data/rest/reference/tools.html#tools.hal-explorer (besucht am 16. 08. 2024).

[66] *Project Lombok*. URL: https://projectlombok.org/ (besucht am 16. 08. 2024).

[67] *Neo4j Graph Database & Analytics – The Leader in Graph Databases*. Graph Database & Analytics. URL: https://neo4j.com/ (besucht am 16. 08. 2024).

[68] *Introduction - Cypher Manual*. Neo4j Graph Data Platform. 14. Aug. 2024. URL: https://neo4j.com/docs/cypher-manual/5/introduction/ (besucht am 16. 08. 2024).

[69] *Apache Jena - Home*. 18. Juli 2024. URL: https://jena.apache.org/ (besucht am 16. 08. 2024).

[70] *Apache Jena - TDB vs. Neo4j Comparison*. URL: https://db-engines.com/en/system/Apache+Jena+-+TDB%3BNeo4j (besucht am 24. 01. 2024).

[71] *Understanding Neo4j's Data on Disk - Knowledge Base*. Neo4j Graph Data Platform. 18. Jan. 2024. URL: https://neo4j.com/developer/kb/understanding-data-on-disk/ (besucht am 23. 01. 2024).

[72] *Neo4j Performance Architecture Explained & 6 Tuning Tips*. 30. Sep. 2021. URL: https://www.graphable.ai/blog/neo4j-performance/ (besucht am 23. 01. 2024).

[73] Rachel Howard. *RDF Triple Stores vs. Labeled Property Graphs: What's the Difference?* Graph Database & Analytics. 18. Aug. 2017. URL: https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/ (besucht am 23. 01. 2024).

[74] Bryce Merkl Sasaki. *Native vs. Non-Native Graph Database*. Graph Database & Analytics. 8. Mai 2023. URL: https://neo4j.com/blog/native-vs-non-native-graph-technology/ (besucht am 23. 01. 2024).

[75] *Awesome Procedures On Cypher (APOC) - Neo4j Labs*. 20. Aug. 2024. URL: https://neo4j.com/labs/apoc/ (besucht am 28. 08. 2024).

[76] *Graph Data Science*. Graph Database & Analytics. 11. Sep. 2024. URL: https://neo4j.com/product/graph-data-science/ (besucht am 28. 08. 2024).

[77] *Neo4j Documentation - Neo4j Documentation*. Neo4j Graph Data Platform. URL: https://neo4j.com/docs/docs/ (besucht am 16. 08. 2024).

[78] *Free, Self-Paced, Hands-on Online Training*. Neo4j Graph Academy. URL: https://graphacademy.neo4j.com/ (besucht am 21. 08. 2024).

[79] *Spring Data Neo4j :: Spring Data Neo4j*. URL: https://docs.spring.io/spring-data/neo4j/reference/index.html (besucht am 16. 08. 2024).

[80] *HATEOAS*. In: *Wikipedia*. 17. Juni 2024. URL: https://en.wikipedia.org/w/index.php?title=HATEOAS&oldid=1229543187 (besucht am 28. 08. 2024).

[81] M. Amundsen. *Application-Level Profile Semantics (ALPS)*. 28. Feb. 2015. URL: http://www.alps.io/spec/drafts/draft-01.html (besucht am 28. 08. 2024).

[82] Alexander Shvets. *Visitor*. 13. Aug. 2024. URL: https://refactoring.guru/design-patterns/visitor (besucht am 16. 08. 2024).

[83] Alexander Shvets. *Dive Into Design Patterns*. v2021-2.28. 2021. URL: https://refactoring.guru/design-patterns/book (besucht am 01. 07. 2024).

[84] Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides. *Design Patterns : Entwurfsmuster als Elemente wiederverwendbarer objektorientierter Software*. mitp Professional. Frechen: mitp Verlags GmbH & Co. KG, 2015. ISBN: 978-3-8266-9700-5. URL: http://www.redi-bw.de/db/ebsco.php/search.ebscohost.com/login.aspx%3fdirect%3dtrue%26db%3dnlebk%26AN%3d967796%26site%3dehost-live (besucht am 25. 07. 2024).

[85] Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman und Michael Leuschel, Hrsg. *Compiler: Prinzipien, Techniken und Werkzeuge*. 2., aktualisierte Aufl. [der engl. Ausg.] it informatik. München: Pearson Studium, 2008. 1253 S. ISBN: 978-3-8273-7097-6.

[86] Andrew W. Appel und Jens Palsberg. *Modern Compiler Implementation in Java*. 2nd ed. Cambridge, UK ; New York, NY, USA: Cambridge University Press, 2002. 501 S. ISBN: 978-0-521-82060-8.

[87] Marco Haupt. „Compilerbau“ (Duale Hochschule Baden-Württemberg Karlsruhe). 2023. URL: Vorlesung.

[88] *Double Dispatch*. In: *Wikipedia*. 8. Aug. 2022. URL: https://en.wikipedia.org/w/index.php?title=Double_dispatch&oldid=1103211310 (besucht am 16. 08. 2024).

[89] *Working with Relationships in Spring Data REST | Baeldung*. 28. Feb. 2017. URL: https://www.baeldung.com/spring-data-rest-relationships (besucht am 26. 08. 2024).

[90] *OpenAPI Initiative | Home*. OpenAPI Initiative. URL: https://www.openapis.org/ (besucht am 29. 08. 2024).

[91] *Neo4jClient :: Spring Data Neo4j*. URL: https://docs.spring.io/spring-data/neo4j/reference/appendix/neo4j-client.html (besucht am 24. 08. 2024).

[92] *Javadoc*. In: *Wikipedia*. 28. Dez. 2023. URL: https://en.wikipedia.org/w/index.php?title=Javadoc&oldid=1192264961 (besucht am 29. 08. 2024).

[93] *API Documentation & Design Tools for Teams | Swagger*. URL: https://swagger.io/ (besucht am 29. 08. 2024).

[94] *GraphQL | A Query Language for Your API*. URL: https://graphql.org/ (besucht am 25. 08. 2024).

[95] *Spring for GraphQL :: Spring GraphQL*. URL: https://docs.spring.io/spring-graphql/reference/index.html (besucht am 25. 08. 2024).

[96] Felix Ernst. *Kit-Data-Manager/EVOKS*. KIT Data Manager, 21. Sep. 2022. URL: https://github.com/kit-data-manager/EVOKS (besucht am 21. 09. 2022).

[97] *Apache Kafka*. Apache Kafka. URL: https://kafka.apache.org/ (besucht am 29. 08. 2024).

[98] *RabbitMQ: One Broker to Queue Them All | RabbitMQ*. URL: https://www.rabbitmq.com/ (besucht am 29. 08. 2024).

[99] *Webhook*. In: *Wikipedia*. 27. Juli 2024. URL: https://en.wikipedia.org/w/index.php?title=Webhook&oldid=1236882775 (besucht am 29. 08. 2024).

[100] *LangChain*. URL: https://www.langchain.com/ (besucht am 25. 08. 2024).

[101] *Take the Build a Neo4j-backed Chatbot Using Python Course with Neo4j GraphAcademy*. URL: https://graphacademy.neo4j.com/courses/llm-chatbot-python/ (besucht am 25. 08. 2024).

[102] *Take the Introduction to Vector Indexes and Unstructured Data Course with Neo4j GraphAcademy*. URL: https://graphacademy.neo4j.com/courses/llm-vectors-unstructured/ (besucht am 25. 08. 2024).

[103] *Take the Neo4j & LLM Fundamentals Course with Neo4j GraphAcademy*. URL: https://graphacademy.neo4j.com/courses/llm-fundamentals/ (besucht am 25. 08. 2024).

A. Anhang

A.1. IDORIS

Der Quellcode für IDORIS ist auf GitHub in folgendem Repository zu finden: `https://github.com/maximiliani/idoris`

Zum Starten der Neo4j-Datenbank mit den Zugangsdaten `neo4j:superSecret` muss folgender Befehl ausgeführt werden:

```
docker run \
   -p 7474:7474 -p 7687:7687 \
   --name neo4j-apoc-gds-idoris \
   --volume=$(pwd)/neo4j-data:/data \
   -e NEO4J_AUTH=neo4j/superSecret \
   -e NEO4J_DEBUG=true \
   -e NEO4J_ACCEPT_LICENSE_AGREEMENT=yes \
   -e NEO4J_apoc_export_file_enabled=true \
   -e NEO4J_apoc_import_file_enabled=true \
   -e NEO4J_apoc_import_file_use__neo4j__config=true \
   -e NEO4J_PLUGINS=\[\"apoc\",\"graph-data-science\"\,\"bloom\"] \
   -e NEO4J_dbms_security_procedures_unrestricted=apoc.\\\*,gds.\\\* \
   neo4j:latest
```

Listing A.1: Starten der Neo4j-Datenbank

Vorausgesetzt, dass die Neo4j-Datenbank auf dem gleichen Host läuft, kann folgender Inhalt als Musterkonfiguration für IDORIS in der `application.properties`-Datei verwendet werden.

```
spring.application.name=idoris
logging.level.root=INFO
spring.neo4j.uri=bolt://localhost:7687
spring.neo4j.authentication.username=neo4j
spring.neo4j.authentication.password=superSecret
spring.data.rest.basePath=/api
server.port=8095
idoris.validation-level=info
idoris.validation-policy=strict
```

A.2. JSON-Schemata aus der ePIC-DTR

Im Folgenden sind die JSON-Schemata für PID-BasicInfoTypes, PID-InfoTypes und Kernel Information Profile aus der Testinstanz der ePIC-Data Type Registry (`https://dtr-test.pidconsortium.eu/#objects/?query=type:%22Schema%22`) abgedruckt.

A.2.1. Schema für PID-BasicInfoTypes

```
{
  "identifier": "21.T11148/8ab89513e7f7fbda1d34",
  "name": "PID-BasicInfoType",
  "description": "Schema used to define basic PID Info Types by a data type, restrictions
   and regular expression. \nWith the regular expression itself the underlying standard
   or flavour of the regular expression used has to be specified.",
  "schema": {
    "type": "object",
    "format": "table",
    "required": [
      "name"
    ],
    "properties": {
      "identifier": {
        "type": "string",
        "net.cnri.repository": {
          "type": {
            "autoGeneratedField": "handle"
          }
        }
      },
      "name": {
        "type": "string",
        "maxLength": 128,
        "title": "Type Name",
        "pattern": "^([!-~])+$",
        "description": "please use printable ascii characters without blank",
        "net.cnri.repository": {
          "preview": {
            "showInPreview": true,
            "isPrimary": true
          }
        }
      },
      "description": {
        "type": "string",
        "maxLength": 2048,
        "title": "Description",
        "pattern": "(.|\n)*",
```

```
        "net.cnri.repository": {
          "preview": {
            "showInPreview": true,
            "isPrimary": true
          }
        }
      },
      "standards": {
        "type": "array",
        "format": "table",
        "title": "Applicable Standards or Recommendations",
        "uniqueItems": true,
        "items": {
          "type": "object",
          "title": "Standard",
          "required": [
            "issuer",
            "name"
          ],
          "properties": {
            "natureOfApplicability": {
              "title": "Nature of Applicability",
              "type": "string",
              "enum": [
                "extends",
                "constrains",
                "specifies",
                "depends",
                "is_previous_version_of",
                "is_new_version_of",
                "is_semantically_identical",
                "is_semantically_similar"
              ]
            },
            "name": {
              "title": "Standard Name",
              "type": "string",
              "maxLength": 1024,
              "description": "Type ID or standard number/name"
            },
            "issuer": {
              "title": "Issued By",
              "type": "string",
              "net.cnri.repository": {
                "type": {
                  "suggestedVocabulary": [
                    "DTR",
                    "ISO",
                    "W3C",
```

```
                    "ITU",
                    "RFC"
                  ]
                }
              }
            },
            "details": {
              "title": "Details",
              "type": "string",
              "format": "textarea",
              "maxLength": 2048
            }
          }
        }
      },
      "provenance": {
        "type": "object",
        "title": "Provenance",
        "properties": {
          "contributors": {
            "type": "array",
            "format": "table",
            "title": "Contributors of this Record",
            "items": {
              "title": "Contributor",
              "type": "object",
              "required": [
                "identifiedUsing",
                "name"
              ],
              "properties": {
                "identifiedUsing": {
                  "title": "Identified Using",
                  "type": "string",
                  "net.cnri.repository": {
                    "type": {
                      "suggestedVocabulary": [
                        "Handle",
                        "ORCID",
                        "URL",
                        "Text"
                      ]
                    }
                  }
                },
                "name": {
                  "title": "Name",
                  "type": "string",
                  "maxLength": 2048
```

```
            },
            "details": {
              "title": "Details",
              "type": "string",
              "format": "textarea",
              "maxLength": 1024
            }
          }
        }
      },
      "creationDate": {
        "title": "Creation Date",
        "type": "string",
        "format": "datetime",
        "net.cnri.repository": {
          "type": {
            "autoGeneratedField": "creationDate"
          }
        }
      },
      "lastModificationDate": {
        "title": "Last Modification Date",
        "type": "string",
        "format": "datetime",
        "net.cnri.repository": {
          "type": {
            "autoGeneratedField": "modificationDate"
          }
        }
      }
    }
  },
  "representationsAndSemantics": {
    "type": "array",
    "format": "table",
    "title": "Representations and Semantics",
    "uniqueItems": true,
    "items": {
      "type": "object",
      "title": "Representation and Semantic Expression",
      "required": [
        "expression",
        "value"
      ],
      "properties": {
        "expression": {
          "title": "Expression",
          "type": "string",
          "net.cnri.repository": {
```

```
                "type": {
                  "suggestedVocabulary": [
                    "Measurement Unit",
                    "Format",
                    "Character Set",
                    "Encoding",
                    "Other"
                  ]
                }
              }
            },
            "value": {
              "title": "Value",
              "description": "Unicode, UTF-8, Meter, etc.",
              "type": "string",
              "maxLength": 1024
            },
            "details": {
              "title": "Details",
              "type": "string",
              "format": "textarea",
              "maxLength": 2048
            }
          }
        }
      },
      "properties": {
        "type": "array",
        "title": "Properties",
        "description": "Restrictions on the type like regular expressions, enumeration max
    /min values etc,",
        "items": {
          "type": "object",
          "title": "Property",
          "required": [
            "dataType"
          ],
          "properties": {
            "expression": {
              "title": "Expression",
              "type": "string",
              "net.cnri.repository": {
                "type": {
                  "suggestedVocabulary": [
                    "Format",
                    "Character Set",
                    "Encoding",
                    "Measurement Unit"
                  ]
```

```
            }
          }
        },
        "value": {
          "title": "Value",
          "description": "For measurement units the name of the quantity (meter,
    second, Joule etc).",
          "type": "string",
          "maxLength": 1024
        },
        "symbol": {
          "title": "Symbol",
          "description": "For measurement units the symbol used for the quantity (m, s
    , J etc).",
          "type": "string",
          "maxLength": 1024
        },
        "definedBy": {
          "title": "Defined By",
          "description": "For derived measurement units the definition given as an
    expression of operator linked type identifiers. Allowed operators are: +. -, *, /, ^,
    (, ) For numerical constants here only the value is given as string. Numerical
    constant values x have to be additionally described in the restrictions area by  \"
    minimum\": x, \"maximum\": x in order to provide also a correct schema for such types
    .",
          "type": "string",
          "maxLength": 65536
        },
        "dataType": {
          "type": "string",
          "maxLength": 1024,
          "title": "Data Type",
          "net.cnri.repository": {
            "type": {
              "suggestedVocabulary": [
                "string",
                "integer",
                "number",
                "boolean",
                "NoType"
              ]
            }
          }
        },
        "standard_uncertainty": {
          "title": "Standard Uncertainty",
          "description": "Relative standard uncertainty (i.e. u_r)",
          "type": "string",
          "maxLength": 1024
```

```
275             },
276             "restrict": {
277               "type": "string",
278               "maxLength": 4096,
279               "title": "Restrictions",
280               "description": "Define type dependent restrictions with special keywords
    like multipleOf, minimum, maximum, exclusiveMinimum and exclusiveMaximum for numbers
    and minLength, maxLength, enum or formats like date-time, email, hostname, ipv4, ipv6
    or uri for strings. Examples: \"minimum\": -3.8, \"maximum\": -9.7  OR  \"minLength\":
     5 , \"maxLength\": 6 OR \"format\" : \"time\" ",
281               "net.cnri.repository": {
282                 "preview": {
283                   "showInPreview": true,
284                   "isPrimary": true
285                 }
286               }
287             },
288             "regexp": {
289               "type": "string",
290               "maxLength": 4096,
291               "title": "Regular Expression",
292               "description": "only used with data type string",
293               "net.cnri.repository": {
294                 "preview": {
295                   "showInPreview": true,
296                   "isPrimary": true
297                 }
298               }
299             },
300             "enum": {
301               "type": "string",
302               "maxLength": 262144,
303               "title": "Enumeration List",
304               "description": "Needs to be compatible with data type: For mixed types like
     [\"a\", \"bc\", 123, \"def\"] use NoType",
305               "net.cnri.repository": {
306                 "preview": {
307                   "showInPreview": true,
308                   "isPrimary": true
309                 }
310               }
311             },
312             "default": {
313               "type": "string",
314               "maxLength": 262144,
315               "title": "Default Value",
316               "description": "A default value can be provided here. Needs to be a string
    representation of a JSON object that is compatible with these restrictions. String
    delimiters need to be escaped",
```

```
            "net.cnri.repository": {
              "preview": {
                "showInPreview": true,
                "isPrimary": true
              }
            }
          },
          "flavour": {
            "type": "string",
            "maxLength": 1024,
            "title": "Flavour of RegExp",
            "description": "required with Regular Expression",
            "net.cnri.repository": {
              "type": {
                "suggestedVocabulary": [
                  "ecma-262-RegExp",
                  "php-RegExp",
                  "javascript-RegExp",
                  "python-RegExp"
                ]
              }
            }
          }
        }
      },
      "validationSchema": {
        "title": "Validation Schema",
        "type": "string",
        "description": "Schema used to validate an instance of this type ",
        "format": "textarea",
        "maxLength": 1048575
      }
    }
  }
}
```

