TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Rechnernetze

Proof of Concept Implementierung und Bewertung eines Systems zur Generierung von modellierten xAPI-Statements zum Befüllen eines Learning Records Stores

Konstantin Köhring

Geboren am: 17. November 1997 in Quedlinburg
Matrikelnummer: 4604507
Immatrikulationsjahr: 2018

Bachelor-Arbeit

zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

Betreuer
Dr.-Ing. Iris Braun
Dr.-Ing. Tommy Kubica

Betreuender Hochschullehrer
Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Eingereicht am: 16. Februar 2022

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur / Professur für Rechnernetze

Aufgabenstellung für die Bachelor-Arbeit

Thema: Proof of Concept Implementierung und Bewertung eines Systems zur Generierung von modellierten xAPI-Statements zum Befüllen eines Learning Records Stores

Name: Köhring, Konstantin	**Studiengang:** Bachelor Informatik
Matrikel-Nummer: 4604507	**Projekt/Fokus:** VerDatAs, eLearning
Verantwortliche/r Hochschullehrerin:	Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill
Betreuer/in:	Dr.-Ing. Iris Braun, M.Sc. Tommy Kubica
Beginn: 01.12.2021	**Abgabe:** 16.02.2022

Zielstellung

Die weiter fortschreitende Digitalisierung der Gesellschaft bringt auch neue Einsatzmöglichkeiten für digitale Lernangebote mit sich. Aufgrund der COVID-19-Pandemie hat sich gezeigt, dass digitale Alternativen oder Ergänzungsangebote zur klassischen Präsenzlehre ein wichtiger Schritt in die Zukunft der Bildung insgesamt sind.

Im Rahmen des Projektes VerDatAs (Vernetzung und datengestützte Assistenz für die berufliche Bildung) soll ein tutorielles Assistenzsystem (TAS) konzipiert werden, das Nutzer beim selbstgesteuerten Lernen unterstützt. Dabei soll es ermöglicht werden, die Suche nach Informationen zu vereinfachen, Lehrstoff und Lernprozesse zu visualisieren, Problemlöseschritte zu optimieren und aus Prozessanalysen Empfehlungen für weitere Lernwege abzuleiten.

Um entsprechende Empfehlungen ableiten zu können, benötigt das Assistenzsystem Informationen über den aktuellen Lernstand des Lernenden und wie er bisher im Lernangebot navigiert hat. Diese Informationen werden von Learning Management Systemen wie z.B. ILIAS über eine xAPI-Schnittstelle zur Verfügung gestellt und in einem sogenannten Learning Record Store gespeichert und für spätere Analysen bereitgestellt.

Innerhalb dieser Bachelorarbeit soll nun ein Prototyp implementiert werden, der einen Learning Record Store (LRS) mit simulierten Testdaten befüllen, diese wieder auslesen und schlussendlich die Daten, welche Lernplattform und LRS bereitstellen, analysieren kann. Grundlage der Implementierung soll dabei das von der Advanced Distributed Learning Initiative (ADL) projektierte Open-Source-Programm DATASIM sein, ein Testframework mit Datensimulator für die Total Learning Architecture (TLA).

A. Schill

Digital unterschrieben
von Alexander Schill
Datum: 2021.11.30
18:36:58 +01'00'

Prof. Dr. Alexander Schill

Schwerpunkte

- State-of-the-Art-Analyse bzgl. Learning Record Store und xAPI-Schnittstellen,
- Definieren von Anforderungen an die zu generierenden Statements im LRS,
- Konzept für die Erzeugung simulierter LRS-Statements,
- Umsetzung und Evaluation einer Proof-of-Concept-Implementierung.

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich das vorliegende Dokument mit dem Titel *Proof of Concept Implementierung und Bewertung eines Systems zur Generierung von modellierten xAPI-Statements zum Befüllen eines Learning Records Stores* selbstständig und ohne unzulässige Hilfe Dritter verfasst habe. Es wurden keine anderen als die in diesem Dokument angegebenen Hilfsmittel und Quellen benutzt. Die wörtlichen und sinngemäß übernommenen Zitate habe ich als solche kenntlich gemacht. Es waren keine weiteren Personen an der geistigen Herstellung des vorliegenden Dokumentes beteiligt. Mir ist bekannt, dass die Nichteinhaltung dieser Erklärung zum nachträglichen Entzug des Hochschulabschlusses führen kann.

Dresden, 16. Februar 2022

Konstantin Köhring

Digitally signed by Konstantin Koehring
Date: 2022.02.16 13:29:25 CET
Reason: Einreichung BA
Location: Freital

Inhaltsverzeichnis

1. Einführung in Lernplattformen

Zu Beginn dieser Arbeit soll die Wichtigkeit des Themas aufgrund des regen wissenschaftlichen und pädagogischen Bedarfes und wirtschaftlichen Interesses belegt werden. Dazu wird in Abschnitt 1.1 auf die aktuelle Verbreitung und mögliche Anwendungszwecke von Lernplattformen eingegangen. Die ökonomische Bedeutung des Wirtschaftszweiges *E-Learning* wird in Abschnitt 1.2 analysiert. Danach werden in Abschnitt 1.3 Begriffe, die zum Verständnis der Arbeit von Belang sein können, erörtert und abschließend in Abschnitt 1.4 die kontextuelle Einbettung des Themas in die sogenannte Total Learning Architecture (TLA) untersucht. In Abbildung 1.2 findet sich ein grafischer Überblick über die grundlegenden Begriffe.

1.1. Verbreitung und Anwendungszwecke

Lernplattformen sind an deutschen Hochschulen weit verbreitet. So geben 74 % der Hochschullehrenden in Deutschland an, mindestens gelegentlich in ihren Lehrveranstaltungen Lernmanagementsysteme zu nutzen. Mehr als die Hälfte der Studierenden in Deutschland findet es motivierend, mit solchen Systemen zu lernen.[12][1]

Auch in Unternehmen werden die Möglichkeiten, welche das digitale Lernen eröffnet, zunehmend genutzt. So gibt ein Drittel der von BITKOM befragten Unternehmen an, *E-Learning* zur Mitarbeitendenbildung zu nutzen. Nur ein Viertel der Firmen ist derzeit nicht an der Integration von digitalen Lernangeboten zur Aus- und Weiterbildung interessiert.[9]

Doch wozu können Lernplattformen in unterschiedlichen Kontexten genutzt werden?

Mögliche Motive für den Einsatz von Lernmanagementsystemen (LMS) sind laut der *American Society for Training & Development* „an erster Stelle die Zentralisierung der Verwaltung von Lernaktivitäten“ sowie das Nachverfolgen von Lernfortschritten. So können solche Lernplattformen durch Bildungseinrichtungen zur Strukturierung von Lehrveranstaltungen, zur Bereitstellung von Lehrmaterialien oder zur Kollaboration der Kursteilnehmenden genutzt werden.[29]

Nah verwandt mit Lernmanagementsystemen sind Kompetenzmanagementsysteme (KMS), deren Zielgruppe vor allem Unternehmen sind. Ihr Ziel ist es, die „Aufgabe, Kompetenzen zu beschreiben, transparent zu machen sowie den Transfer, die Nutzung und Entwicklung der Kompetenzen, orientiert an den persönlichen Zielen des Mitarbeiters sowie

[1] Diese Statistiken wurden 2017 (vor Beginn der Corona-Pandemie) erhoben, wodurch diese Werte nicht mehr aktuell sein könnten.

den Zielen der Unternehmung"[39], zu lösen. Daraus folgt, dass nicht primär die Vermittlung von Wissen, sondern die Lehre von Fähigkeiten angestrebt wird. So betten sich KMS in das sogenannte „Lebenslange Lernen" ein.[29]
Lernmanagementsysteme und Kompetenzmanagementsystemen unterscheiden sich hauptsächlich hinsichtlich ihres Aufbaus. So werden in LMS Lehrinhalte in der Regel in „Kursen" organisiert, während bei KMS „Kompetenzmodelle" verwaltet werden. Üblicherweise können jeweils durch das Absolvieren der untergeordneten Lernmodule Kurse bestanden beziehungsweise Kompetenzen erworben werden.[29]

In dieser Arbeit wird unabhängig von der Typisierung als Lern- oder Kompetenzmanagementsystem auf die Lernfortschritte Bezug genommen, welche in einigen Implementierungen von Lernplattformen in Learning Record Stores (LRS) gespeichert werden.