Listing A.2: Schema für PID-BasicInfoTypes

A.2.2. Schema für PID-InfoTypes

```
{
  "identifier": "21.T11148/b72cf35b541e2ef79830",
  "name": "PID-InfoType",
  "description": "Schema used to define PID Info Types dependent on PID Info Types or
    Basic PID Info Types given by a list together with parameters describing the kind of
    dependency.",
```

```
  "schema": {
    "type": "object",
    "required": [
      "name",
      "description"
    ],
    "properties": {
      "identifier": {
        "type": "string",
        "net.cnri.repository": {
          "type": {
            "autoGeneratedField": "handle"
          }
        }
      },
      "name": {
        "type": "string",
        "maxLength": 128,
        "title": "Type Name",
        "net.cnri.repository": {
          "preview": {
            "showInPreview": true,
            "isPrimary": true
          }
        }
      },
      "description": {
        "type": "string",
        "format": "textarea",
        "maxLength": 2048,
        "title": "Description",
        "net.cnri.repository": {
          "preview": {
            "showInPreview": true,
            "excludeTitle": true
          }
        }
      },
      "standards": {
        "type": "array",
        "format": "table",
        "title": "Applicable Standards or Recommendations",
        "uniqueItems": true,
        "items": {
          "type": "object",
          "title": "Standard",
          "required": [
            "issuer",
            "name"
```

```
        ],
        "properties": {
          "natureOfApplicability": {
            "title": "Nature of Applicability",
            "type": "string",
            "enum": [
              "depends",
              "extends",
              "constrains",
              "specifies",
              "is_previous_version_of",
              "is_new_version_of",
              "is_semantically_identical",
              "is_semantically_similar"
            ]
          },
          "name": {
            "title": "Standard Name",
            "type": "string",
            "maxLength": 1024,
            "description": "Type ID or standard number/name"
          },
          "issuer": {
            "title": "Issued By",
            "type": "string",
            "net.cnri.repository": {
              "type": {
                "suggestedVocabulary": [
                  "DTR",
                  "ISO",
                  "W3C",
                  "ITU",
                  "RFC"
                ]
              }
            }
          },
          "details": {
            "title": "Details",
            "type": "string",
            "format": "textarea",
            "maxLength": 2048
          }
        }
      },
      "provenance": {
        "type": "object",
        "title": "Provenance",
```

```
        "properties": {
          "contributors": {
            "type": "array",
            "format": "table",
            "title": "Contributors of this Record",
            "items": {
              "title": "Contributor",
              "type": "object",
              "required": [
                "identifiedUsing",
                "name"
              ],
              "properties": {
                "identifiedUsing": {
                  "title": "Identified Using",
                  "type": "string",
                  "net.cnri.repository": {
                    "type": {
                      "suggestedVocabulary": [
                        "Handle",
                        "ORCID",
                        "URL",
                        "Text"
                      ]
                    }
                  }
                },
                "name": {
                  "title": "Name",
                  "type": "string",
                  "maxLength": 2048
                },
                "details": {
                  "title": "Details",
                  "type": "string",
                  "format": "textarea",
                  "maxLength": 1024
                }
              }
            }
          },
          "creationDate": {
            "title": "Creation Date",
            "type": "string",
            "format": "datetime",
            "net.cnri.repository": {
              "type": {
                "autoGeneratedField": "creationDate"
              }
```

```
            }
          },
          "lastModificationDate": {
            "title": "Last Modification Date",
            "type": "string",
            "format": "datetime",
            "net.cnri.repository": {
              "type": {
                "autoGeneratedField": "modificationDate"
              }
            }
          }
        }
      },
      "expectedUses": {
        "type": "array",
        "maxItems": 3,
        "format": "table",
        "title": "Expected Uses",
        "items": {
          "type": "string",
          "title": "Use",
          "format": "textarea",
          "maxLength": 4096
        }
      },
      "representationsAndSemantics": {
        "type": "array",
        "format": "table",
        "title": "Representations and Semantics",
        "uniqueItems": true,
        "items": {
          "type": "object",
          "title": "Representation and Semantic Expression",
          "required": [
            "subSchemaRelation",
            "allowAbbreviatedForm"
          ],
          "properties": {
            "expression": {
              "title": "Expression",
              "type": "string",
              "net.cnri.repository": {
                "type": {
                  "suggestedVocabulary": [
                    "Format",
                    "Character Set",
                    "Encoding",
                    "Measurement Unit"
```

```
                ]
              }
            }
          },
          "value": {
            "title": "Value",
            "description": "Unicode, UTF-8, Meter, etc.",
            "type": "string",
            "maxLength": 1024
          },
          "subSchemaRelation": {
            "title": "Relations for Sub-Schemas in Properties-Section",
            "description": "Describes, how the properties, the type is derived from, are
     related to each other. Each of them is assumed to be either mandatory or optional,
    which is defined as an obligation for each individual property below. With \"
    denyAdditionalProperties\" one defines an object with given properties below where
    additional properties are not allowed. With \"requestAll/Any/OneOfProperties\" and \"
    isNot\" the property obligations below are not reflected (possible contradictions) and
     all/any/exactly_one of the properties have to be given in the object or \"isNot\"
    negates the sub-schemas of the properties. With \"isArrayWithGivenProperties\" one
    defines arrays. If only one property is given, the array contains arbitrary many
    elements of only the given property (list validation, but with \"minItems\" : 1). If
    more than one property is given, the array contains a tuple of the given properties (
    tuple validation, but with  \"additionalItems\": false). With \"isSet\" an array is
    defined that is unordered, unchangeable, does not allow duplicate values and does not
    rely on the indices. With \"isTuple\" an array is defined that is unchangeable.",
            "type": "string",
            "net.cnri.repository": {
              "type": {
                "suggestedVocabulary": [
                  "denyAdditionalProperties",
                  "requestAllOfProperties",
                  "requestAnyOfProperties",
                  "requestOneOfProperties",
                  "isNot",
                  "isArrayWithGivenProperties",
                  "isSet",
                  "isTuple"
                ],
                "default": "denyAdditionalProperties"
              }
            }
          },
          "allowAbbreviatedForm": {
            "title": "Abbreviated Form",
            "description": "Usually entries are represented by a key value pair. To
    simplify notation it can also be allowed to present the entries also (Both) or only (
    Yes) as list in an array or just as the object. In this case all entries are treated
    as having the mandatory property (see Properties section below).",
```

```
            "type": "string",
            "net.cnri.repository": {
              "type": {
                "suggestedVocabulary": [
                  "No",
                  "Yes",
                  "Both"
                ],
                "default": "No"
              }
            }
          },
          "restrict": {
            "type": "string",
            "maxLength": 4096,
            "title": "Restrictions",
            "description": "Define type dependent restrictions with special keywords like  uniqueItems, maxItems, minItems or minProperties and maxProperties. Examples: \" uniqueItems\": true, \"maxItems\": 9",
            "net.cnri.repository": {
              "preview": {
                "showInPreview": true,
                "isPrimary": true
              }
            }
          },
          "default": {
            "type": "string",
            "maxLength": 262144,
            "title": "Default Value",
            "description": "A default value can be provided here. Needs to be a string representation of a JSON object that is compatible with these restrictions. String delimiters need to be escaped",
            "net.cnri.repository": {
              "preview": {
                "showInPreview": true,
                "isPrimary": true
              }
            }
          },
          "details": {
            "title": "Details",
            "type": "string",
            "format": "textarea",
            "maxLength": 2048
          }
        }
      }
    },
```

```
    "properties": {
      "type": "array",
      "title": "Properties",
      "description": "Type dependencies used for expressing how this type is built from other types",
      "items": {
        "type": "object",
        "headerTemplate": "{{self.name}}",
        "title": "Property",
        "required": [
          "identifier",
          "name"
        ],
        "properties": {
          "name": {
            "type": "string",
            "description": "Name assigned to dependent type in this context",
            "title": "Name",
            "pattern": "^([!-~])+$",
            "maxLength": 256
          },
          "identifier": {
            "type": "string",
            "title": "TID of Existing Data Type",
            "net.cnri.repository": {
              "type": {
                "handleReference": {
                  "types": [
                    "PID-InfoType",
                    "PID-BasicInfoType"
                  ],
                  "name": "{{../name}}"
                }
              }
            }
          },
          "representationsAndSemantics": {
            "type": "array",
            "format": "table",
            "title": "Representations and Semantics of Dependent Type",
            "description": "Restrictions on representations and semantics of the dependent type in this context",
            "uniqueItems": true,
            "items": {
              "type": "object",
              "title": "Representation and Semantic Expression",
              "required": [
                "repeatable",
                "obligation"
```

```
                ],
                "properties": {
                  "expression": {
                    "title": "Expression",
                    "description": "Purpose this property is used for",
                    "type": "string",
                    "net.cnri.repository": {
                      "type": {
                        "suggestedVocabulary": [
                          "Format",
                          "Character Set",
                          "Encoding",
                          "Measurement Unit",
                          "Display Option",
                          "Search Option"
                        ],
                        "default": "Measurement Unit"
                      }
                    }
                  },
                  "value": {
                    "title": "Value",
                    "description": "Used if the type has a constant value on all possible instances for this parameter. Needs to be a string representation of a JSON object that is compatible with the restrictions given by the here referred type",
                    "type": "string",
                    "maxLength": 262144
                  },
                  "obligation": {
                    "type": "string",
                    "minItems": 1,
                    "format": "table",
                    "description": "Optional/Mandatory",
                    "title": "Obligation",
                    "net.cnri.repository": {
                      "type": {
                        "suggestedVocabulary": [
                          "Optional",
                          "Mandatory"
                        ],
                        "default": "Mandatory"
                      }
                    }
                  },
                  "repeatable": {
                    "type": "string",
                    "minItems": 1,
                    "format": "table",
                    "title": "Repeatable",
```

```
                    "description": "Cardinality of this property",
                    "net.cnri.repository": {
                      "type": {
                        "suggestedVocabulary": [
                          "Yes",
                          "No"
                        ]
                      }
                    }
                  },
                  "allowOmitSubsidiaries": {
                    "title": "Omit Name as Subsidiary",
                    "description": "Name of the reference to the subtype: if Yes, it is only subsidiary and can be omitted.",
                    "type": "string",
                    "net.cnri.repository": {
                      "type": {
                        "suggestedVocabulary": [
                          "No",
                          "Yes"
                        ],
                        "default": "No"
                      }
                    }
                  },
                  "details": {
                    "title": "Details",
                    "type": "string",
                    "format": "textarea",
                    "maxLength": 2048
                  }
                }
              }
            }
          }
        }
      },
      "relationships": {
        "type": "array",
        "format": "table",
        "title": "Experimental. Likely to change soon",
        "description": "Intent: How the properties are related to each other, e.g., grouping of properties, cardinality, etc., should be captured here.",
        "items": {
          "type": "object",
          "title": "Relationship",
          "required": [
            "name",
            "relativeNames"
```

```
          ],
          "properties": {
            "name": {
              "title": "Name of Relationship",
              "type": "string",
              "maxLength": 256
            },
            "relativeNames": {
              "type": "array",
              "minItems": 1,
              "format": "table",
              "title": "Relative Names",
              "items": {
                "type": "string",
                "title": "Name of Property"
              }
            },
            "details": {
              "title": "Details",
              "type": "string",
              "format": "textarea",
              "maxLength": 2048
            }
          }
        },
        "validationSchema": {
          "title": "Validation Schema",
          "type": "string",
          "description": "Schema used to validate an instance of this type ",
          "format": "textarea",
          "maxLength": 1048575
        }
      }
    }
}
```