1.2. Wirtschaftliche Bedeutung

Insbesondere in Westeuropa, Asien und Nordamerika wurde der *E-Learning*-Markt großflächig erschlossen. Für das Jahr 2016 wurde ein Umsatz von 46,67 Milliarden US-Dollar weltweit prognostiziert. Davon entfallen 42,25 Milliarden US-Dollar auf die drei genannten Märkte. Bedingt durch eine Sättigung des Software- und Lerninhaltsbedarfes wird ein Umsatzrückgang von 20 % zwischen 2016 und 2021 vorhergesagt.[2] Dies lässt sich damit begründen, dass die Lerninhalte zwischen verschiedenen Lernplattformen austauschbar sind (siehe Kapitel 2) und die Plattformen damit leicht migrierbar sind, wodurch hauptsächlich horizontale Wechsel[2] ohne Auswirkungen auf den Umsatz durchgeführt werden. Bedingt durch die Corona-Pandemie, welche lange nach Erhebung der Statistik begann, sind diese Werte nicht mehr aktuell. Durch das Pandemiegeschehen und die damit verbundene Einschränkung der Präsenzlehre in allen Bildungsformen und die dadurch induzierte Digitalisierung des Lernens ist der *E-Learning*-Markt exponentiell gewachsen.[35]

Weiterhin bewerten viele Unternehmen *E-Learning* als zunehmend wichtigen Baustein in der Aus- und Weiterbildung. Insbesondere in den Bereichen Nutzerschulung, *Compliance*, Datenschutz, Arbeitssicherheit und IT-Kompetenz schätzen Firmen digitales Lernen als wichtiger werdende Kompetenzvermittlungsstrategie ein.[27]

Eine ebensolche Bewertung konnte einem Gespräch mit dem CTO der Peerox GmbH[3] entnommen werden, nach welchem die klassische Bedienungsanleitung nur eine Daseinsberechtigung als Vorbereitungsdokument für externe Trainer habe, sowie um *Compliance*-Ansprüchen zu genügen. Diese Anleitungen kämen häufig nicht bei den Endnutzern an, die darauf angewiesen wären. *E-Learning* im Kleinen (beispielsweise in Software eingebettete Erklärvideos) und Großen (digitale Kurse) seien sinnvolle Wege, um den tatsächlichen Nutzern Kompetenzen in der Verwendung von Softwareprodukten zu vermitteln.[32]

Weniger bedeutsam für den *E-Learning*-Markt werden die Vermittlung von Fremdsprachen und betriebswirtschaftlichen Themen im beruflichen Kontext bewertet.[27]

Neue Umsatzmöglichkeiten in der Branche bietet beispielsweise die Entwicklung von Lernassistenzsystemen (auch tutorielles Assistenzsystem, LAS/TAS), welche die Lernenden durch das Kursmaterial begleiten und unter anderem anhand von Fortschritts-, Lernelementzustands- und Zeitmessungsmeldungen Vorschläge unterbreiten können, wie das Lernen beziehungsweise der Kompetenzerwerb effizienter und/oder zielführender gestaltet werden kann[41]. Die Analyse, Entwicklung und Evaluation eines solchen Assistenzsystems ist der Schwerpunkt des Forschungsprojektes, in welches diese Bachelorarbeit eingebettet ist. Da

[2] Damit sind Wechsel zwischen Produkten mit ähnlichem Funktionsumfang und einem ähnlichen Preis gemeint.
[3] Markus Windisch; Produkt der Firma ist ein Assistenzsystem für Maschinenbediener.

ein Lernassistenzsystem (ohne Beschränkung der Allgemeinheit) große variante Datenmengen zur Modellbildung (und daraus folgend Vorschlagsberechnung) benötigt, bedarf es eines zeitschonenden Weges, diese Daten zu generieren. Einen Prototyp zur Datengenerierung zu entwickeln ist ein zentraler Aspekt der zu bearbeitenden Aufgabe.

1.3. Begriffserörterungen und Kontext

Als dem kompletten Kontext übergeordnet lässt sich der Begriff *Medienpädagogik* einschätzen. Nach dem häufig zitiertem Professor (eremitus) Ludwig Issing ist Medienpädagogik eine „übergeordnete Bezeichnung für alle pädagogisch orientierten Beschäftigungen mit Medien in Theorie und Praxis"[28].

Unter diesem Begriff ordnen sich nach Professor Kerres und Dr. Preußler die Disziplinen *Mediendidaktik* und *Medienerziehung* unter. Laut ihnen setzt sich die Medienerziehung mit „[dem] reflektierten Medienkonsum und kritischen Umgang mit Medienangeboten" auseinander, während die Mediendidaktik die Untersuchung von „Funktion und Bedeutung von Medien in Lehr- und Lernprozessen" umfasst.[31]
Obwohl diese Disziplinen heute häufig nicht mehr differenziert betrachtet werden können, lässt sich diese Arbeit klar in das Feld der Mediendidaktik einordnen.

Ein Forschungsgebiet der Mediendidaktik ist *E-Learning*. In seinem Buch „Mediendidaktik" definiert Professor Michael Kerres *E-Learning* als „Oberbegriff für alle Varianten der Nutzung digitaler Medien für Lehr- und Lernzwecke, [...] etwa um Wissen zu vermitteln, für den zwischenmenschlichen Austausch oder das gemeinsame Arbeiten an digitalen Artefakten".[30] Offensichtlich kann diese Arbeit dem Thema zugeordnet werden.

Lernplattformen sind eine Möglichkeit, um *E-Learning* anzubieten. Peter Baumgartner definiert webbasierte Lernplattformen als „serverseitig installierte Software [...], die beliebige Lerninhalte über das Internet zu vermitteln hilft und die Organisation der dabei notwendigen Lernprozesse unterstützt"[10]. Im Weiteren bezieht sich der Begriff Lernplattform implizit immer auf webbasierte Lernplattformen.

Eine Ausprägung der Lernplattform ist das *Lernmanagementsystem*. Grundlegende Funktionen einer Lernplattform sind die Präsentation von Lerninhalten, die Bereitstellung von Werkzeugen zur Erstellung von Aufgaben und Übungen sowie von Evaluations- und Bewertungshilfen, die Administration der verfügbaren Entitäten der Plattform[4] und die Kommunikation zwischen sowie unter Lernenden und Lehrenden. Darüber hinaus bieten Lernmanagementsysteme Funktionen zur Unterstützung des Lernprozesses, erweiterte Kommunikationstools und Autorenwerkzeuge an.[19]

Kompetenzmanagementsysteme (siehe Abschnitt 1.2) sind eine weitere Gattung der Lernplattform. Diese liegen jedoch nicht im primären Fokus dieser Arbeit.

Viele moderne Lernmanagementsysteme können mit Learning Record Stores verbunden werden oder diese beinhalten, um detaillierte Lernfortschritte generisch zu erfassen und abrufbar zu hinterlegen. Gemäß einer Definition der *Advanced Distributed Learning Initiative* ist ein Learning Record Store „ein Server [...], welcher für den Empfang, die Speicherung und die Bereitstellung von *xAPI learning records* zuständig ist"[3]

xAPI ist ein Standard zur Aufzeichnung und zum Transfer von Lernerfahrungsdaten (siehe Abschnitt 2.4)[4].

[4] Darunter fallen beispielsweise Lernende und Lehrende, Inhalte, Kurse, Lernfortschritte und Termine.

Die gegenwärtige Forschung im Bereich *E-Learning* umfasst insbesondere die Entwicklung und Evaluation von *Lernanalysesystemen* (*Learning Analytics Systems*), von denen *Lernassistenzsysteme* wiederum eine Untergattung sind. Solche Systeme erstellen auf Basis von Daten, beispielsweise aus Learning Record Stores, „Vorhersagen zu Lernfortschritt und -verhalten, geben Empfehlungen für die inhaltliche und methodische Gestaltung des Lehr- und Lernprozesses [und] ermöglichen neue, dynamische und personalisierte Lernformen".[41] Wie Lernanalysesysteme prinzipiell funktionieren ist in Abbildung 1.1 schematisch abgebildet.

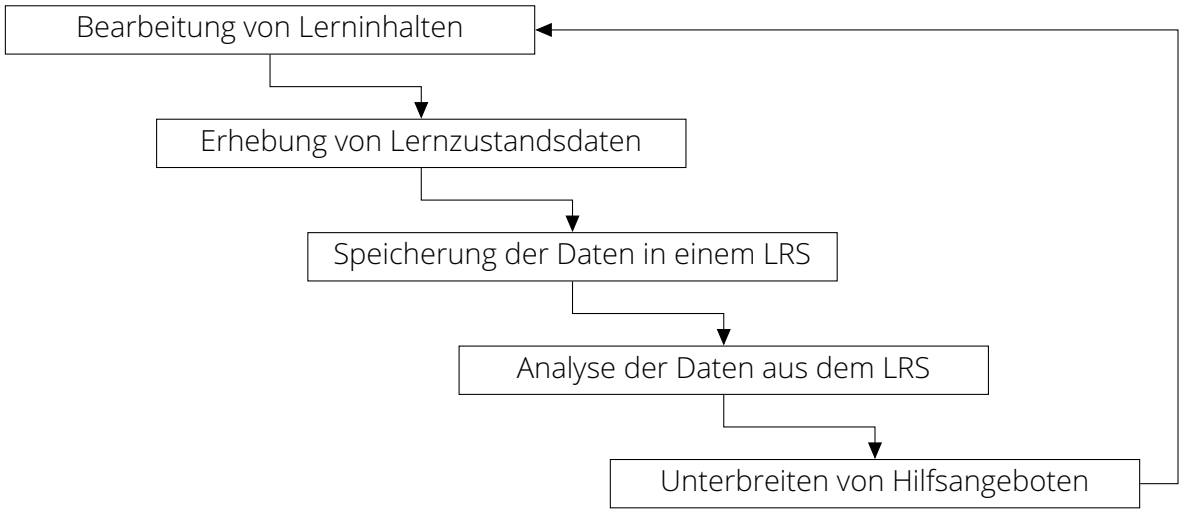


Abbildung 1.1.: Schema zum Lernanalyseprozess. Basierend auf [41]

In Abbildung 1.2 sind die in diesem Abschnitt erklärten Begriffe kontextuell visualisiert.

1.4. Total Learning Architecture

Unter dem Namen *Total Learning Architecture* (TLA) bezeichnet die *Advanced Distributed Learning Initiative*, ein Programm der Regierung der Vereinigten Staaten von Amerika zur Forschung und Entwicklung im Themengebiet des verteilten Lernens, eine Sammlung von Spezifikationen und Standards zur Definition eines einheitlichen Ansatzes zur Integration von existierenden und neuen Lerntechnologien in ein zukünftiges Ökosystem. Die TLA umfasst insbesondere die Datenhaltung und -bereitstellung sowie Datenaustauschschnittstellen. Das Datenmodell ist vierteilig untergliedert in einen Aktivitätenkatalog, Lernaufzeichnungen, Kompetenzverknüpfungen und Lernprofile.[20] Damit bezieht sich die TLA derzeit insbesondere auf Lernmanagementsysteme, Kompetenzmanagementsysteme und Learning Record Stores. Die Ergebnisse dieses Programms sind die Standards IEEE P9274.1 (Experience API 2.0, Lernzustandsmeldungen), IEEE 1484.12.1 (Learning Object Metadata, Lerninhaltsbeschreibungen), IEEE 1484.20.3 (Sharable Competency Definitions, Abbildung von Lerninhalten auf Kompetenzen) und IEEE 2997 (Enterprise Learner Record, Lernprofile)[6][5]. Im Rahmen der Forschung an der TLA wurde unter anderem die Anwendung DATASIM (ein generischer (Zufalls-)Generator für xAPI-Statements) entwickelt, welche in den im Umfang der Arbeit zu erstellenden Prototypen eingebunden werden wird.

[5] Die Titel von IEEE 1484.12.1 und IEEE 2997 sind in der angegebenen Quelle falsch. Hier sind die offiziellen IEEE-Titel angegeben.

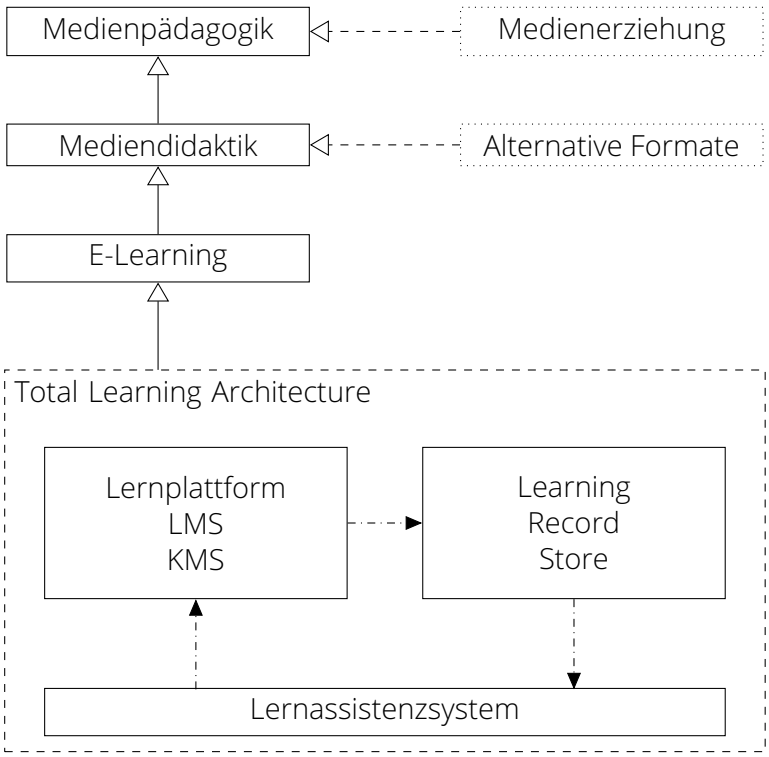


Abbildung 1.2.: Einbettung der Total Learning Architecture in den globalen Kontext.

2. Datenschnittstellen zwischen Lernplattformen und Lerninhalten

Um eine Kommunikation mit Lernplattformen und zwischen Lerninhalten oder externen Tools und LMS zu ermöglichen, bedarf es festgelegter Schnittstellen, über welche Daten in einem jeweils spezifischen Format übertragen werden können. Insbesondere werden umfangreich und detailliert aufgezeichnete Daten von Lernprozessen zur Modellbildung für Lernassistenzsysteme benötigt.

In diesem Kapitel soll auf in der Praxis derzeit übliche und unterschiedlich verbreitete APIs im Rahmen der Total Learning Architecture eingegangen werden. Hierzu wird in Abschnitt 2.1 kurz die relative Verbreitung von Lerninhalten in Bezug auf ihre neueste implementierte Lerndatenschnittstelle analysiert. Danach wird der „ältere" Standard SCORM bezüglich seines Inhaltsmodells und der (wenigen) existierenden Interaktionsmöglichkeiten in Abschnitt 2.2 untersucht. Eine Spezifikation zur verteilten Bereitstellung von Lehrinhalten wird in Abschnitt 2.3 (LTI) näher betrachtet. Der moderne Experience API-Standard wird in Abschnitt 2.4 detailliert auf Basis seines Spezifikationsdokuments beschrieben. Schlussendlich wird in Abschnitt 2.5 auf neuartige Datenschnittstellen, insbesondere zur Nutzung von Lernassistenzsystemen, eingegangen und in Abschnitt 2.6 die Verknüpfung all dieser Kommunikationsmöglichkeiten im Kontext der TLA erörtert.