Listing A.3: Schema für PID-InfoTypes

A.2.3. Schema für Kernel Information Profile

```
{
  "identifier": "21.T11148/532ce6796e2828dd2be6",
  "name": "KernelInformationProfile",
  "description": "Schema for Kernel Information profiles, describing which attributes must
    or may be included in a conforming Kernel Information record.\n\n\nSchema used to
```

```
    define PID Info Types dependent on PID Info Types or Basic PID Info Types given by a
    list together with parameters describing the kind of dependency.",
  "schema": {
    "type": "object",
    "required": [
      "name",
      "description"
    ],
    "properties": {
      "identifier": {
        "type": "string",
        "net.cnri.repository": {
          "type": {
            "autoGeneratedField": "handle"
          }
        }
      },
      "name": {
        "type": "string",
        "maxLength": 128,
        "title": "Type Name",
        "net.cnri.repository": {
          "preview": {
            "showInPreview": true,
            "isPrimary": true
          }
        }
      },
      "description": {
        "type": "string",
        "format": "textarea",
        "maxLength": 2048,
        "title": "Description",
        "net.cnri.repository": {
          "preview": {
            "showInPreview": true,
            "excludeTitle": true
          }
        }
      },
      "standards": {
        "type": "array",
        "format": "table",
        "title": "Applicable Standards or Recommendations",
        "uniqueItems": true,
        "items": {
          "type": "object",
          "title": "Standard",
          "required": [
```

```
            "issuer",
            "name"
          ],
          "properties": {
            "natureOfApplicability": {
              "title": "Nature of Applicability",
              "type": "string",
              "enum": [
                "depends",
                "extends",
                "constrains",
                "specifies",
                "is_new_version_of",
                "is_semantically_identical",
                "is_semantically_similar"
              ]
            },
            "name": {
              "title": "Standard Name",
              "type": "string",
              "maxLength": 1024,
              "description": "Type ID or standard number/name"
            },
            "issuer": {
              "title": "Issued By",
              "type": "string",
              "net.cnri.repository": {
                "type": {
                  "suggestedVocabulary": [
                    "DTR",
                    "ISO",
                    "W3C",
                    "ITU",
                    "RFC"
                  ]
                }
              }
            },
            "details": {
              "title": "Details",
              "type": "string",
              "format": "textarea",
              "maxLength": 2048
            }
          }
        }
      },
      "provenance": {
        "type": "object",
```

```
        "title": "Provenance",
        "properties": {
          "contributors": {
            "type": "array",
            "format": "table",
            "title": "Contributors of this Record",
            "items": {
              "title": "Contributor",
              "type": "object",
              "required": [
                "identifiedUsing",
                "name"
              ],
              "properties": {
                "identifiedUsing": {
                  "title": "Identified Using",
                  "type": "string",
                  "net.cnri.repository": {
                    "type": {
                      "suggestedVocabulary": [
                        "Handle",
                        "ORCID",
                        "URL",
                        "Text"
                      ]
                    }
                  }
                },
                "name": {
                  "title": "Name",
                  "type": "string",
                  "maxLength": 2048
                },
                "details": {
                  "title": "Details",
                  "type": "string",
                  "format": "textarea",
                  "maxLength": 1024
                }
              }
            }
          },
          "creationDate": {
            "title": "Creation Date",
            "type": "string",
            "format": "datetime",
            "net.cnri.repository": {
              "type": {
                "autoGeneratedField": "creationDate"
```

```
            }
          }
        },
        "lastModificationDate": {
          "title": "Last Modification Date",
          "type": "string",
          "format": "datetime",
          "net.cnri.repository": {
            "type": {
              "autoGeneratedField": "modificationDate"
            }
          }
        }
      }
    },
    "expectedUses": {
      "type": "array",
      "maxItems": 3,
      "format": "table",
      "title": "Expected Uses",
      "items": {
        "type": "string",
        "title": "Use",
        "format": "textarea",
        "maxLength": 4096
      }
    },
    "representationsAndSemantics": {
      "type": "array",
      "format": "table",
      "title": "Representations and Semantics",
      "uniqueItems": true,
      "items": {
        "type": "object",
        "title": "Representation and Semantic Expression",
        "required": [
          "subSchemaRelation"
        ],
        "properties": {
          "expression": {
            "title": "Expression",
            "type": "string",
            "net.cnri.repository": {
              "type": {
                "suggestedVocabulary": [
                  "Format",
                  "Character Set",
                  "Encoding",
                  "Measurement Unit"
```

```
                ]
              }
            }
          },
          "value": {
            "title": "Value",
            "description": "Unicode, UTF-8, Meter, etc.",
            "type": "string",
            "maxLength": 1024
          },
          "subSchemaRelation": {
            "title": "Relations for Sub-Schemas in Properties-Section",
            "description": "Describes, how the properties, the type is derived from, are related to each other. Each of them is assumed to be either mandatory or optional, which is defined as an obligation for each individual property below. With \"denyAdditionalProperties\" one defines an object with given properties below where additional properties are not allowed, and with \"allowAdditionalProperties\" one defines an object with given properties below where additional properties are allowed. With \"requestAll/Any/OneOfProperties\" and \"isNot\" the property obligations below are not reflected (possible contradictions) and all/any/exactly_one of the properties have to be given in the object or \"isNot\" negates the sub-schemas of the properties.",
            "type": "string",
            "net.cnri.repository": {
              "type": {
                "suggestedVocabulary": [
                  "denyAdditionalProperties",
                  "allowAdditionalProperties",
                  "requestAllOfProperties",
                  "requestAnyOfProperties",
                  "requestOneOfProperties",
                  "isNot"
                ],
                "default": "denyAdditionalProperties"
              }
            }
          },
          "restrict": {
            "type": "string",
            "maxLength": 4096,
            "title": "Restrictions",
            "description": "Define type dependent restrictions with special keywords like  uniqueItems, maxItems, minItems or minProperties and maxProperties. Examples: \"uniqueItems\": true, \"maxItems\": 9",
            "net.cnri.repository": {
              "preview": {
                "showInPreview": true,
                "isPrimary": true
              }
```

```
            }
          },
          "default": {
            "type": "string",
            "maxLength": 262144,
            "title": "Default Value",
            "description": "A default value can be provided here. Needs to be a string representation of a JSON object that is compatible with these restrictions. String delimiters need to be escaped",
            "net.cnri.repository": {
              "preview": {
                "showInPreview": true,
                "isPrimary": true
              }
            }
          },
          "details": {
            "title": "Details",
            "type": "string",
            "format": "textarea",
            "maxLength": 2048
          }
        }
      }
    },
    "properties": {
      "type": "array",
      "title": "Properties",
      "description": "Type dependencies used for expressing how this type is built from other types",
      "items": {
        "type": "object",
        "headerTemplate": "{{self.name}}",
        "title": "Property",
        "required": [
          "identifier",
          "name"
        ],
        "properties": {
          "name": {
            "type": "string",
            "description": "Name assigned to dependent type in this context",
            "title": "Name",
            "pattern": "^([!-~])+$",
            "maxLength": 256
          },
          "identifier": {
            "type": "string",
            "title": "TID of Existing Data Type",
```

```
283             "net.cnri.repository": {
284               "type": {
285                 "handleReference": {
286                   "types": [
287                     "PID-InfoType",
288                     "PID-BasicInfoType",
289                     "KernelInformationProfile"
290                   ],
291                   "name": "{{../name}}"
292                 }
293               }
294             }
295           },
296           "representationsAndSemantics": {
297             "type": "array",
298             "format": "table",
299             "title": "Representations and Semantics of Dependent Type",
300             "description": "Restrictions on representations and semantics of the
    dependent type in this context",
301             "uniqueItems": true,
302             "items": {
303               "type": "object",
304               "title": "Representation and Semantic Expression",
305               "required": [
306                 "repeatable",
307                 "obligation"
308               ],
309               "properties": {
310                 "expression": {
311                   "title": "Expression",
312                   "description": "Purpose this property is used for",
313                   "type": "string",
314                   "net.cnri.repository": {
315                     "type": {
316                       "suggestedVocabulary": [
317                         "Format",
318                         "Character Set",
319                         "Encoding",
320                         "Measurement Unit",
321                         "Display Option",
322                         "Search Option"
323                       ],
324                       "default": "Measurement Unit"
325                     }
326                   }
327                 },
328                 "value": {
329                   "title": "Value",
```

```
                    "description": "Used if the type has a constant value on all possible instances for this parameter. Needs to be a string representation of a JSON object that is compatible with the restrictions given by the here referred type",
                    "type": "string",
                    "maxLength": 262144
                  },
                  "obligation": {
                    "type": "string",
                    "minItems": 1,
                    "format": "table",
                    "description": "Optional/Mandatory",
                    "title": "Obligation",
                    "net.cnri.repository": {
                      "type": {
                        "suggestedVocabulary": [
                          "Optional",
                          "Mandatory"
                        ],
                        "default": "Mandatory"
                      }
                    }
                  },
                  "repeatable": {
                    "type": "string",
                    "minItems": 1,
                    "format": "table",
                    "title": "Repeatable",
                    "description": "Cardinality of this property",
                    "net.cnri.repository": {
                      "type": {
                        "suggestedVocabulary": [
                          "Yes",
                          "No"
                        ]
                      }
                    }
                  },
                  "details": {
                    "title": "Details",
                    "type": "string",
                    "format": "textarea",
                    "maxLength": 2048
                  }
                }
              }
            }
          }
        }
      }
```

```
    }
  },
  "javascript": ""
}
```

Listing A.4: Schema für Kernel Information Profile (KIPs)

A.3. Anwendungsbeispiel

Im Anhang sind nur wenige ausgewählte Anfragen und Antworten verfügbar. Alle Anfragen und Antworten von IDORIS sind auch in einer Postman-Collection abrufbar. Diese ist unter `https://api.postman.com/collections/28926386-a6e5c1ee-161d-4a9e-92c2-bb4a5e02b494?access_key=PMAT-01J69Y1DTANRWZ430PF7V9ADBW` aufrufbar.

A.3.1. Erstellen von Nutzern, Lizenzen und Standards

A.3.1.1. Erstellen eines Nutzers

```
POST /api/users HTTP/1.1
Host: localhost:8095
Content-Length: 73

{
  "type": "orcid",
  "orcid": "https://orcid.org/0009-0005-2800-4833"
}
```

Listing A.5: Request zum Erstellen eines Nutzers

```
HTTP/1.1 201 Created
Location: http://localhost:8095/api/users/4:8436039a-b995-4c8c-b6e9-befe8cd30793:0
Content-Type: application/hal+json

{
    "createdAt": "2024-08-22T09:39:27.464169Z",
    "type": "orcid",
    "orcid": "https://orcid.org/0009-0005-2800-4833",
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/users/4:8436039a-b995-4c8c-b6e9-befe8cd30793:0"
        },
        "user": {
            "href": "http://localhost:8095/api/users/4:8436039a-b995-4c8c-b6e9-befe8cd30793:0"
        }
    }
}
```

Listing A.6: Response auf Erstellen eines Nutzers

A.3.1.2. Erstellen von Lizenzen

```
POST /api/licenses HTTP/1.1
Host: localhost:8095
Content-Type: application/json
Content-Length: 86

{
    "name": "CC-BY 4.0",
    "url": "https://creativecommons.org/licenses/by/4.0/"
}
```

Listing A.7: Request zum Erstellen einer Lizenz

```
HTTP/1.1 201 Created
Location: http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-befe8cd30793:1
Content-Type: application/hal+json

{
    "name" : "CC-BY 4.0",
    "url" : "https://creativecommons.org/licenses/by/4.0/",
    "_links" : {
        "self" : {
            "href" : "http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-
befe8cd30793:1"
        },
        "license" : {
            "href" : "http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-
befe8cd30793:1"
        }
    }
}
```

Listing A.8: Response auf Erstellen einer Lizenz

```
POST /api/licenses HTTP/1.1
Host: localhost:8095
Content-Length: 84

{
    "name": "Apache 2.0",
    "url": "https://spdx.org/licenses/Apache-2.0.html"
}
```

Listing A.9: Request zum Erstellen einer anderen Lizenz

```
HTTP/1.1 201 Created
Location: http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-befe8cd30793:4
```

```
Content-Type: application/hal+json

{
    "name": "Apache 2.0",
    "url": "https://spdx.org/licenses/Apache-2.0.html",
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-befe8cd30793:4"
        },
        "license": {
            "href": "http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-befe8cd30793:4"
        }
    }
}
```

Listing A.10: Response auf Erstellen einer anderen Lizenz

A.3.1.3. Erstellen von Standards

```
POST /api/standards HTTP/1.1
Host: localhost:8095
Content-Type: application/json
Content-Length: 127

{
    "name": "27729",
    "issuer": "ISO",
    "details": "see https://www.isni.org/ or for resolution https://isni.oclc.nl"
}
```

Listing A.11: Request zum Erstellen eines Standards

```
HTTP/1.1 201 Created
Location: http://localhost:8095/api/standards/4:8436039a-b995-4c8c-b6e9-befe8cd30793:2
Content-Type: application/hal+json

{
    "name": "27729",
    "issuer": "ISO",
    "details": "see https://www.isni.org/ or for resolution https://isni.oclc.nl",
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/standards/4:8436039a-b995-4c8c-b6e9-befe8cd30793:2"
        },
```

```
        "standard": {
            "href": "http://localhost:8095/api/standards/4:8436039a-b995-4c8c-b6e9-befe8cd30793:2"
        }
    }
}
```

Listing A.12: Response auf Erstellen eines Standards

```
POST /api/standards HTTP/1.1
Host: localhost:8095
Content-Length: 121

{
    "name": "ORCiD",
    "issuer": "Other",
    "details": "see https://orcid.org/ or for API https://orcid.org/api/"
}
```

Listing A.13: Request zum Erstellen eines anderen Standards

```
HTTP/1.1 201 Created
Location: http://localhost:8095/api/standards/4:8436039a-b995-4c8c-b6e9-befe8cd30793:3
Content-Type: application/hal+json

{
    "name": "ORCiD",
    "issuer": "Other",
    "details": "see https://orcid.org/ or for API https://orcid.org/api/",
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/standards/4:8436039a-b995-4c8c-b6e9-befe8cd30793:3"
        },
        "standard": {
            "href": "http://localhost:8095/api/standards/4:8436039a-b995-4c8c-b6e9-befe8cd30793:3"
        }
    }
}
```

Listing A.14: Response auf Erstellen eines anderen Standards

A.3.2. Erstellen von BasicDataTypes

A.3.2.1. Fehlerhafte Erstellung eines BasicDataTypes für eine URL

```
POST /api/dataTypes HTTP/1.1
Host: localhost:8095
Content-Type: application/json
Content-Length: 293

{
"type": "BasicDataType",
"name": "URL",
"contributors": [
"http://localhost:8095/api/users/4:02578f7b-d43a-4138-9f59-a1eaa1940877:35"
],
"license": "http://localhost:8095/api/licenses/4:02578f7b-d43a-4138-9f59-a1eaa1940877:38",
"primitiveDataType": "string"
}
```

Listing A.15: Request zum Erstellen eines fehlerhaften BasicDataTypes für eine URL

```
HTTP/1.1 406 Not Acceptable
Content-Type: application/hal+json
[
    {
    "codes": [
        "SyntaxValidator.BasicDataType",
        "SyntaxValidator"
    ],
    "arguments": [
        {
        "WARNING": [
            {
            "message": "For better human readability and understanding, you SHOULD provide a description for the data type.",
            "element": {
                "pid": null,
                "version": null,
                "createdAt": null,
                "lastModifiedAt": null,
                "contributors": [
                    null
                ],
                "license": null,
                "standards": null,
                "type": "BasicDataType",
                "name": "URL",
                "description": null,
                "expectedUses": null,
                "defaultValue": null,
                "inheritsFrom": null,
                "primitiveDataType": "string",
```

```
                "category": "Format",
                "unitName": null,
                "unitSymbol": null,
                "definedBy": null,
                "standard_uncertainty": null,
                "restrictions": null,
                "regex": null,
                "regexFlavour": "ecma-262-RegExp",
                "valueEnum": null
                },
            "severity": "WARNING"
            }, {
            "message": "For better human readability and understanding, you SHOULD provide a list of expected uses for the data type.",
            "element": {
                "pid": null,
                "version": null,
                "createdAt": null,
                "lastModifiedAt": null,
                "contributors": [
                    null
                ],
                "license": null,
                "standards": null,
                "type": "BasicDataType",
                "name": "URL",
                "description": null,
                "expectedUses": null,
                "defaultValue": null,
                "inheritsFrom": null,
                "primitiveDataType": "string",
                "category": "Format",
                "unitName": null,
                "unitSymbol": null,
                "definedBy": null,
                "standard_uncertainty": null,
                "restrictions": null,
                "regex": null,
                "regexFlavour": "ecma-262-RegExp",
                "valueEnum": null
                },
            "severity": "WARNING"
            }
        ]
    }, {
        "ERROR": [
            {
            "message": "A format data type MUST have a regex.",
            "element": {
```

```
                "pid": null,
                "version": null,
                "createdAt": null,
                "lastModifiedAt": null,
                "contributors": [
                    null
                ],
                "license": null,
                "standards": null,
                "type": "BasicDataType",
                "name": "URL",
                "description": null,
                "expectedUses": null,
                "defaultValue": null,
                "inheritsFrom": null,
                "primitiveDataType": "string",
                "category": "Format",
                "unitName": null,
                "unitSymbol": null,
                "definedBy": null,
                "standard_uncertainty": null,
                "restrictions": null,
                "regex": null,
                "regexFlavour": "ecma-262-RegExp",
                "valueEnum": null
                },
            "severity": "ERROR"
            }
        ]
    }],
    "defaultMessage": "Validation with SyntaxValidator failed.",
    "objectName": "BasicDataType",
    "code": "SyntaxValidator"
    }
]
```


Listing A.16: Response auf fehlerhafte Erstellung eines BasicDataTypes für eine URL

A.3.2.2. Erstellen eines BasicDataTypes für eine HTTP-URL

```
POST /dataTypes HTTP/1.1
Host: http://localhost:8095/api
Content-Length: 428

{
"type": "BasicDataType",
"name": "HTTP-URL",
```

```
"description": "A HTTP URL refers to a ressource made available via HTTP or HTTPS.",
"expectedUses":[
"Referring to resources",
"Hyperlinks to webpages"
],
"contributors": [
"http://localhost:8095/api/users/4:8436039a-b995-4c8c-b6e9-befe8cd30793:0"
],
"regex":"^http(s)?:(\\/\\/([^/?#]*))?([^?#]*)(\\?([^#]*))?(#(.*))?$",
"license": "http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-befe8cd30793:1",
"primitiveDataType": "string"
}
```

Listing A.17: Request zum Erstellen eines BasicDataTypes für eine HTTP-URL

```
HTTP/1.1 201 Created
Content-Type: application/hal+json
Location: http://localhost:8095/api/basicDataTypes/876bf2c7-1746-4197-9a81-5fffee501b43
Etag: "0"

{
    "pid": "876bf2c7-1746-4197-9a81-5fffee501b43",
    "createdAt": "2024-08-22T10:31:50.965644Z",
    "lastModifiedAt": "2024-08-22T10:31:50.965644Z",
    "standards": null,
    "type": "BasicDataType",
    "name": "HTTP-URL",
    "description": "A HTTP URL refers to a ressource made available via HTTP or HTTPS.",
    "expectedUses": [
        "Referring to resources",
        "Hyperlinks to webpages"
    ],
    "defaultValue": null,
    "primitiveDataType": "string",
    "category": "Format",
    "unitName": null,
    "unitSymbol": null,
    "definedBy": null,
    "standard_uncertainty": null,
    "restrictions": null,
    "regex": "^http(s)?:(\\/\\/([^/?#]*))?([^?#]*)(\\?([^#]*))?(#(.*))?$",
    "regexFlavour": "ecma-262-RegExp",
    "valueEnum": null,
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/basicDataTypes/876bf2c7-1746-4197-9a81-5
    fffee501b43"
        },
```

```
33          "basicDataType": {
34              "href": "http://localhost:8095/api/basicDataTypes/876bf2c7-1746-4197-9a81-5
    fffee501b43"
35          },
36          "inheritsFrom": {
37              "href": "http://localhost:8095/api/basicDataTypes/876bf2c7-1746-4197-9a81-5
    fffee501b43/inheritsFrom"
38          },
39          "operations": {
40              "href": "http://localhost:8095/api/operations/search/getOperationsForDataType?
    pid=876bf2c7-1746-4197-9a81-5fffee501b43"
41          },
42          "license": {
43              "href": "http://localhost:8095/api/basicDataTypes/876bf2c7-1746-4197-9a81-5
    fffee501b43/license"
44          },
45          "contributors": {
46              "href": "http://localhost:8095/api/basicDataTypes/876bf2c7-1746-4197-9a81-5
    fffee501b43/contributors"
47          }
48      }
49  }
```

Listing A.18: Response auf Erstellen eines BasicDataTypes für eine HTTP-URL

Erstellen eines BasicDataTypes für ORCiD-URLs mit Vererbung

```
1   POST /dataTypes HTTP/1.1
2   Host: http://localhost:8095/api
3   Content-Length: 936
4
5   {
6       "type": "BasicDataType",
7       "name": "ORCiD-URL",
8       "description": "This data type defines a link to an ORCiD.",
9       "standards": [
10          {
11              "standard": "http://localhost:8095/api/standards/4:8436039a-b995-4c8c-b6e9-
    befe8cd30793:2",
12              "natureOfApplicability": "Extends"
13          },
14          {
15              "standard": "http://localhost:8095/api/standards/4:8436039a-b995-4c8c-b6e9-
    befe8cd30793:3",
16              "natureOfApplicability": "constrains"
17          }
18      ],
```

```
    "contributors": [
        "http://localhost:8095/api/users/4:8436039a-b995-4c8c-b6e9-befe8cd30793:0"
    ],
    "license": "http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-befe8cd30793
    :1",
    "expectedUses": [
        "Identification of a person",
        "Linking to a persons ORCiD profile"
    ],
    "inheritsFrom": "http://localhost:8095/api/basicDataTypes/876bf2c7-1746-4197-9a81-5
    fffee501b43",
    "primitiveDataType": "string",
    "regex": "https://orcid.org/[0-9]{4}-[0-9]{4}-[0-9]{4}-[0-9]{3}[0-9X]",
    "regexFlavour": "ecma-262-RegExp"
}
```

Listing A.19: Request zum Erstellen eines BasicDataTypes für eine ORCiD-URL

```
HTTP/1.1 201 Created
Content-Type: application/hal+json
Location: http://localhost:8095/api/basicDataTypes/b5a64c2f-68eb-4c32-bff6-afd39394a85a
Etag: "0"

{
    "pid": "b5a64c2f-68eb-4c32-bff6-afd39394a85a",
    "createdAt": "2024-08-22T10:57:51.705355Z",
    "lastModifiedAt": "2024-08-22T10:57:51.705355Z",
    "standards": [
        {
            "natureOfApplicability": "constrains",
            "details": null,
            "_links": {
                "standard": {
                    "href": "http://localhost:8095/api/standards/4:8436039a-b995-4c8c-b6e9
    -befe8cd30793:3"
                }
            }
        },
        {
            "natureOfApplicability": "Extends",
            "details": null,
            "_links": {
                "standard": {
                    "href": "http://localhost:8095/api/standards/4:8436039a-b995-4c8c-b6e9
    -befe8cd30793:2"
                }
            }
        }
```

```
    ],
    "type": "BasicDataType",
    "name": "ORCiD-URL",
    "description": "This data type defines a link to an ORCiD.",
    "expectedUses": [
        "Identification of a person",
        "Linking to a persons ORCiD profile"
    ],
    "defaultValue": null,
    "primitiveDataType": "string",
    "category": "Format",
    "unitName": null,
    "unitSymbol": null,
    "definedBy": null,
    "standard_uncertainty": null,
    "restrictions": null,
    "regex": "https://orcid.org/[0-9]{4}-[0-9]{4}-[0-9]{4}-[0-9]{3}[0-9X]",
    "regexFlavour": "ecma-262-RegExp",
    "valueEnum": null,
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/basicDataTypes/b5a64c2f-68eb-4c32-bff6-
    afd39394a85a"
        },
        "basicDataType": {
            "href": "http://localhost:8095/api/basicDataTypes/b5a64c2f-68eb-4c32-bff6-
    afd39394a85a"
        },
        "license": {
            "href": "http://localhost:8095/api/basicDataTypes/b5a64c2f-68eb-4c32-bff6-
    afd39394a85a/license"
        },
        "operations": {
            "href": "http://localhost:8095/api/operations/search/getOperationsForDataType?
    pid=b5a64c2f-68eb-4c32-bff6-afd39394a85a"
        },
        "inheritsFrom": {
            "href": "http://localhost:8095/api/basicDataTypes/b5a64c2f-68eb-4c32-bff6-
    afd39394a85a/inheritsFrom"
        },
        "contributors": {
            "href": "http://localhost:8095/api/basicDataTypes/b5a64c2f-68eb-4c32-bff6-
    afd39394a85a/contributors"
        }
    }
}
```


Listing A.20: Response auf Erstellen eines BasicDataTypes für eine ORCiD-URL

A.3.2.3. Erstellen eines BasicDataTypes für die HTTP-Methoden mit Enumeration aller möglichen Werte

```
POST /dataTypes HTTP/1.1
Host: http://localhost:8095/api
Content-Length: 500

{
    "type": "BasicDataType",
    "name": "HTTPMethod",
    "description": "Definition of the HTTP-Verbs",
    "expectedUses": [
        "Access to an HTTP resource",
        "Specification of an HTTP API"
    ],
    "contributors": [
        "http://localhost:8095/api/users/4:8436039a-b995-4c8c-b6e9-befe8cd30793:0"
    ],
    "category":"Enumeration",
    "license": "http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-befe8cd30793
    :1",
    "valueEnum": [
        "GET",
        "POST",
        "PUT",
        "PATCH",
        "DELETE",
        "HEAD",
        "OPTIONS"
    ],
    "primitiveDataType": "string"
}
```

Listing A.21: Request zum Erstellen eines BasicDataTypes für die HTTP-Methoden

```
HTTP/1.1 201 Created
Content-Type: application/hal+json
Location: http://localhost:8095/api/basicDataTypes/3699f1af-50cf-4f5e-ac4f-611251dcc313
Etag: "0"

{
    "pid": "3699f1af-50cf-4f5e-ac4f-611251dcc313",
    "createdAt": "2024-08-22T11:27:33.187030Z",
    "lastModifiedAt": "2024-08-22T11:27:33.187030Z",
    "standards": null,
    "type": "BasicDataType",
    "name": "HTTPMethod",
    "description": "Definition of the HTTP-Verbs",
    "expectedUses": [
```

```
        "Access to an HTTP resource",
        "Specification of an HTTP API"
    ],
    "defaultValue": null,
    "primitiveDataType": "string",
    "category": "Enumeration",
    "unitName": null,
    "unitSymbol": null,
    "definedBy": null,
    "standard_uncertainty": null,
    "restrictions": null,
    "regex": null,
    "regexFlavour": "ecma-262-RegExp",
    "valueEnum": [
        "HEAD",
        "DELETE",
        "POST",
        "GET",
        "OPTIONS",
        "PUT",
        "PATCH"
    ],
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/basicDataTypes/3699f1af-50cf-4f5e-ac4f-611251dcc313"
        },
        "basicDataType": {
            "href": "http://localhost:8095/api/basicDataTypes/3699f1af-50cf-4f5e-ac4f-611251dcc313"
        },
        "license": {
            "href": "http://localhost:8095/api/basicDataTypes/3699f1af-50cf-4f5e-ac4f-611251dcc313/license"
        },
        "operations": {
            "href": "http://localhost:8095/api/operations/search/getOperationsForDataType?pid=3699f1af-50cf-4f5e-ac4f-611251dcc313"
        },
        "inheritsFrom": {
            "href": "http://localhost:8095/api/basicDataTypes/3699f1af-50cf-4f5e-ac4f-611251dcc313/inheritsFrom"
        },
        "contributors": {
            "href": "http://localhost:8095/api/basicDataTypes/3699f1af-50cf-4f5e-ac4f-611251dcc313/contributors"
        }
    }
}
```

Listing A.22: Response auf Erstellen eines BasicDataTypes für die HTTP-Methoden

A.3.3. Erstellen von Attributen

```
POST /attributes HTTP/1.1
Host: http://localhost:8095/api
Content-Length: 248

{
    "dataType": "http://localhost:8095/api/basicDataTypes/b5a64c2f-68eb-4c32-bff6-afd39394a85a",
    "name": "ORCiD-URL",
    "description": "An URL possibly referring to an ORCiD profile",
    "obligation": "Mandatory",
    "repeatable": false
}
```