2.1. Verbreitung von Datenschnittstellen in Lerninhalten

Rustici Software, ein langjähriger Entwickler und Betreiber eines leichtgewichtigen Lernmanagementsystems, hat für das Jahr 2020 ermittelt, welche Schnittstellenspezifikation die Lehrinhalte erfüllen, die in ihre Cloud-Software „SCORM Cloud" importiert werden. Es wurden dazu 624.000 eingespielte Lehrinhalte analysiert. Die Ergebnisse sind in Tabelle 2.1 dargestellt.[1] Die Autoren der Analyse stellen außerdem fest, dass SCORM 1.2 bezogen auf die tatsächlichen Ausführungen (engl. *launches*) von Lehrinhalten einen Marktanteil von ≈ 75 % hält. Der Anteil von xAPI-Inhalten stieg im Vergleich zum vorhergehenden Erhebungszeitraum deutlich.[40]

[1] SCORM 2004 bezieht sich auf alle Ausgaben des Standards. cmi5 ist eine Unterart der xAPI.

Tabelle 2.1.: Verbreitung von Lerndatenschnittstellen 2020. Aggregiert aus [40]

Spezifikation	AICC	SCORM 1.2	SCORM 2004	xAPI (davon cmi5)
Anteil	31.6 %	56.3 %	10.4 %	1.6 % (0.2 %)

2.2. Sharable Content Object Reference Model (SCORM®)

Nach Tabelle 2.1 ist festzustellen, dass der nunmehr volljährige Standard SCORM 1.2 eine wesentlich relevantere Marktposition als sein Nachfolger hat. Deshalb wird an dieser Stelle nicht die moderne Version SCORM 2004 (welche bis 2009 weiterentwickelt wurde) evaluiert, sondern die ältere Spezifikation SCORM 1.2 aus dem Jahr 2001.

Das Sharable Content Object Reference Model (SCORM) ist ein von der ADL entwickelter Standard, welcher Interoperabilität, Wiederverwendbarkeit und Beständigkeit von digitalen Lerninhalten schaffen soll. Der Standard wurde im Auftrag des US-amerikanischen Verteidigungsministeriums zwischen 1999 und 2001 (SCORM 1.2), beziehungsweise 2009 (SCORM 2004) formuliert. Ziel war es, die Austauschbarkeit von Ausbildungsinhalten zwischen einzelnen Regierungsbehörden und auch in der freien Wirtschaft zu ermöglichen.[5]

SCORM bezieht sich ausschließlich auf Lehrinhalte, Nutzerdaten sind darin nicht enthalten. Zur Implementierung von Interoperabilität definiert der Standard „SCORM Run-time Environment" ein Datenmodell und eine Programmierschnittstelle (API) für Lerninhalte. Jedes LMS, welches diesen Standard implementiert, kann SCORM-kompatible Lerninhalte laden und ausführen. Über das API kann ein Lerninhalt einem LMS mitteilen, in welchem Zustand[2] es sich befindet. Außerdem können definierte Daten (beispielsweise Punktzahlen oder Verweilzeiten) zwischen LMS und Lerninhalt ausgetauscht werden.[18]

Portabilität und Wiederverwendbarkeit werden durch das „SCORM Content Aggregation Model" ermöglicht. In diesem Standard ist festgelegt, wie Lerninhalte paketiert werden müssen, um von SCORM-kompatiblen Lernplattformen importiert werden zu können. Ein solches Inhaltspaket besteht aus der Beschreibung der Struktur eines Lerninhaltes (engl. *Content Structure*), Inhaltskomponenten (engl. *Content Model*) und Metadaten, die definieren, wie die Komponenten des *Content Models* zusammenhängen. Das *Content Model* umfasst Komponenten (engl. *Assets*) wie Mediendateien, *Javascript*- und *HTML*-Dokumente, Inhaltsobjekte (engl. *Sharable Content Objects* – eine Sammlung von Komponenten, wovon eine der ausführbare Startpunkt ist) und Inhaltsstrukturen (engl. *Content Structures* – eine Struktur von Inhaltsobjekten, beispielsweise Kurse). Ein solches Lernmodul kann zum Transfer beispielsweise von *Authoring*-Werkzeugen in eine Datei gepackt werden. Diese kann dann von der SCORM-kompatiblen Lernplattform importiert werden.[17] Eine Übersicht über die Inhaltsaggregation findet sich in Abbildung 2.1.

Ihrer Spezifikation ist zu entnehmen, dass die SCORM-Standards zwar auf Verschachtelung bei Datenmodellen setzen, die Komplexität bei der Implementierung durch den beschränkten Funktionsumfang und die Modularität jedoch vergleichsweise gering ist. SCORM in der Version 1.2 wird von den meisten modernen Lernmanagementsystemen unterstützt und ist damit ein maßgeblicher Standard zur Inhaltsentwicklung, wenngleich die ADL für Neuentwicklungen und -anschaffungen sowohl von Lerninhalten als auch von LMS das Vorhandensein einer xAPI-Schnittstelle empfiehlt. Dennoch bietet sich Inhaltsdistributoren und -entwicklern durch die hohe Verbreitung von SCORM ein großer Absatzmarkt.

[2] Unterstützt werden „gestartet", „abgeschlossen" und „Fehler".

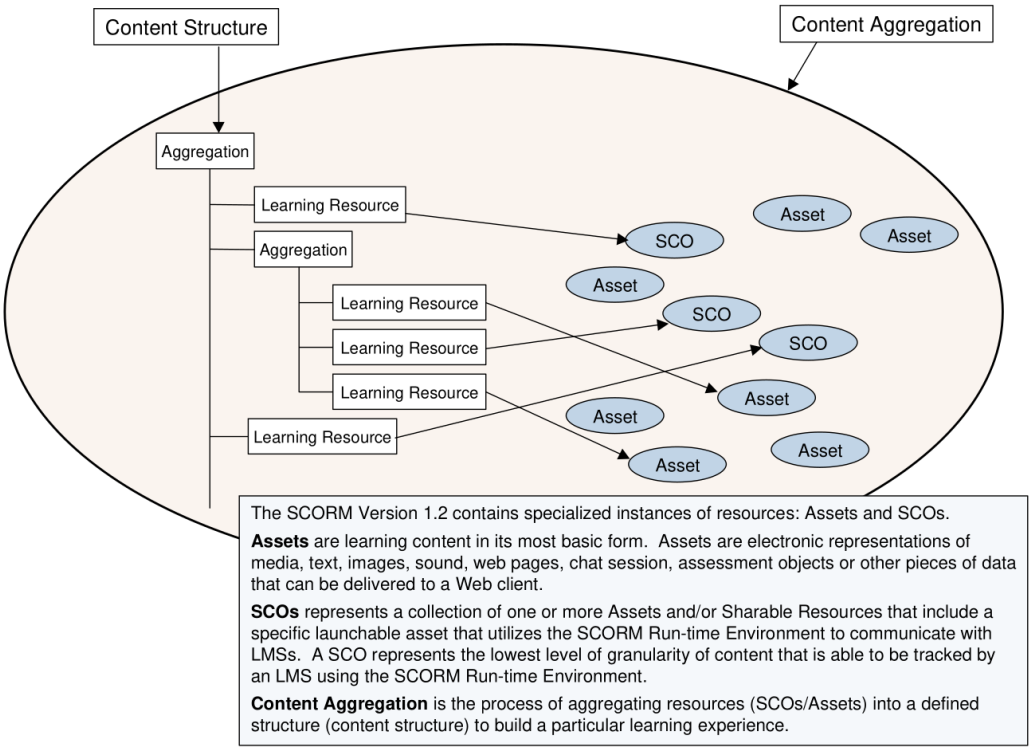


Abbildung 2.1.: Inhaltsaggregation in SCORM[17]