Listing A.23: Request zum Erstellen eines Attributs für eine ORCiD-URL

```
HTTP/1.1 201 Created
Content-Type: application/hal+json
Location: http://localhost:8095/api/attributes/398ef1b0-d3bb-4665-ad90-ee9be990ed36

{
    "name": "ORCiD-URL",
    "description": "An URL possibly referring to an ORCiD profile",
    "repeatable": false,
    "obligation": "Mandatory",
    "value": null,
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/attributes/398ef1b0-d3bb-4665-ad90-ee9be990ed36"
        },
        "attribute": {
            "href": "http://localhost:8095/api/attributes/398ef1b0-d3bb-4665-ad90-ee9be990ed36"
        },
        "override": {
            "href": "http://localhost:8095/api/attributes/398ef1b0-d3bb-4665-ad90-ee9be990ed36/override"
        },
        "dataType": {
            "href": "http://localhost:8095/api/attributes/398ef1b0-d3bb-4665-ad90-ee9be990ed36/dataType"
```

```
        }
    }
}
```

Listing A.24: Response auf Erstellen eines Attributs für eine ORCiD-URL

A.3.4. Erstellen von Typprofilen

```
POST /api/dataTypes HTTP/1.1
Host: localhost:8095
Content-Type: application/json
Content-Length: 709

{
    "type": "TypeProfile",
    "name": "Key-Value pair",
    "description": "This TypeProfile represents a single key-value pair. To create a set of KV-Pairs, just repeat this TypeProfile.",
    "expectedUses": [
        "Representing elements in map data structures",
        "Single HTTP-Header"
    ],
    "contributors": [
        "http://localhost:8095/api/users/4:8436039a-b995-4c8c-b6e9-befe8cd30793:0"
    ],
    "license": "http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-befe8cd30793:1",
    "attributes": [
        "http://localhost:8095/api/attributes/53c7f3a4-1b2c-4b11-90b2-8114aaa66c11",
        "http://localhost:8095/api/attributes/119cefd2-4862-4245-8243-187789089634"
    ]
}
```

Listing A.25: Request zum Erstellen eines Typprofils für ein Key-Value-Paar

```
HTTP/1.1 201 Created
Content-Type: application/hal+json
Location: http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e
Etag: "0"

{
    "pid": "dff4084f-67f9-4229-96df-2fab738cf96e",
    "createdAt": "2024-08-26T09:23:58.054886Z",
    "lastModifiedAt": "2024-08-26T09:23:58.054886Z",
    "standards": null,
    "type": "TypeProfile",
```

```
    "name": "Key-Value pair",
    "description": "This TypeProfile represents a single key-value pair. To create a set of KV-Pairs, just repeat this TypeProfile.",
    "expectedUses": [
        "Representing elements in map data structures",
        "Single HTTP-Header"
    ],
    "defaultValue": null,
    "subSchemaRelation": "allowAdditionalProperties",
    "abstract": false,
    "embeddable": true,
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e"
        },
        "typeProfile": {
            "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e"
        },
        "validate": {
            "href": "/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/validate"
        },
        "inheritedAttributes": {
            "href": "/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/inheritedAttributes"
        },
        "inheritanceTree": {
            "href": "/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/inheritanceTree"
        },
        "operations": {
            "href": "http://localhost:8095/api/operations/search/getOperationsForDataType?pid=dff4084f-67f9-4229-96df-2fab738cf96e"
        },
        "contributors": {
            "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/contributors"
        },
        "attributes": {
            "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/attributes"
        },
        "license": {
            "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/license"
        },
        "inheritsFrom": {
```

```
            "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/inheritsFrom"
        }
    }
}
```

Listing A.26: Response auf Erstellen eines Typprofils für ein Key-Value-Paar

```
{
    "pid": "29c7d6fb-21df-469d-b75a-dae2811b008d",
    "createdAt": "2024-08-26T09:28:18.667507Z",
    "lastModifiedAt": "2024-08-26T09:28:18.667507Z",
    "standards": null,
    "type": "TypeProfile",
    "name": "Useless",
    "description": "Just for testing of multi-inheritance",
    "expectedUses": [
        "DO NOT USE!",
        "Just for testing purposes"
    ],
    "defaultValue": null,
    "subSchemaRelation": "allowAdditionalProperties",
    "abstract": false,
    "embeddable": true,
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/typeProfiles/29c7d6fb-21df-469d-b75a-dae2811b008d"
        },
        "typeProfile": {
            "href": "http://localhost:8095/api/typeProfiles/29c7d6fb-21df-469d-b75a-dae2811b008d"
        },
        "validate": {
            "href": "/api/typeProfiles/29c7d6fb-21df-469d-b75a-dae2811b008d/validate"
        },
        "inheritedAttributes": {
            "href": "/api/typeProfiles/29c7d6fb-21df-469d-b75a-dae2811b008d/inheritedAttributes"
        },
        "inheritanceTree": {
            "href": "/api/typeProfiles/29c7d6fb-21df-469d-b75a-dae2811b008d/inheritanceTree"
        },
        "operations": {
            "href": "http://localhost:8095/api/operations/search/getOperationsForDataType?pid=29c7d6fb-21df-469d-b75a-dae2811b008d"
        },
```

```
        "contributors": {
            "href": "http://localhost:8095/api/typeProfiles/29c7d6fb-21df-469d-b75a-dae2811b008d/contributors"
        },
        "attributes": {
            "href": "http://localhost:8095/api/typeProfiles/29c7d6fb-21df-469d-b75a-dae2811b008d/attributes"
        },
        "license": {
            "href": "http://localhost:8095/api/typeProfiles/29c7d6fb-21df-469d-b75a-dae2811b008d/license"
        },
        "inheritsFrom": {
            "href": "http://localhost:8095/api/typeProfiles/29c7d6fb-21df-469d-b75a-dae2811b008d/inheritsFrom"
        }
    }
}
```

Listing A.27: Response auf die Erstellung des Useless Typprofil zu Demonstrationszwecken

```
POST /dataTypes HTTP/1.1
Host: http://localhost:8095/api
Content-Length: 504

{
    "type": "TypeProfile",
    "name": "HTTP Header",
    "description": "A single HTTP Header",
    "expectedUses": [
        "Headers in HTTP requests",
        "Headers in HTTP responses"
    ],
    "contributors": [
        "http://localhost:8095/api/users/4:8436039a-b995-4c8c-b6e9-befe8cd30793:0"
    ],
    "license": "http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-befe8cd30793:1",
    "inheritsFrom": [
        "http://localhost:8095/api/typeProfiles/29c7d6fb-21df-469d-b75a-dae2811b008d",
        "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e"
    ],
    "attributes": []
}
```

Listing A.28: Request zum Erstellen eines Typprofils für HTTP-Header

```
HTTP/1.1 201 Created
```

```
Content-Type: application/hal+json
Location: http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85
Etag: "0"

{
    "pid": "3a2bfb7e-be49-4b92-8476-b6d0efa4ab85",
    "createdAt": "2024-08-26T09:30:14.459706Z",
    "lastModifiedAt": "2024-08-26T09:30:14.459706Z",
    "standards": null,
    "type": "TypeProfile",
    "name": "HTTP Header",
    "description": "A single HTTP Header",
    "expectedUses": [
        "Headers in HTTP requests",
        "Headers in HTTP responses"
    ],
    "defaultValue": null,
    "subSchemaRelation": "allowAdditionalProperties",
    "abstract": false,
    "embeddable": true,
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-
    b6d0efa4ab85"
        },
        "typeProfile": {
            "href": "http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-
    b6d0efa4ab85"
        },
        "validate": {
            "href": "/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85/validate"
        },
        "inheritedAttributes": {
            "href": "/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85/
    inheritedAttributes"
        },
        "inheritanceTree": {
            "href": "/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85/
    inheritanceTree"
        },
        "operations": {
            "href": "http://localhost:8095/api/operations/search/getOperationsForDataType?
    pid=3a2bfb7e-be49-4b92-8476-b6d0efa4ab85"
        },
        "contributors": {
            "href": "http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-
    b6d0efa4ab85/contributors"
        },
        "attributes": {
```

```
            "href": "http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85/attributes"
        },
        "license": {
            "href": "http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85/license"
        },
        "inheritsFrom": {
            "href": "http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85/inheritsFrom"
        }
    }
}
```

Listing A.29: Response auf Erstellen eines Typprofils für HTTP-Header

```
POST /dataTypes HTTP/1.1
Host: http://localhost:8095/api
Content-Length: 630

{
    "type": "TypeProfile",
    "name": "ORCiD profile",
    "description": "An excerpt of the information provided in an ORCiD profile. DO NOT USE IN PRODUCTION!",
    "expectedUses": [
        "Identification of persons",
        "Demo for Bachelor's thesis"
    ],
    "contributors": [
        "http://localhost:8095/api/users/4:8436039a-b995-4c8c-b6e9-befe8cd30793:0"
    ],
    "license": "http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-befe8cd30793:1",
    "attributes": [
        "http://localhost:8095/api/attributes/398ef1b0-d3bb-4665-ad90-ee9be990ed36",
        "http://localhost:8095/api/attributes/24d3e0ab-c451-4b78-8403-50a75ac84aba",
        "http://localhost:8095/api/attributes/acc08442-8764-4c2e-b160-045d0589cf7a"
    ]
}
```

Listing A.30: Request zum Erstellen eines Typprofils für einen Ausschnitt eines ORCiD-Profils

```
HTTP/1.1 201 Created
Content-Type: application/hal+json
Location: http://localhost:8095/api/typeProfiles/29bb291c-59e4-4710-bfea-43dc9e9fb792
Etag: "0"

```

```
{
    "pid": "29bb291c-59e4-4710-bfea-43dc9e9fb792",
    "createdAt": "2024-08-26T09:33:13.249607Z",
    "lastModifiedAt": "2024-08-26T09:33:13.249607Z",
    "standards": null,
    "type": "TypeProfile",
    "name": "ORCiD profile",
    "description": "An excerpt of the information provided in an ORCiD profile. DO NOT USE IN PRODUCTION!",
    "expectedUses": [
        "Identification of persons",
        "Demo for Bachelor‘s thesis"
    ],
    "defaultValue": null,
    "subSchemaRelation": "allowAdditionalProperties",
    "abstract": false,
    "embeddable": true,
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/typeProfiles/29bb291c-59e4-4710-bfea-43dc9e9fb792"
        },
        "typeProfile": {
            "href": "http://localhost:8095/api/typeProfiles/29bb291c-59e4-4710-bfea-43dc9e9fb792"
        },
        "validate": {
            "href": "/api/typeProfiles/29bb291c-59e4-4710-bfea-43dc9e9fb792/validate"
        },
        "inheritedAttributes": {
            "href": "/api/typeProfiles/29bb291c-59e4-4710-bfea-43dc9e9fb792/inheritedAttributes"
        },
        "inheritanceTree": {
            "href": "/api/typeProfiles/29bb291c-59e4-4710-bfea-43dc9e9fb792/inheritanceTree"
        },
        "operations": {
            "href": "http://localhost:8095/api/operations/search/getOperationsForDataType?pid=29bb291c-59e4-4710-bfea-43dc9e9fb792"
        },
        "contributors": {
            "href": "http://localhost:8095/api/typeProfiles/29bb291c-59e4-4710-bfea-43dc9e9fb792/contributors"
        },
        "attributes": {
            "href": "http://localhost:8095/api/typeProfiles/29bb291c-59e4-4710-bfea-43dc9e9fb792/attributes"
        },
```

```
47          "license": {
48              "href": "http://localhost:8095/api/typeProfiles/29bb291c-59e4-4710-bfea-43
    dc9e9fb792/license"
49          },
50          "inheritsFrom": {
51              "href": "http://localhost:8095/api/typeProfiles/29bb291c-59e4-4710-bfea-43
    dc9e9fb792/inheritsFrom"
52          }
53      }
54  }
```

Listing A.31: Response auf Erstellen eines Typprofils für einen Ausschnitt eines ORCiD-Profils

A.3.5. Erstellen von OperationTypeProfiles

```
1   POST /operationTypeProfiles HTTP/1.1
2   Host: http://localhost:8095/api
3   Content-Length: 833
4
5   {
6       "name": "HTTP Request",
7       "description": "This pattern is used to make a HTTP request to a server.",
8       "license": "http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-befe8cd30793
    :1",
9       "contributors": [
10          "http://localhost:8095/api/users/4:8436039a-b995-4c8c-b6e9-befe8cd30793:0"
11      ],
12      "attributes": [
13          "http://localhost:8095/api/attributes/d0d63d94-1176-4b00-8ba8-9ea243ce17be",
14          "http://localhost:8095/api/attributes/5fcf6db6-261e-4b61-a9c8-05e5f159dc81",
15          "http://localhost:8095/api/attributes/e09cb2f3-0d84-40d7-b4dd-6164a8c2deac",
16          "http://localhost:8095/api/attributes/4bcbd5b5-d87d-46d8-a4d5-c46a23b02cca",
17          "http://localhost:8095/api/attributes/2c0cd79d-783e-4c0c-8682-32526bbe09a8"
18      ],
19      "outputs": [
20          "http://localhost:8095/api/attributes/34e2a7ea-f9ac-4125-859c-99e67aa1136a",
21          "http://localhost:8095/api/attributes/41cb04ae-22d1-4cbe-b871-1945f24de19e"
22      ]
23  }
```

Listing A.32: Request zum Erstellen eines OperationTypeProfiles für HTTP

```
1   HTTP/1.1 201 Created
2   Content-Type: application/hal+json
3   Location: http://localhost:8095/api/operationTypeProfiles/02661f03-31e4-4e03-97df-8155482
    e166b
```

```
Etag: "0"

{
    "pid": "02661f03-31e4-4e03-97df-8155482e166b",
    "createdAt": "2024-08-26T09:45:45.796677Z",
    "lastModifiedAt": "2024-08-26T09:45:45.796677Z",
    "standards": null,
    "name": "HTTP Request",
    "description": "This pattern is used to make a HTTP request to a server.",
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/operationTypeProfiles/02661f03-31e4-4e03-97df-8155482e166b"
        },
        "operationTypeProfile": {
            "href": "http://localhost:8095/api/operationTypeProfiles/02661f03-31e4-4e03-97df-8155482e166b"
        },
        "adapters": {
            "href": "http://localhost:8095/api/operationTypeProfiles/02661f03-31e4-4e03-97df-8155482e166b/adapters"
        },
        "license": {
            "href": "http://localhost:8095/api/operationTypeProfiles/02661f03-31e4-4e03-97df-8155482e166b/license"
        },
        "attributes": {
            "href": "http://localhost:8095/api/operationTypeProfiles/02661f03-31e4-4e03-97df-8155482e166b/attributes"
        },
        "contributors": {
            "href": "http://localhost:8095/api/operationTypeProfiles/02661f03-31e4-4e03-97df-8155482e166b/contributors"
        },
        "inheritsFrom": {
            "href": "http://localhost:8095/api/operationTypeProfiles/02661f03-31e4-4e03-97df-8155482e166b/inheritsFrom"
        },
        "outputs": {
            "href": "http://localhost:8095/api/operationTypeProfiles/02661f03-31e4-4e03-97df-8155482e166b/outputs"
        }
    }
}
```

Listing A.33: Response auf Erstellen eines OperationTypeProfiles für eine HTTP-Request

```
POST /api/operationTypeProfiles HTTP/1.1
```

```
Host: localhost:8095
Content-Length: 558

{
    "name": "Regex",
    "description": "This pattern is used to verify using Regex.",
    "contributors": [
        "http://localhost:8095/api/users/4:8436039a-b995-4c8c-b6e9-befe8cd30793:0"
    ],
    "license": "http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-befe8cd30793:4",
    "attributes": [
        "http://localhost:8095/api/attributes/38719343-1ef9-42f0-af47-b13209c5e057",
        "http://localhost:8095/api/attributes/fe1c5c71-f7a0-41df-94ab-5167f37d484f",
        "http://localhost:8095/api/attributes/4960a4a7-0250-4f57-a693-78242e00d141"
    ],
    "outputs": [
        "http://localhost:8095/api/attributes/64331e45-4479-4c32-aefb-e270c39b7378"
    ]
}
```

Listing A.34: Request zum Erstellen eines OperationTypeProfiles für Regex

```
HTTP/1.1 201 Created
Content-Type: application/hal+json
Location: http://localhost:8095/api/operationTypeProfiles/41b9ab30-b892-4f9b-8504-4344399530ea
Etag: "0"

{
    "pid": "41b9ab30-b892-4f9b-8504-4344399530ea",
    "createdAt": "2024-08-26T09:47:11.778236Z",
    "lastModifiedAt": "2024-08-26T09:47:11.778236Z",
    "standards": null,
    "name": "Regex",
    "description": "This pattern is used to verify using Regex.",
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/operationTypeProfiles/41b9ab30-b892-4f9b-8504-4344399530ea"
        },
        "operationTypeProfile": {
            "href": "http://localhost:8095/api/operationTypeProfiles/41b9ab30-b892-4f9b-8504-4344399530ea"
        },
        "adapters": {
            "href": "http://localhost:8095/api/operationTypeProfiles/41b9ab30-b892-4f9b-8504-4344399530ea/adapters"
```

```
        },
        "license": {
            "href": "http://localhost:8095/api/operationTypeProfiles/41b9ab30-b892-4f9b-8504-4344399530ea/license"
        },
        "attributes": {
            "href": "http://localhost:8095/api/operationTypeProfiles/41b9ab30-b892-4f9b-8504-4344399530ea/attributes"
        },
        "contributors": {
            "href": "http://localhost:8095/api/operationTypeProfiles/41b9ab30-b892-4f9b-8504-4344399530ea/contributors"
        },
        "inheritsFrom": {
            "href": "http://localhost:8095/api/operationTypeProfiles/41b9ab30-b892-4f9b-8504-4344399530ea/inheritsFrom"
        },
        "outputs": {
            "href": "http://localhost:8095/api/operationTypeProfiles/41b9ab30-b892-4f9b-8504-4344399530ea/outputs"
        }
    }
}
```

Listing A.35: Response auf Erstellen eines OperationTypeProfiles für Regex

A.3.6. Erstellen von Operationen

```
POST /api/operations HTTP/1.1
Host: localhost:8095
Content-Length: 1848

{
    "name": "Extract ORCiD number out of text",
    "description": "This operation extracts an ORCiD number out of a URL and returns it.",
    "license": "http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-befe8cd30793:1",
    "contributors": [
        "http://localhost:8095/api/users/4:8436039a-b995-4c8c-b6e9-befe8cd30793:0"
    ],
    "executableOn": "http://localhost:8095/api/attributes/398ef1b0-d3bb-4665-ad90-ee9be990ed36",
    "returns": [
        "http://localhost:8095/api/attributes/73551d9b-f18f-4a1c-8724-17fddbedb738"
    ],
    "execution": [
        {
```

```
            "name": "Execute Regex",
            "executionOrderIndex": 0,
            "operationTypeProfile": "http://localhost:8095/api/operationTypeProfiles/41b9ab30-b892-4f9b-8504-4344399530ea",
            "attributes": [
                {
                    "name": "Regex Pattern",
                    "value": "https?://orcid.org/(\\d{4}-\\d{4}-\\d{4}-\\d{3}X?)",
                    "output": "http://localhost:8095/api/attributes/fe1c5c71-f7a0-41df-94ab-5167f37d484f"
                },
                {
                    "name": "Regex flavour",
                    "value": "ECMA262",
                    "output": "http://localhost:8095/api/attributes/4960a4a7-0250-4f57-a693-78242e00d141"
                },
                {
                    "name": "Regex input from orcidURL",
                    "input": "http://localhost:8095/api/attributes/398ef1b0-d3bb-4665-ad90-ee9be990ed36",
                    "output": "http://localhost:8095/api/attributes/38719343-1ef9-42f0-af47-b13209c5e057"
                }
            ],
            "output": [
                {
                    "name": "ORCiD number",
                    "input": "http://localhost:8095/api/attributes/64331e45-4479-4c32-aefb-e270c39b7378",
                    "output": "http://localhost:8095/api/attributes/73551d9b-f18f-4a1c-8724-17fddbedb738"
                }
            ]
        }
    ]
}
```

Listing A.36: Request zum Erstellen einer Operation zur Extraktion der ORCiD-Nummer aus einer ORCiD-URL

```
HTTP/1.1 201 Created
Content-Type: application/hal+json
Location: http://localhost:8095/api/operations/c379e4ca-3184-4dea-99a9-89994ab54690
Etag: "0"

{
    "pid": "c379e4ca-3184-4dea-99a9-89994ab54690",
```

```
    "createdAt": "2024-08-26T09:56:34.648525Z",
    "lastModifiedAt": "2024-08-26T09:56:34.648525Z",
    "standards": null,
    "name": "Extract ORCiD number out of text",
    "description": "This operation extracts an ORCiD number out of a URL and returns it.",
    "execution": [
        {
            "executionOrderIndex": 0,
            "name": "Execute Regex",
            "mode": "sync",
            "steps": null,
            "attributes": [
                {
                    "name": "Regex Pattern",
                    "replaceCharactersInValueWithInput": "${{input}}",
                    "value": "https?://orcid.org/(\\d{4}-\\d{4}-\\d{4}-\\d{3}X?)",
                    "index": null,
                    "_links": {
                        "output": {
                            "href": "http://localhost:8095/api/attributes/fe1c5c71-f7a0-41
df-94ab-5167f37d484f"
                        }
                    }
                },
                {
                    "name": "Regex flavour",
                    "replaceCharactersInValueWithInput": "${{input}}",
                    "value": "ECMA262",
                    "index": null,
                    "_links": {
                        "output": {
                            "href": "http://localhost:8095/api/attributes/4960a4a7-0250-4
f57-a693-78242e00d141"
                        }
                    }
                },
                {
                    "name": "Regex input from orcidURL",
                    "replaceCharactersInValueWithInput": "${{input}}",
                    "value": null,
                    "index": null,
                    "_links": {
                        "output": {
                            "href": "http://localhost:8095/api/attributes/38719343-1ef9-42
f0-af47-b13209c5e057"
                        },
                        "input": {
                            "href": "http://localhost:8095/api/attributes/398ef1b0-d3bb
-4665-ad90-ee9be990ed36"
```

```
                        }
                    }
                }
            ],
            "output": [
                {
                    "name": "ORCiD number",
                    "replaceCharactersInValueWithInput": "${{input}}",
                    "value": null,
                    "index": null,
                    "_links": {
                        "output": {
                            "href": "http://localhost:8095/api/attributes/73551d9b-f18f-4a1c-8724-17fddbedb738"
                        },
                        "input": {
                            "href": "http://localhost:8095/api/attributes/64331e45-4479-4c32-aefb-e270c39b7378"
                        }
                    }
                }
            ],
            "_links": {
                "operationTypeProfile": {
                    "href": "http://localhost:8095/api/operationTypeProfiles/41b9ab30-b892-4f9b-8504-4344399530ea"
                }
            }
        }
    ],
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/operations/c379e4ca-3184-4dea-99a9-89994ab54690"
        },
        "operation": {
            "href": "http://localhost:8095/api/operations/c379e4ca-3184-4dea-99a9-89994ab54690"
        },
        "environment": {
            "href": "http://localhost:8095/api/operations/c379e4ca-3184-4dea-99a9-89994ab54690/environment"
        },
        "executableOn": {
            "href": "http://localhost:8095/api/operations/c379e4ca-3184-4dea-99a9-89994ab54690/executableOn"
        },
        "contributors": {
```

```
            "href": "http://localhost:8095/api/operations/c379e4ca-3184-4dea-99a9-89994ab54690/contributors"
        },
        "license": {
            "href": "http://localhost:8095/api/operations/c379e4ca-3184-4dea-99a9-89994ab54690/license"
        },
        "returns": {
            "href": "http://localhost:8095/api/operations/c379e4ca-3184-4dea-99a9-89994ab54690/returns"
        }
    }
}
```

Listing A.37: Response auf Erstellen einer Operation zur Extraktion der ORCiD-Nummer aus einer ORCiD-URL

```
POST /api/operations HTTP/1.1
Host: localhost:8095
Content-Length: 3842

{
    "name": "Get ORCiD Profile Information via REST API",
    "description": "This Operation retrieves the profile information of an ORCiD ID via the ORCiD API. The ORCiD number sequence is extracted from the ORCiD-Profile URL. An API Key for the ORCiD API is required and to be set in the environment.",
    "license": "http://localhost:8095/api/licenses/4:8436039a-b995-4c8c-b6e9-befe8cd30793:4",
    "contributors": [
        "http://localhost:8095/api/users/4:8436039a-b995-4c8c-b6e9-befe8cd30793:0"
    ],
    "executableOn": "http://localhost:8095/api/attributes/398ef1b0-d3bb-4665-ad90-ee9be990ed36",
    "returns": [
        "http://localhost:8095/api/attributes/41cb04ae-22d1-4cbe-b871-1945f24de19e",
        "http://localhost:8095/api/attributes/bbded7e7-df26-4638-a80a-7bb470d13433"
    ],
    "environment": [
        "http://localhost:8095/api/attributes/abadfa92-9001-4986-94aa-6f0a6ebe73b9"
    ],
    "execution": [
        {
            "name": "Extract ORCiD ID from ORCiD-URL",
            "executionOrderIndex": 0,
            "operation": "http://localhost:8095/api/operations/c379e4ca-3184-4dea-99a9-89994ab54690",
            "attributes": [
                {
```

```
                "name": "Regex input from orcidURL",
                "input": "http://localhost:8095/api/attributes/398ef1b0-d3bb-4665-ad90-ee9be990ed36",
                "output": "http://localhost:8095/api/attributes/398ef1b0-d3bb-4665-ad90-ee9be990ed36"
            }
        ],
        "output": [
            {
                "name": "ORCiD number",
                "input": "http://localhost:8095/api/attributes/73551d9b-f18f-4a1c-8724-17fddbedb738",
                "output": "http://localhost:8095/api/attributes/73551d9b-f18f-4a1c-8724-17fddbedb738"
            }
        ]
    },
    {
        "name": "Get profile data from ORCID.org API",
        "executionOrderIndex": 1,
        "mode": "sync",
        "operationTypeProfile": "http://localhost:8095/api/operationTypeProfiles/02661f03-31e4-4e03-97df-8155482e166b",
        "attributes": [
            {
                "name": "URL",
                "input": "http://localhost:8095/api/attributes/73551d9b-f18f-4a1c-8724-17fddbedb738",
                "value": "https://orcid.org/api/${{input}}",
                "output": "http://localhost:8095/api/attributes/d0d63d94-1176-4b00-8ba8-9ea243ce17be"
            },
            {
                "name": "Method",
                "value": "GET",
                "output": "http://localhost:8095/api/attributes/e09cb2f3-0d84-40d7-b4dd-6164a8c2deac"
            },
            {
                "name": "Header",
                "output": "http://localhost:8095/api/attributes/e09cb2f3-0d84-40d7-b4dd-6164a8c2deac",
                "value": "{'key': 'Accept','value': 'application/json'}"
            },
            {
                "name": "Header",
                "input": "http://localhost:8095/api/attributes/abadfa92-9001-4986-94aa-6f0a6ebe73b9",
```

```
                    "output": "http://localhost:8095/api/attributes/e09cb2f3-0d84-40d7-b4dd-6164a8c2deac",
                    "value": " {'key': 'Authorization','value': 'Bearer ${{input}}'}"
                }
            ],
            "output": [
                {
                    "name": "HTTP Status",
                    "input": "http://localhost:8095/api/attributes/41cb04ae-22d1-4cbe-b871-1945f24de19e",
                    "output": "http://localhost:8095/api/attributes/41cb04ae-22d1-4cbe-b871-1945f24de19e"
                },
                {
                    "name": "ORCiD profile",
                    "input": "http://localhost:8095/api/attributes/34e2a7ea-f9ac-4125-859c-99e67aa1136a",
                    "output": "http://localhost:8095/api/attributes/bbded7e7-df26-4638-a80a-7bb470d13433"
                }
            ]
        }
    ]
}
```