Doch warum bedarf es eines Nachfolgers für SCORM? Ein Fallbeispiel:

> Eine Universität betreibt seit mehreren Jahren eine Lernplattform *X*. Die Lehrenden haben viele Arbeitsstunden investiert, um hochqualitatives *E-Learning* zu ermöglichen, indem sie ausführliche Lehrinhalte erstellt und in Kursen strukturiert haben. Die Lernenden schreiben häufig über das in die Plattform integrierte Messaging-System untereinander und mit den Lehrenden. Das Lernmanagementsystem verfügt über einen Editor, in welchem derzeit neue Lerninhalte erarbeitet werden.
>
> Nun hat die Universität ein Angebot vorliegen, auf eine Lernplattform *Y* zu wechseln, um Geld einzusparen.
>
> Unter Verwendung von SCORM 1.2 können nur die publizierten Lehrinhalte und Kursstrukturen migriert werden. Möglicherweise gibt es weiterhin eine Funktion des LMS *Y*, die unfertigen Kurse aus dem Editor von *X* unter Verwendung von SCORM zu importieren – das ist jedoch nicht im Standard definiert. Jegliche Nutzerdaten (wie Nachrichten und Profile, insbesondere Kursfortschritte und Lernerfolge) sind nicht über SCORM migrierbar und gingen möglicherweise verloren.

Weiterhin können auf Basis der aus SCORM-Inhalten übermittelten Informationen nur bedingt Daten für Lernassistenzsysteme abgeleitet werden, wodurch deren Potenziale nicht genutzt werden können. Ebenfalls ist es mit SCORM nicht möglich, aktive[3] Lerninhalte bereitzustellen.

[3] Aktiv bezeichnet hier serverseitig gerenderte sowie vollständig interaktive Inhalte wie beispielsweise eine Jupyter-Einbindung.
HTML5-Inhalte mit einem eingeschränkten Interaktionsradius sind in diesem Sinne *passiv*.

2.3. Learning Tools Interoperability (LTI)

Gegeben sei das folgende (reale) Beispiel:

> An der Fakultät Physik der Technischen Universität Dresden sollte die Lehrveranstaltung „Programmierung" von Frontalunterricht mit begleitenden Übungsaufgaben auf *E-Learning* umgestellt werden. Hierfür wurden bestehende Lehrinhalte von PDFs in *Jupyter Notebooks*[4] migriert und mit interaktiven Code-Beispielen, die vom Server gerendert werden sollen, aufgewertet. (Offensichtlich ist dies mit SCORM-kompatiblen Lerninhalten nicht möglich.)
>
> Zum Starten und Verwalten der Nutzerumgebungen wird *JupyterHub* verwendet, eine Serveranwendung, welche ebenfalls die Nutzerauthentifikation implementiert.
>
> Nun sollten diese Lerninhalte in ein Lernmanagementsystem eingebunden werden.

Wir betrachten nun *JupyterHub* generalisiert als *Lern-Tool*. Unter Verwendung des Standards „Learning Tools Interoperability" (LTI), welcher vom *IMS Global Learning Consortium* entwickelt wurde, kann die oben beschriebene Aufgabe gelöst werden. In LTI wird unter anderem spezifiziert, wie in einem Kurs in einem Lernmanagementsystem ein externes *Lern-Tool* sicher eingebunden werden kann. Beim Start einer LTI-Anwendung werden Nutzerattribute wie Nutzername und E-Mail-Adresse (gegebenenfalls auch pseudonymisiert) auf Basis von *OpenID Connect* vom LMS an das *Lern-Tool* übertragen.[16]

Ein Rückkanal zur Übertragung von Ergebnissen (wie Punktzahlen oder Abschluss-Vermerke) aus dem *Lern-Tool* an die Lernplattform kann ebenfalls implementiert werden, ist jedoch nicht Teil der Basis-Spezifikation (*Core*) und damit nicht überall verfügbar.[15]

Während die *Core*-Spezifikation heute von vielen Lernmanagementsystemen implementiert wird, ist die Übertragung von Lernergebnissen häufig nicht vorgesehen. *Moodle* ist ein Beispiel für ein LMS, das eine vollständige LTI-(*Advantage*-)Schnittstelle anbietet[37]. *OPAL*, das sächsische Bildungsportal, verfügt nur über eine LTI-*Core*-Schnittstelle[13].

LTI ermöglicht keine Portabilität von Lehrinhalten, in einem größeren Umfang als SCORM aber deren Interoperabilität. Die separate Bereitstellung eines Servers für den jeweiligen LTI-Inhalt erhöht zwar den Administrationsaufwand für den Betrieb des E-Learnings, dafür sind die Einsatzszenarien *de facto* nur durch die Verfügbarkeit eines LTI-Adapters für die den Inhalt bereitstellende Anwendung eingeschränkt[5].

2.4. Experience API (xAPI)

Der nachfolgende Abschnitt bezieht sich vollständig auf das Spezifikationsdokument der xAPI ([26]).

Die Experience API (xAPI) ist eine Schnittstelle zum Austausch von Lernfortschritts- und Lernzustandsmeldungen, welche wie SCORM von der *Advanced Distributed Learning Initiative* spezifiziert wurde. Während die erste Version des Standards aus dem Jahr 2015 stammt, wird im Folgenden immer auf den aktuellen Stand (1.0.3) aus dem Jahr 2016 Bezug genommen. Laut ADL zielt die Entwicklung der xAPI darauf ab, unabhängig von Kontext, Plattformen und Technologien Lernerfahrungen und Lernergebnisse verständlich und vergleichbar zu

[4] `https://jupyter.org/` – Zugriff: 28.01.2022

[5] Hierbei handelt es sich um eine Pseudoeinschränkung, da ein LTI-Adapter frei und mit vergleichsweise wenig Aufwand implementiert werden kann – es handelt sich im einfachsten Falle lediglich um einen *OpenID Connect*-Consumer.

gestalten. Außerdem soll die Interoperabilität von Datenquellen (Lerninhalte und Lernmanagementsysteme), Datensenken (Learning Record Stores) und Datenverarbeitungsanwendungen (bspw. Lernassistenzsysteme oder Lernplattformen) bezüglich solcher Lerndaten hergestellt werden.

Der übliche Datenfluss sieht vor, dass der Lernende eine Lernerfahrung hat, welche von einem *Learning Record Provider* aufgezeichnet und in einem Learning Record Store (LRS) gespeichert wird. Ein *Learning Record Consumer* kann dann aus diesem LRS Daten zu jener Lernerfahrung abrufen.

2.4.1. xAPI-Statements

xAPI-Statements sind JSON-Dokumente und der Standarddatentyp in der Kommunikation mit Learning Record Stores.
Ein solches *Statement* folgt der Form

$$\underbrace{\text{I}}_{\text{Actor}}\ \underbrace{\text{did}}_{\text{Verb}}\ \underbrace{\text{that}}_{\text{Activity}}\ .$$

Bezogen auf die xAPI ist ein *Actor* ein oder mehrere *Agent(s)*. Ein *Agent* ist die Abbildung einer natürlichen Person, welche durch einen eindeutigen IRI identifiziert wird. *Verben* sind eindeutig identifizierte Interaktionstypen bezogen auf Inhalte. Ein Verbverzeichnis wird von der ADL unter `http://xapi.vocab.pub/` gepflegt. Das *Verb* bezieht sich immer auf ein *Object*, in der Regel eine *Activity*. Lerninhalte sind Beispiele für *Activities*. Optional können einem *Statement* Informationen über das Ergebnis einer Handlung (zum Beispiel eine Bewertung, eine erlernte Fähigkeit oder erlangte Kompetenz) oder den Kontext beigefügt werden. Jedem Statement wird entweder vom *Learning Record Provider* oder dem LRS ein (eindeutiger) UUID zugeordnet.