Listing A.38: Request zum Erstellen einer komplexen Operation für die Abfrage eines ORCiD-Profils

```
HTTP/1.1 201 Created
Content-Type: application/hal+json
Location: http://localhost:8095/api/operations/c1f0f675-d1d7-4bed-b43f-a264a958d24b
ETag: "0"

{
    "_embedded": {
        "operations": [
            {
                "pid": "c1f0f675-d1d7-4bed-b43f-a264a958d24b",
                "createdAt": "2024-08-26T10:02:39.029469Z",
                "lastModifiedAt": "2024-08-26T10:02:39.029469Z",
                "standards": null,
                "name": "Get ORCiD Profile Information via REST API",
                "description": "This Operation retrieves the profile information of an ORCiD ID via the ORCiD API. The ORCiD number sequence is extracted from the ORCiD-Profile URL. An API Key for the ORCiD API is required and to be set in the environment.",
                "execution": [
                    {
                        "executionOrderIndex": 0,
```

```
                        "name": "Extract ORCiD ID from ORCiD-URL",
                        "mode": "sync",
                        "steps": null,
                        "attributes": [
                            {
                                "name": "Regex input from orcidURL",
                                "replaceCharactersInValueWithInput": "${{input}}",
                                "value": null,
                                "index": null,
                                "_links": {
                                    "output": {
                                        "href": "http://localhost:8095/api/attributes/398ef1b0-d3bb-4665-ad90-ee9be990ed36"
                                    },
                                    "input": {
                                        "href": "http://localhost:8095/api/attributes/398ef1b0-d3bb-4665-ad90-ee9be990ed36"
                                    }
                                }
                            }
                        ],
                        "output": [
                            {
                                "name": "ORCiD number",
                                "replaceCharactersInValueWithInput": "${{input}}",
                                "value": null,
                                "index": null,
                                "_links": {
                                    "output": {
                                        "href": "http://localhost:8095/api/attributes/73551d9b-f18f-4a1c-8724-17fddbedb738"
                                    },
                                    "input": {
                                        "href": "http://localhost:8095/api/attributes/73551d9b-f18f-4a1c-8724-17fddbedb738"
                                    }
                                }
                            }
                        ],
                        "_links": {
                            "operation": {
                                "href": "http://localhost:8095/api/operations/c379e4ca-3184-4dea-99a9-89994ab54690"
                            }
                        }
                    },
                    {
                        "_embedded": {
                            "operationSteps": [
```

```
                                {
                                    "executionOrderIndex": 1,
                                    "name": "Get profile data from ORCID.org API",
                                    "mode": "sync",
                                    "steps": null,
                                    "attributes": [
                                        {
                                            "_embedded": {
                                                "attributeMappings": [
                                                    {
                                                        "name": "URL",
                                                        "replaceCharactersInValueWithInput": "${{input}}",
                                                        "value": "https://orcid.org/api/${{input}}",
                                                        "index": null,
                                                        "_links": {
                                                            "output": {
                                                                "href": "http://localhost:8095/api/attributes/d0d63d94-1176-4b00-8ba8-9ea243ce17be"
                                                            },
                                                            "input": {
                                                                "href": "http://localhost:8095/api/attributes/73551d9b-f18f-4a1c-8724-17fddbedb738"
                                                            }
                                                        }
                                                    }
                                                ],
                                                "hashMaps": [
                                                    {
                                                        "SyntaxValidator": {
                                                            "ERROR": [
                                                                {
                                                                    "message": "The input is repeatable, the output is not repeatable and no index is specified.",
                                                                    "element": {
                                                                        "name": "URL",
                                                                        "replaceCharactersInValueWithInput": "${{input}}",
                                                                        "value": "https://orcid.org/api/${{input}}",
                                                                        "index": null
                                                                    },
                                                                    "severity": "ERROR"
                                                                }
                                                            ]
                                                        }
                                                    }
                                                ]
```

```
                                    }
                                },
                                {
                                    "name": "Method",
                                    "replaceCharactersInValueWithInput": "${{input
    }}",
                                    "value": "GET",
                                    "index": null,
                                    "_links": {
                                        "output": {
                                            "href": "http://localhost:8095/api/
    attributes/e09cb2f3-0d84-40d7-b4dd-6164a8c2deac"
                                        }
                                    }
                                },
                                {
                                    "name": "Header",
                                    "replaceCharactersInValueWithInput": "${{input
    }}",
                                    "value": "{'key': 'Accept','value': '
    application/json'}",
                                    "index": null,
                                    "_links": {
                                        "output": {
                                            "href": "http://localhost:8095/api/
    attributes/e09cb2f3-0d84-40d7-b4dd-6164a8c2deac"
                                        }
                                    }
                                },
                                {
                                    "name": "Header",
                                    "replaceCharactersInValueWithInput": "${{input
    }}",
                                    "value": " {'key': 'Authorization','value': '
    Bearer ${{input}}'}",
                                    "index": null,
                                    "_links": {
                                        "output": {
                                            "href": "http://localhost:8095/api/
    attributes/e09cb2f3-0d84-40d7-b4dd-6164a8c2deac"
                                        },
                                        "input": {
                                            "href": "http://localhost:8095/api/
    attributes/abadfa92-9001-4986-94aa-6f0a6ebe73b9"
                                        }
                                    }
                                }
                            ],
                            "output": [
```

```
                                    {
                                        "name": "HTTP Status",
                                        "replaceCharactersInValueWithInput": "${{input
    }}",
                                        "value": null,
                                        "index": null,
                                        "_links": {
                                            "output": {
                                                "href": "http://localhost:8095/api/
    attributes/41cb04ae-22d1-4cbe-b871-1945f24de19e"
                                            },
                                            "input": {
                                                "href": "http://localhost:8095/api/
    attributes/41cb04ae-22d1-4cbe-b871-1945f24de19e"
                                            }
                                        }
                                    },
                                    {
                                        "name": "ORCiD profile",
                                        "replaceCharactersInValueWithInput": "${{input
    }}",
                                        "value": null,
                                        "index": null,
                                        "_links": {
                                            "output": {
                                                "href": "http://localhost:8095/api/
    attributes/bbded7e7-df26-4638-a80a-7bb470d13433"
                                            },
                                            "input": {
                                                "href": "http://localhost:8095/api/
    attributes/34e2a7ea-f9ac-4125-859c-99e67aa1136a"
                                            }
                                        }
                                    }
                                ],
                                "_links": {
                                    "operationTypeProfile": {
                                        "href": "http://localhost:8095/api/
    operationTypeProfiles/02661f03-31e4-4e03-97df-8155482e166b"
                                    }
                                }
                            }
                        ],
                        "hashMaps": [
                            {
                                "SyntaxValidator": {
                                    "ERROR": [
                                        {
```

```
                                            "message": "The input is repeatable, the
    output is not repeatable and no index is specified.",
                                            "element": {
                                                "name": "URL",
                                                "replaceCharactersInValueWithInput": "
    ${{input}}",
                                                "value": "https://orcid.org/api/${{
    input}}",
                                                "index": null
                                            },
                                            "severity": "ERROR"
                                        }
                                    ]
                                }
                            }
                        ]
                    }
                }
            ],
            "_links": {
                "self": {
                    "href": "http://localhost:8095/api/operations/c1f0f675-d1d7-4bed-
    b43f-a264a958d24b"
                },
                "operation": {
                    "href": "http://localhost:8095/api/operations/c1f0f675-d1d7-4bed-
    b43f-a264a958d24b"
                },
                "environment": {
                    "href": "http://localhost:8095/api/operations/c1f0f675-d1d7-4bed-
    b43f-a264a958d24b/environment"
                },
                "executableOn": {
                    "href": "http://localhost:8095/api/operations/c1f0f675-d1d7-4bed-
    b43f-a264a958d24b/executableOn"
                },
                "contributors": {
                    "href": "http://localhost:8095/api/operations/c1f0f675-d1d7-4bed-
    b43f-a264a958d24b/contributors"
                },
                "license": {
                    "href": "http://localhost:8095/api/operations/c1f0f675-d1d7-4bed-
    b43f-a264a958d24b/license"
                },
                "returns": {
                    "href": "http://localhost:8095/api/operations/c1f0f675-d1d7-4bed-
    b43f-a264a958d24b/returns"
                }
            }
```

```
            }
        ],
        "hashMaps": [
            {
                "SyntaxValidator": {
                    "ERROR": [
                        {
                            "message": "The input is repeatable, the output is not repeatable and no index is specified.",
                            "element": {
                                "name": "URL",
                                "replaceCharactersInValueWithInput": "${{input}}",
                                "value": "https://orcid.org/api/${{input}}",
                                "index": null
                            },
                            "severity": "ERROR"
                        }
                    ]
                }
            }
        ]
    }
}
```

Listing A.39: Response auf Erstellen einer komplexen Operation für die Abfrage eines ORCiD-Profils

A.3.7. Suche nach ausführbaren Operationen

```
MATCH (d:DataType {pid: "b5a64c2f-68eb-4c32-bff6-afd39394a85a"})
-[:attributes]->(:Attribute)<-[:executableOn]-(o:Operation) RETURN o
UNION
MATCH (d:DataType {pid: "b5a64c2f-68eb-4c32-bff6-afd39394a85a"})
-[:attributes]->(:Attribute)-[:dataType]->(:DataType)
<-[:dataType]-(:Attribute)<-[:executableOn]-(o:Operation) RETURN o
UNION
MATCH (d:DataType {pid: "b5a64c2f-68eb-4c32-bff6-afd39394a85a"})
-[:inheritsFrom*]->(:DataType)-[:attributes]->(:Attribute)
-[:dataType]->(:DataType)<-[:dataType]-(:Attribute)
<-[:executableOn]-(o:Operation) RETURN o
UNION
MATCH (d:DataType {pid: "b5a64c2f-68eb-4c32-bff6-afd39394a85a"})
-[:inheritsFrom*]->(:DataType)-[:attributes]->(:Attribute)
-[:dataType]->(:DataType)-[:inheritsFrom*]->(:DataType)
<-[:dataType]-(:Attribute)<-[:executableOn]-(o:Operation) RETURN o
```

Listing A.40: Cypher-Query zum Suchen von Operationen für einen bestimmten Datentyp

```
GET /operations/search/getOperationsForDataType?pid=b5a64c2f-68eb-4c32-bff6-afd39394a85a
    HTTP/1.1
Host: http://localhost:8095/api
```

Listing A.41: Request zum Suchen von Operationen für einen bestimmten Datentyp

```
HTTP/1.1 200 OK
Content-Type: application/hal+json

{
    "_embedded": {
        "collectionModels": [
            {
                "_embedded": {
                    "operations": [
                        {
                            "pid": "c379e4ca-3184-4dea-99a9-89994ab54690",
                            "createdAt": "2024-08-26T09:56:34.648525Z",
                            "lastModifiedAt": "2024-08-26T10:02:39.208828Z",
                            "standards": [],
                            "name": "Extract ORCiD number out of text",
                            "description": "This operation extracts an ORCiD number out of
     a URL and returns it.",
                            "execution": [],
                            "_links": {
                                "self": {
                                    "href": "http://localhost:8095/api/operations/c379e4ca
    -3184-4dea-99a9-89994ab54690"
                                },
                                "operation": {
                                    "href": "http://localhost:8095/api/operations/c379e4ca
    -3184-4dea-99a9-89994ab54690"
                                },
                                "contributors": {
                                    "href": "http://localhost:8095/api/operations/c379e4ca
    -3184-4dea-99a9-89994ab54690/contributors"
                                },
                                "environment": {
                                    "href": "http://localhost:8095/api/operations/c379e4ca
    -3184-4dea-99a9-89994ab54690/environment"
                                },
                                "license": {
                                    "href": "http://localhost:8095/api/operations/c379e4ca
    -3184-4dea-99a9-89994ab54690/license"
                                },
                                "executableOn": {
                                    "href": "http://localhost:8095/api/operations/c379e4ca
    -3184-4dea-99a9-89994ab54690/executableOn"
```

```
                                },
                                "returns": {
                                    "href": "http://localhost:8095/api/operations/c379e4ca-3184-4dea-99a9-89994ab54690/returns"
                                }
                            }
                        }
                    ],
                    "hashMaps": [
                        {
                            "SyntaxValidator": {
                                "INFO": [
                                    {
                                        "message": "There are no return values provided for this Operation.",
                                        "element": {
                                            "pid": "c379e4ca-3184-4dea-99a9-89994ab54690",
                                            "createdAt": "2024-08-26T09:56:34.648525Z",
                                            "lastModifiedAt": "2024-08-26T10:02:39.208828Z",
                                            "standards": [],
                                            "name": "Extract ORCiD number out of text",
                                            "description": "This operation extracts an ORCiD number out of a URL and returns it.",
                                            "execution": []
                                        },
                                        "severity": "INFO"
                                    }
                                ],
                                "ERROR": [
                                    {
                                        "message": "You MUST specify an attribute on which the operation can be executed.",
                                        "element": {
                                            "pid": "c379e4ca-3184-4dea-99a9-89994ab54690",
                                            "createdAt": "2024-08-26T09:56:34.648525Z",
                                            "lastModifiedAt": "2024-08-26T10:02:39.208828Z",
                                            "standards": [],
                                            "name": "Extract ORCiD number out of text",
                                            "description": "This operation extracts an ORCiD number out of a URL and returns it.",
                                            "execution": []
                                        },
                                        "severity": "ERROR"
                                    },
                                    {
                                        "message": "You MUST specify at least one execution step for a valid operation.",
```

```
                                    "element": {
                                        "pid": "c379e4ca-3184-4dea-99a9-89994ab54690",
                                        "createdAt": "2024-08-26T09:56:34.648525Z",
                                        "lastModifiedAt": "2024-08-26T10:02:39.208828Z
    ",
                                        "standards": [],
                                        "name": "Extract ORCiD number out of text",
                                        "description": "This operation extracts an
    ORCiD number out of a URL and returns it.",
                                        "execution": []
                                    },
                                    "severity": "ERROR"
                                }
                            ]
                        }
                    }
                ]
            }
        },
        {
            "_embedded": {
                "operations": [
                    {
                        "pid": "c1f0f675-d1d7-4bed-b43f-a264a958d24b",
                        "createdAt": "2024-08-26T10:02:39.029469Z",
                        "lastModifiedAt": "2024-08-26T10:02:39.029469Z",
                        "standards": [],
                        "name": "Get ORCiD Profile Information via REST API",
                        "description": "This Operation retrieves the profile
    information of an ORCiD ID via the ORCiD API. The ORCiD number sequence is extracted
    from the ORCiD-Profile URL. An API Key for the ORCiD API is required and to be set in
    the environment.",
                        "execution": [],
                        "_links": {
                            "self": {
                                "href": "http://localhost:8095/api/operations/c1f0f675
    -d1d7-4bed-b43f-a264a958d24b"
                            },
                            "operation": {
                                "href": "http://localhost:8095/api/operations/c1f0f675
    -d1d7-4bed-b43f-a264a958d24b"
                            },
                            "contributors": {
                                "href": "http://localhost:8095/api/operations/c1f0f675
    -d1d7-4bed-b43f-a264a958d24b/contributors"
                            },
                            "environment": {
                                "href": "http://localhost:8095/api/operations/c1f0f675
    -d1d7-4bed-b43f-a264a958d24b/environment"
```

```
                                },
                                "license": {
                                    "href": "http://localhost:8095/api/operations/c1f0f675-d1d7-4bed-b43f-a264a958d24b/license"
                                },
                                "executableOn": {
                                    "href": "http://localhost:8095/api/operations/c1f0f675-d1d7-4bed-b43f-a264a958d24b/executableOn"
                                },
                                "returns": {
                                    "href": "http://localhost:8095/api/operations/c1f0f675-d1d7-4bed-b43f-a264a958d24b/returns"
                                }
                            }
                        }
                    ],
                    "hashMaps": [
                        {
                            "SyntaxValidator": {
                                "INFO": [
                                    {
                                        "message": "There are no return values provided for this Operation.",
                                        "element": {
                                            "pid": "c1f0f675-d1d7-4bed-b43f-a264a958d24b",
                                            "createdAt": "2024-08-26T10:02:39.029469Z",
                                            "lastModifiedAt": "2024-08-26T10:02:39.029469Z",
                                            "standards": [],
                                            "name": "Get ORCiD Profile Information via REST API",
                                            "description": "This Operation retrieves the profile information of an ORCiD ID via the ORCiD API. The ORCiD number sequence is extracted from the ORCiD-Profile URL. An API Key for the ORCiD API is required and to be set in the environment.",
                                            "execution": []
                                        },
                                        "severity": "INFO"
                                    }
                                ],
                                "ERROR": [
                                    {
                                        "message": "You MUST specify an attribute on which the operation can be executed.",
                                        "element": {
                                            "pid": "c1f0f675-d1d7-4bed-b43f-a264a958d24b",
                                            "createdAt": "2024-08-26T10:02:39.029469Z",
                                            "lastModifiedAt": "2024-08-26T10:02:39.029469Z",
```

```
                                        "standards": [],
                                        "name": "Get ORCiD Profile Information via REST API",
                                        "description": "This Operation retrieves the profile information of an ORCiD ID via the ORCiD API. The ORCiD number sequence is extracted from the ORCiD-Profile URL. An API Key for the ORCiD API is required and to be set in the environment.",
                                        "execution": []
                                    },
                                    "severity": "ERROR"
                                },
                                {
                                    "message": "You MUST specify at least one execution step for a valid operation.",
                                    "element": {
                                        "pid": "c1f0f675-d1d7-4bed-b43f-a264a958d24b",
                                        "createdAt": "2024-08-26T10:02:39.029469Z",
                                        "lastModifiedAt": "2024-08-26T10:02:39.029469Z",
                                        "standards": [],
                                        "name": "Get ORCiD Profile Information via REST API",
                                        "description": "This Operation retrieves the profile information of an ORCiD ID via the ORCiD API. The ORCiD number sequence is extracted from the ORCiD-Profile URL. An API Key for the ORCiD API is required and to be set in the environment.",
                                        "execution": []
                                    },
                                    "severity": "ERROR"
                                }
                            ]
                        }
                    }
                ]
            }
        }
    ],
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/operations/search/getOperationsForDataType?pid=b5a64c2f-68eb-4c32-bff6-afd39394a85a"
        }
    }
}
```