2.4.2. Learning Record Stores

Die zentrale Software im xAPI-Kontext ist der Learning Record Store, ein Server, welcher sogenannte *Learning Records* (im Weiteren auch als xAPI-Statements bezeichnet) annimmt, speichert und bereitstellt.[26] Der LRS wird in der Regel unabhängig von den lesenden und schreibenden Anwendungen betrieben, da die Spezifikation Universalität vorgibt. Außerdem sind diese Server in der Regel mandantenfähig[6].

Im einfachsten Fall lässt sich ein LRS als Dokumentenspeicher beschreiben. Die xAPI-Statements sind dann die Dokumente. Der Standard sieht vor, dass diese *Statements* unter anderem nach ihrer Identifikationsnummer (ID), ihrem Urheber (*Agent*), ihrem *Verb* oder dem verknüpften Lerninhalt (*Activity*) abgerufen werden können. Auch die Einschränkung auf den Datenabruf für einen gegebenen Zeitraum ist möglich.

Learning Record Stores müssen eine HTTP-Schnittstelle aufweisen, um dem Standard zu genügen. Wie üblich können mit den Methoden `POST` und `PUT` xAPI-Statements übermittelt werden; mit `GET` können diese wieder aus dem LRS abgerufen werden. Des Weiteren können Zustände sowie *Activity*- und *Agent*-Profile angelegt werden. Die Zustandsdaten müssen nur wenigen Ansprüchen genügen und können in der Regel beliebige JSON-Inhalte umfassen.

Agent-Profile können genutzt werden, um Lernende eindeutig zu identifizieren und verschiedene Lernendenkennungen zu unifizieren.[7] Dieser Prozess ist in Abbildung 2.2 visualisiert. Mit *Activity*-Profilen können arbiträre Informationen über Lerninhalte gespeichert und abgerufen werden.

[6] Auf einem Server können mehrere Speicher für xAPI-Statements mit unabhängigen Zugangsdaten existieren.
[7] Ein und derselbe Lernende kann von zwei unterschiedlichen Lernanwendungen verschiedene Kennungen erhalten; diese können so zusammengefasst werden.

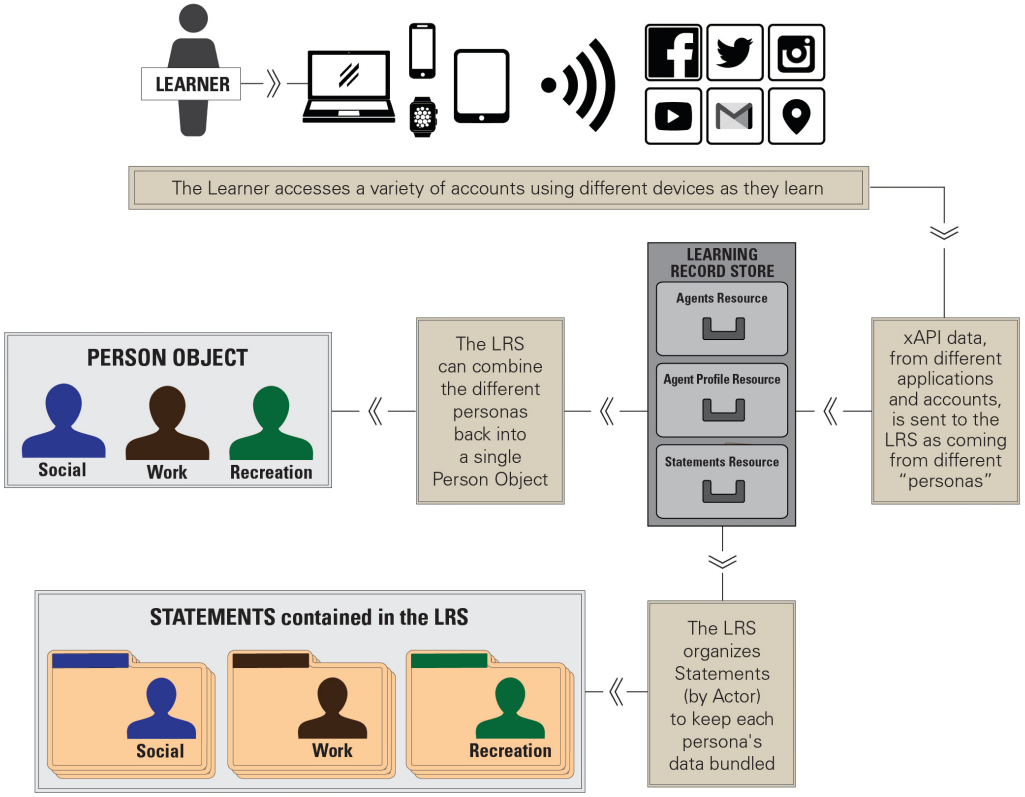


Abbildung 2.2.: Profilunifikation in Learning Record Stores.[26]

Da Learning Record Stores über HTTP kommunizieren, ist eine Transportverschlüsselung einfach zu erreichen. Zur Authentifizierung können je nach Software mindestens *OAuth 1.0*, *HTTP Basic Authentication* oder *Smartcards* verwendet werden. Weitere Sicherheitsfunktionen sind derzeit im Spezifizierungsprozess.

2.5. Representational State Transfer (REST)

Trotz der gegebenen Relevanz und Verbreitung von Programmier- und Datenschnittstellen auf Basis von Representational State Transfer (REST) gibt es in vielen Lernmanagementsystemen bisher keine oder nur rudimentäre Unterstützung für diesen Standard. So gab es 2020 in *OPAL*, dem sächsischen Bildungsportal, lediglich eine REST-Schnittstelle für den Abruf des Kataloges der Lehrveranstaltungen.[13]

Im quelloffenen LMS *ILIAS* gibt es gar keine offizielle REST-Implementierung, sondern lediglich ein von der Universität Marburg entwickeltes, nachinstallierbares Plugin, welches zuletzt 2018 gepflegt wurde.[24, 42]

Im Gegensatz dazu bietet das in Deutschland ebenfalls weit verbreitete LMS *Moodle* eine umfangreiche REST-Schnittstelle an, welche zum Abruf und Senden von Daten verwendet werden kann, sowie ein *XML-Remote-Procedure-Call-Interface* zum entfernten Auslösen von Aktionen auf der Plattform.[38] Das in Nordamerika am weitesten verbreitete Lernmanagementsystem[22] *CANVAS* verfügt ebenfalls über ein umfassendes REST-API.[14]

Insbesondere für die Integration von Lernassistenzsystemen dürfte das Vorhandensein einer generischen Datenschnittstelle über HTTP (wie beispielsweise REST) wesentlich an Bedeutung gewinnen. Aufgrund der durch das *Machine Learning* zur Modellbildung benötigten Rechenleistung werden LAS/TAS nicht im gleichen Server-Kontext betrieben werden wie das LMS, über welches ihre Ergebnisse angezeigt werden. Um dieser Komplikation gerecht zu werden, bedarf es eines Standards für diesen Datenaustausch. Durch die Generik des Representational State Transfers bietet sich die Kommunikation über diesen Standard an.

Hierbei ist es noch offen, ob die Datenschnittstelle für die Assistenzdaten vom Lernassistenzsystem (Pull-Prinzip) oder vom Lernmanagementsystem (Push-Prinzip) bereitgestellt werden sollte und welches Format die ausgetauschten Daten haben könnten. Eine Arbeit von taiwanesischen Forschern schlug 2010 vor, dass LAS/TAS die Schnittstelle implementieren sollten, welche dann von LMSn konsumiert wird[23]. Der Datenabruf ist gegenüber des kontinuierlichen Aktualisierens der Assistenzdaten dahingehend vorteilhaft, dass kein unnötiger Datentransfer stattfindet. Bedingt durch die möglicherweise sehr umfangreichen Berechnungen zur Klassifikation des Lernfortschrittes eines Lernenden kann es jedoch zu hohen Ladedauern beim Abruf der Assistenzdaten kommen.

Außerdem können über REST Lerninhalte mit Nutzdaten initialisiert und gestartet werden. Insbesondere ist der Start einer Lernanwendung mit LTI-Core durch eine REST-Schnittstelle gelöst.[16]

2.6. Modell zur Kommunikation in der Total Learning Architecture

Das folgende Konzept zur Kommunikation in der TLA ist in Abbildung 2.3 grafisch aufbereitet.

Nun sollen Lerninhalte all dieser Standards in der Total Learning Architecture harmonisiert eingesetzt werden. Experience API wurde von der ADL als Standard zum Austausch von Lernfortschritten und Zuständen entwickelt.

Um aus SCORM-implementierendem *E-Learning-Content* xAPI-Statements zu erhalten, muss das SCORM-*Runtime Environment* (beispielsweise im Kontext eines LMS oder KMS oder als eigenständige Anwendung) einen *Proxy* bereitstellen. Dieser lauscht auf vom SCORM-Inhalt versendete API-Aufrufe, rechnet den Inhalt in xAPI-Statements um und sendet diese an einen Learning Record Store. Mit dieser Methode können höchstens die Metadaten erfasst werden, die durch das Sharable Content Object Reference Model definiert wurden: Der Zustand der Inhaltsausführung (begonnen, beendet, fehlerhaft) und eine eventuelle Bewertung (vgl. Abschnitt 2.2).

In Lernplattformen wie *Moodle* oder *OPAL* werden Inhalte, die LTI implementieren, als *externes Tool* bezeichnet. Diese Tools werden von einer Anwendungskomponente, welche den Start (LTI *Core*) und eventuell die Ergebnisübermittlung (LTI *Advantage*) koordiniert, aufgerufen. Aus den Spezifika des Standards geht hervor, dass bei Implementierung von lediglich LTI *Core* (zum Beispiel in *OPAL*) nur Informationen über den Start eines LTI-Inhaltes erfasst werden können. Wird auch LTI *Advantage* implementiert (beispielsweise in *Moodle*), so können die Lernergebnisse ebenfalls entgegengenommen werden (vgl. Abschnitt 2.3). Die so erhaltenen Daten müssen dann ebenfalls vom LMS, beziehungsweise dessen *App-Manager*, in xAPI-Statements konvertiert werden und an einen LRS gesendet werden.

Ein LMS selbst kann in der Regel viele Metadaten über das Verhalten eines Lernenden sammeln. So kann die Navigation in Kursen und der Aufruf von internen und externen Kurselementen, die Interaktion mit nativen Lerninhalten und die darin erzielten Fortschritte, so-

wie die Bewertung von Kursen und Kurselementen getrackt werden. Diese Daten können ebenfalls von einer Lernplattform in xAPI-Statements umgerechnet und in einem Learning Record Store gespeichert werden.

Für *E-Learning-Content*, welcher bereits xAPI implementiert, muss ein LMS nur das Übermitteln des Start-Kommandos im *App-Manager* (wie bei LTI auch) tracken. Die weiteren Metadaten werden vom Inhalt selbst an den LRS gesendet und benötigte oder interessante Daten können vom Inhalt abgerufen werden. Hier sind die Tracking-Möglichkeiten *de facto* unbeschränkt, da xAPI-Inhalte vollständig unter der Kontrolle des Erstellenden und/oder Betreibenden sind (vgl. Abschnitt 2.4).

Schlussendlich können die auf diverse Art und Weise generierten xAPI-Statements von einem LMS, Lernanalysesystem oder zukünftig auch LAS/TAS abgerufen und weiter ausgewertet werden. Der Datentransfer von einem Lernassistenzsystem in ein Lernmanagementsystem und der Datenabruf aus einem Lernanalysesystem ist über eine REST-Schnittstelle denkbar.

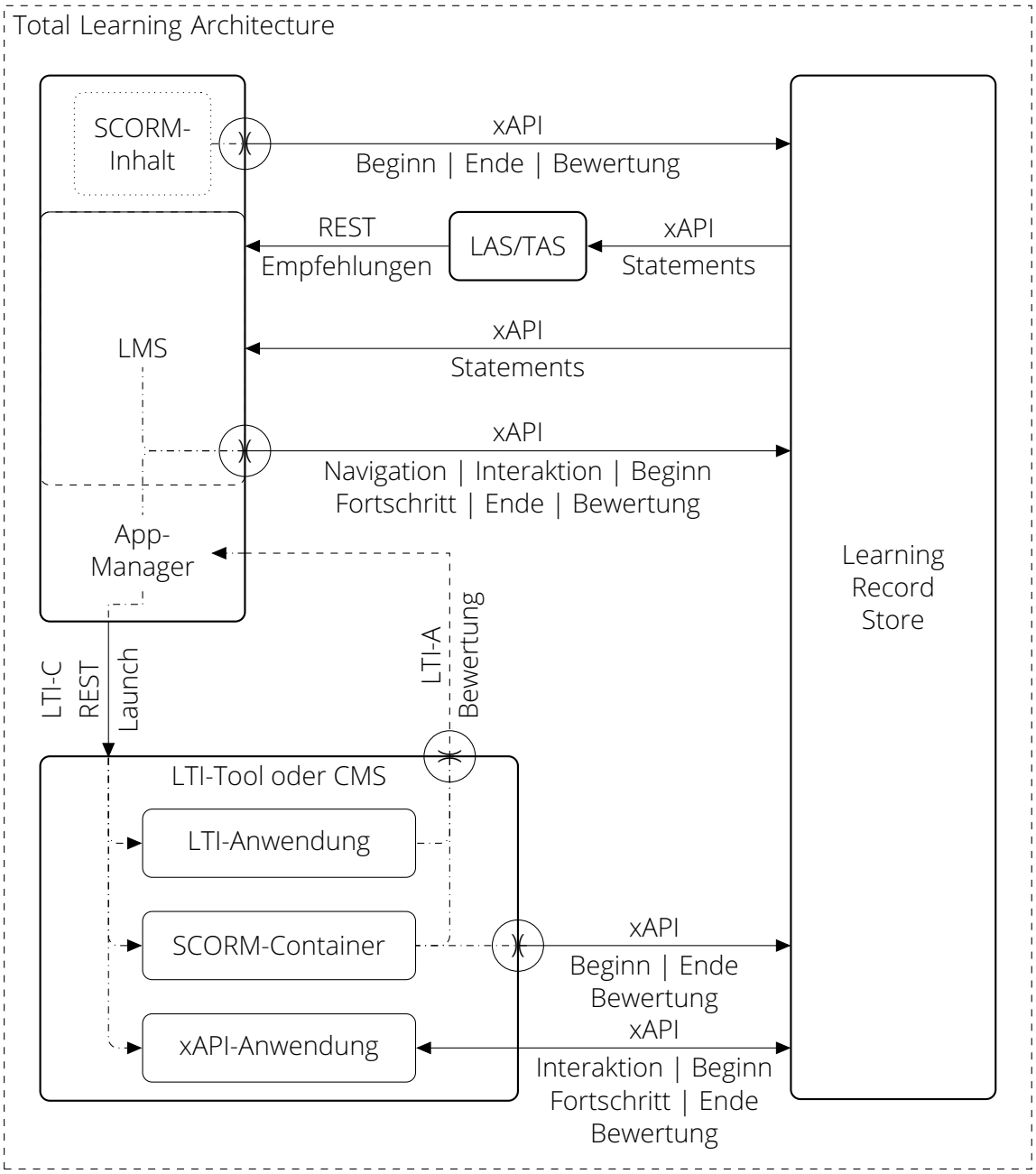


Abbildung 2.3.: Modernes Kommunikationsmodell in der Total Learning Architecture

3. Konzeption der Datengenerierungsanwendung

> The most important single aspect of software development is to be clear about what you are trying to build.
>
> Bjarne Stroustrup

Wie in Kapitel 2, insbesondere im Abschnitt zum Modell zur Kommunikation in der Total Learning Architecture, leicht ersichtlich ist, hat die Experience API das Potenzial, Lernergebnisse und -fortschritte aus diversen Lerninhalten zu erfassen, zusammenzuführen und analysierbar zu gestalten. Vor allem die Unspezifität des Standards erlaubt es sowohl Inhaltsproduzenten und -anbietern, Lernplattformentwicklern und -betreibern und damit auch deren Nutzern, ihre volle Kreativität bei der Ausgestaltung des Lernprozesses auszuschöpfen.