Listing A.42: Response auf Suchen von Operationen für einen bestimmten Datentyp

A.3.8. Auflösung der Vererbungshierarchie

```
HTTP/1.1 200 OK
Content-Type: application/hal+json

{
    "_embedded": {
        "typeProfiles": [
            {
                "pid": "dff4084f-67f9-4229-96df-2fab738cf96e",
                "createdAt": "2024-08-26T09:23:58.054886Z",
                "lastModifiedAt": "2024-08-26T16:22:40.677867Z",
                "standards": [],
                "type": "TypeProfile",
                "name": "Key-Value pair",
                "description": "This TypeProfile represents a single key-value pair. To create a set of KV-Pairs, just repeat this TypeProfile.",
                "expectedUses": [
                    "Representing elements in map data structures",
                    "Single HTTP-Header"
                ],
                "defaultValue": null,
                "subSchemaRelation": "allowAdditionalProperties",
                "abstract": false,
                "embeddable": true,
                "_links": {
                    "self": {
                        "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e"
                    },
                    "typeProfile": {
                        "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e"
                    },
                    "validate": {
                        "href": "/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/validate"
                    },
                    "inheritedAttributes": {
                        "href": "/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/inheritedAttributes"
                    },
                    "inheritanceTree": {
                        "href": "/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/inheritanceTree"
                    },
                    "operations": {
                        "href": "http://localhost:8095/api/operations/search/getOperationsForDataType?pid=dff4084f-67f9-4229-96df-2fab738cf96e"
```

```
                },
                "attributes": {
                    "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/attributes"
                },
                "inheritsFrom": {
                    "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/inheritsFrom"
                },
                "contributors": {
                    "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/contributors"
                },
                "license": {
                    "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/license"
                }
            }
        },
        {
            "pid": "7480c1eb-6530-4042-b1f1-69bfd57ec52e",
            "createdAt": "2024-08-26T16:21:20.040272Z",
            "lastModifiedAt": "2024-08-26T16:22:40.782488Z",
            "standards": [],
            "type": "TypeProfile",
            "name": "Even more useless",
            "description": "Just for testing of multi-inheritance",
            "expectedUses": [
                "DO NOT USE!",
                "Just for testing purposes"
            ],
            "defaultValue": null,
            "subSchemaRelation": "allowAdditionalProperties",
            "abstract": false,
            "embeddable": true,
            "_links": {
                "self": {
                    "href": "http://localhost:8095/api/typeProfiles/7480c1eb-6530-4042-b1f1-69bfd57ec52e"
                },
                "typeProfile": {
                    "href": "http://localhost:8095/api/typeProfiles/7480c1eb-6530-4042-b1f1-69bfd57ec52e"
                },
                "validate": {
                    "href": "/api/typeProfiles/7480c1eb-6530-4042-b1f1-69bfd57ec52e/validate"
                },
                "inheritedAttributes": {
```

```
                        "href": "/api/typeProfiles/7480c1eb-6530-4042-b1f1-69bfd57ec52e/
    inheritedAttributes"
                    },
                    "inheritanceTree": {
                        "href": "/api/typeProfiles/7480c1eb-6530-4042-b1f1-69bfd57ec52e/
    inheritanceTree"
                    },
                    "operations": {
                        "href": "http://localhost:8095/api/operations/search/
    getOperationsForDataType?pid=7480c1eb-6530-4042-b1f1-69bfd57ec52e"
                    },
                    "attributes": {
                        "href": "http://localhost:8095/api/typeProfiles/7480c1eb
    -6530-4042-b1f1-69bfd57ec52e/attributes"
                    },
                    "inheritsFrom": {
                        "href": "http://localhost:8095/api/typeProfiles/7480c1eb
    -6530-4042-b1f1-69bfd57ec52e/inheritsFrom"
                    },
                    "contributors": {
                        "href": "http://localhost:8095/api/typeProfiles/7480c1eb
    -6530-4042-b1f1-69bfd57ec52e/contributors"
                    },
                    "license": {
                        "href": "http://localhost:8095/api/typeProfiles/7480c1eb
    -6530-4042-b1f1-69bfd57ec52e/license"
                    }
                }
            }
        ]
    },
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-
    b6d0efa4ab85/inheritsFrom"
        }
    }
}
```

Listing A.43: Auflösung der Vererbungshierarchie eines Datentyps mithilfe einer `GET`-Request für `/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85/inheritsFrom`

```
HTTP/1.1 200 OK
Content-Type: application/hal+json

{
    "_embedded": {
        "attributes": [
```

```
            {
                "name": "Key",
                "description": "The key of a key-value pair",
                "repeatable": false,
                "obligation": "Mandatory",
                "value": null,
                "_links": {
                    "dataType": {
                        "href": "http://localhost:8095/api/dataTypes/e2c9ea28-6d63-4188-9cc7-1492d6709e2a"
                    }
                }
            },
            {
                "name": "Value",
                "description": "The value of a key-value pair",
                "repeatable": false,
                "obligation": "Mandatory",
                "value": null,
                "_links": {
                    "dataType": {
                        "href": "http://localhost:8095/api/dataTypes/e2c9ea28-6d63-4188-9cc7-1492d6709e2a"
                    }
                }
            },
            {
                "name": "Useless Dummy",
                "description": "This attribute is an absolutely useless text. DO NOT USE!",
                "repeatable": false,
                "obligation": "Mandatory",
                "value": null,
                "_links": {
                    "dataType": {
                        "href": "http://localhost:8095/api/dataTypes/e2c9ea28-6d63-4188-9cc7-1492d6709e2a"
                    }
                }
            }
        ]
    },
    "_links": {
        "self": {
            "href": "http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85/inheritedAttributes"
        },
        "typeProfile": {
```

```
            "href": "http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85"
        }
    }
}
```

Listing A.44: Auflösung der Vererbungshierarchie eines Datentyps mithilfe einer `GET`-Request für `/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85/inheritedAttributes`

```
HTTP/1.1 200 OK
Content-Type: application/hal+json

{
    "pid": "3a2bfb7e-be49-4b92-8476-b6d0efa4ab85",
    "name": "HTTP Header",
    "description": "A single HTTP Header",
    "attributes": {},
    "inheritsFrom": {
        "_embedded": {
            "typeProfileInheritances": [
                {
                    "pid": "dff4084f-67f9-4229-96df-2fab738cf96e",
                    "name": "Key-Value pair",
                    "description": "This TypeProfile represents a single key-value pair. To create a set of KV-Pairs, just repeat this TypeProfile.",
                    "attributes": {
                        "_embedded": {
                            "attributes": [
                                {
                                    "name": "Key",
                                    "description": "The key of a key-value pair",
                                    "repeatable": false,
                                    "obligation": "Mandatory",
                                    "value": null,
                                    "_links": {
                                        "dataType": {
                                            "href": "http://localhost:8095/api/dataTypes/e2c9ea28-6d63-4188-9cc7-1492d6709e2a"
                                        }
                                    }
                                },
                                {
                                    "name": "Value",
                                    "description": "The value of a key-value pair",
                                    "repeatable": false,
                                    "obligation": "Mandatory",
                                    "value": null,
                                    "_links": {
```

```
                                    "dataType": {
                                        "href": "http://localhost:8095/api/dataTypes/e2c9ea28-6d63-4188-9cc7-1492d6709e2a"
                                    }
                                }
                            }
                        ]
                    }
                },
                "inheritsFrom": {},
                "_links": {
                    "inheritanceTree": {
                        "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/inheritanceTree"
                    },
                    "typeProfile": {
                        "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e"
                    },
                    "attributes": {
                        "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/attributes"
                    },
                    "inheritedAttributes": {
                        "href": "http://localhost:8095/api/typeProfiles/dff4084f-67f9-4229-96df-2fab738cf96e/inheritedAttributes"
                    },
                    "operations": {
                        "href": "http://localhost:8095/api/operations/search/getOperationsForDataType?pid=dff4084f-67f9-4229-96df-2fab738cf96e"
                    }
                }
            },
            {
                "pid": "7480c1eb-6530-4042-b1f1-69bfd57ec52e",
                "name": "Even more useless",
                "description": "Just for testing of multi-inheritance",
                "attributes": {},
                "inheritsFrom": {
                    "_embedded": {
                        "typeProfileInheritances": [
                            {
                                "pid": "29c7d6fb-21df-469d-b75a-dae2811b008d",
                                "name": "Useless",
                                "description": "Just for testing of multi-inheritance",
                                "attributes": {
                                    "_embedded": {
                                        "attributes": [
```

```
                                        {
                                            "name": "Useless Dummy",
                                            "description": "This attribute is an absolutely useless text. DO NOT USE!",
                                            "repeatable": false,
                                            "obligation": "Mandatory",
                                            "value": null,
                                            "_links": {
                                                "dataType": {
                                                    "href": "http://localhost:8095/api/dataTypes/e2c9ea28-6d63-4188-9cc7-1492d6709e2a"
                                                }
                                            }
                                        }
                                    ]
                                }
                            },
                            "inheritsFrom": {},
                            "_links": {
                                "inheritanceTree": {
                                    "href": "http://localhost:8095/api/typeProfiles/29c7d6fb-21df-469d-b75a-dae2811b008d/inheritanceTree"
                                },
                                "typeProfile": {
                                    "href": "http://localhost:8095/api/typeProfiles/29c7d6fb-21df-469d-b75a-dae2811b008d"
                                },
                                "attributes": {
                                    "href": "http://localhost:8095/api/typeProfiles/29c7d6fb-21df-469d-b75a-dae2811b008d/attributes"
                                },
                                "inheritedAttributes": {
                                    "href": "http://localhost:8095/api/typeProfiles/29c7d6fb-21df-469d-b75a-dae2811b008d/inheritedAttributes"
                                },
                                "operations": {
                                    "href": "http://localhost:8095/api/operations/search/getOperationsForDataType?pid=29c7d6fb-21df-469d-b75a-dae2811b008d"
                                }
                            }
                        }
                    ]
                }
            },
            "_links": {
                "inheritanceTree": {
                    "href": "http://localhost:8095/api/typeProfiles/7480c1eb-6530-4042-b1f1-69bfd57ec52e/inheritanceTree"
                },
```

```
                        "typeProfile": {
                            "href": "http://localhost:8095/api/typeProfiles/7480c1eb-6530-4042-b1f1-69bfd57ec52e"
                        },
                        "attributes": {
                            "href": "http://localhost:8095/api/typeProfiles/7480c1eb-6530-4042-b1f1-69bfd57ec52e/attributes"
                        },
                        "inheritedAttributes": {
                            "href": "http://localhost:8095/api/typeProfiles/7480c1eb-6530-4042-b1f1-69bfd57ec52e/inheritedAttributes"
                        },
                        "operations": {
                            "href": "http://localhost:8095/api/operations/search/getOperationsForDataType?pid=7480c1eb-6530-4042-b1f1-69bfd57ec52e"
                        }
                    }
                }
            ]
        }
    },
    "_links": {
        "inheritanceTree": {
            "href": "http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85/inheritanceTree"
        },
        "typeProfile": {
            "href": "http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85"
        },
        "attributes": {
            "href": "http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85/attributes"
        },
        "inheritedAttributes": {
            "href": "http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85/inheritedAttributes"
        },
        "operations": {
            "href": "http://localhost:8095/api/operations/search/getOperationsForDataType?pid=3a2bfb7e-be49-4b92-8476-b6d0efa4ab85"
        },
        "self": {
            "href": "http://localhost:8095/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85/inheritanceTree"
        }
    }
}
```

Listing A.45: Auflösung der Vererbungshierarchie eines Datentyps mithilfe einer `GET`-Request für `/api/typeProfiles/3a2bfb7e-be49-4b92-8476-b6d0efa4ab85/inheritanceTree`