Eine Möglichkeit zur Auswertung der Daten, welche über xAPI gesammelt werden können, sind Lernassistenzsysteme und Lernanalysesysteme. Im Rahmen des Forschungsprojektes „VerDatAs“, in welches diese Arbeit eingebettet ist, soll ein solches LAS/TAS erforscht werden. Um die nötigen umfangreichen Daten zur Modellbildung in einem LRS speichern zu können, müssen diese zuerst generiert werden.

Da eine Datenerfassung für LAS/TAS nur mit xAPI-Statements sinnvoll ist (vgl. Kapitel 2), wird sich im Folgenden auf diese konzentriert. Manuelle Durchläufe von Lernprozessen sind zeitaufwändig und lassen auch *Edge Cases* außer Acht. Eine Anwendung zur Simulation solcher Lernaufzeichnungen gibt es mit DATASIM[1] bereits, diese ist jedoch per Kommandozeile oder REST-API zu bedienen. Dieses Tool wurde vom US-amerikanischen Softwareentwickler *Yet Analytics* im Auftrag der ADL entwickelt.[25]

Um die im Kontext des Forschungsprojektes notwendigen Daten zu generieren, soll also eine Anwendung entstehen, welche den Vorgang der Simulation auf einer auch für nichtversierte Nutzende verständlichen Web-Oberfläche abbildet.

3.1. Anforderungsanalyse

Die Anwendung soll laut Aufgabenstellung folgende funktionale Anforderungen erfüllen:

Anforderung 1 (Statementimport)**.** Ein Nutzender soll einen existierenden Satz von xAPI-Statements durch die Anwendung in einen LRS importieren können.

[1] `https://github.com/yetanalytics/datasim` – Zugriff: 28.01.2022

Anforderung 2 (Simulationsparametrisierung)**.** Ein Nutzender soll für eine Simulation mit DATASIM notwendige Parameter in der Anwendung festlegen und Basisdaten (*Seeds*) für die Simulation speichern können. Es soll keine softwareseitige Einschränkung für die Anzahl der speicherbaren Simulationsbasisdaten geben.

Anforderung 3 (Simulationsausführung)**.** Ein Nutzender soll aus vorher festgelegten Basisdaten eine Datensimulation mit DATASIM erzeugen können. Die Daten sind sowohl in maschinenlesbarer Form (als Liste von xAPI-Statements in einem JSON-Dokument) herunterladbar zur Verfügung zu stellen als auch direkt in einen LRS eintragbar.

Aus Anforderung 2 und Anforderung 3 folgt, dass Simulationen wiederholbar sind.

Anforderung 4 (Statementexport)**.** Es soll möglich sein, über die Anwendung einen Datenabzug des Learning Record Stores zu beziehen. Die Daten sind in maschinenlesbarer Form (als Liste von xAPI-Statements in einem JSON-Dokument) herunterladbar bereitzustellen.

Die xAPI-Spezifikation sieht das Löschen von xAPI-Statements nicht vor[26]. Somit könnte eine solche Funktion nur für eine spezielle LRS-Implementierung programmiert werden. Um die Interoperabilität nicht zu gefährden wird deshalb von einer solchen Funktion abgesehen.

Außerdem sind die nachstehenden nicht-funktionalen Anforderungen wie bewertet zu erfüllen:

Anforderung 5 (Zugänglichkeit)**.** Die Anwendung soll barrierefrei zu bedienen sein.

Anforderung 6 (Erweiterbarkeit)**.** Die Architektur der Anwendung soll stabil sein und zukünftigen Erweiterungen nicht im Wege stehen.

Anforderung 7 (Konfigurierbarkeit)**.** Die Anwendung soll die statische Konfiguration von Datenbank- und DATASIM-Verbindungen ermöglichen. Alle weiteren Parameter sind als dynamisch (also zur Laufzeit änderbar) zu betrachten.

Anforderung 8 (Dokumentation)**.** Für das Projekt ist ein Überblick über die Anwendungskomponenten und für die Nutzenden und Betreibenden eine Kurzanleitung zu erstellen.

Anforderung 9 (Fehlertoleranz)**.** Fehler in Nutzereingaben oder der Netzwerkinfrastruktur dürfen nicht zu einem Programmabsturz oder *Internal Server Errors* führen.

Anforderung 10 (Internationalisierung)**.** Die Anwendung soll in englischer Sprache programmiert werden. Die Nutzeroberfläche muss mindestens englischsprachig sein; eine deutsche Übersetzung ist erstrebenswert.

Anforderung 11 (Sicherheit)**.** Für dieses Projekt sind grundlegende Sicherheitsmechanismen zum Schutz vor Fremdzugriffen vorzusehen.

Anforderung 12 (Auslieferung)**.** Die entstehende Anwendung soll unabhängig von der Ausführungsumgebung sein.

Die Wichtung der Anforderungen in Tabelle 3.1 ergibt sich aus dem geplanten Anwendungsfall als internes oder lokal von Angehörigen der TU Dresden einzusetzendes Tool und wurde durch die Betreuenden dieser Arbeit bestätigt. Durch die Verfügbarkeit einer englischsprachigen Dokumentation und Nutzeroberfläche wird auch der Einsatz durch Personen, die der deutschen Sprache nicht mächtig sind, ermöglicht.

Tabelle 3.1.: Wichtung der nicht-funktionalen Anforderungen

Anforderung	++	+	o	–	––
Zugänglichkeit					×
Erweiterbarkeit	×				
Konfigurierbarkeit		×			
Dokumentation		×			
Fehlertoleranz	×				
Internationalisierung				×	
Sicherheit			×		
Auslieferung	×				

3.2. Evaluation bestehender Anwendungen und Ansätze

Es ist derzeit eine Anwendung bekannt, welche ein *Frontend* zu DATASIM bereitstellt. Dabei handelt es sich um *DATASIM-UI*[2], welches ebenfalls von *Yet Analytics* im Auftrag der ADL entwickelt wird[25]. Diese Anwendung wird beschrieben als „*JSFiddle*-ähnliche Erfahrung zur Erstellung, Bearbeitung, Ausführung und zum Teilen einer DATASIM Eingabedatei"[8]. Sie stellt hauptsächlich einen Editor zur Verfügung, mit der Voraussetzung, dass die Nutzenden exakt über die nötigen Handgriffe Bescheid wissen und insbesondere die semantischen und syntaktischen Feinheiten beherrschen müssen.

Bedingt durch die Aktualität der xAPI-Spezifikation ist nicht davon auszugehen, dass weitere Simulationstools zur Generierung von xAPI-Statements verfügbar sind. Eine Internetrecherche[3] sowie eine weitere Studienarbeit ([44]) bestätigen diesen Sachverhalt.

Weiterhin ist allgemein nur wenig vergleichbare Simulationssoftware auf dem Markt verfügbar. Untersuchungen von generischen Simulationsanwendungen, welche in der Regel mehr auf die Bedürfnisse von Ingenieuren abgestimmt sind (wie beispielsweise *Simulink*[4]), legen nahe, dass die Zusammenstellung einer Simulation per *Drag-and-Drop* industrieüblich ist. Des Weiteren gibt es Programme, insbesondere Kommandozeilen-basierte wie *Avida*[5] oder auch *GPSS*[6], bei welchen die Simulationsbeschreibung aus einer Textdatei gelesen wird. Zum Anlegen von wiederverwendbaren Entitäten wird in vielen Softwareprodukten eine Menübasierte Navigation angewendet.

Moderne serviceorientierte Softwarearchitekturen schlagen eine getrennte Entwicklung und Bereitstellung von *Backend* und *Frontend* vor. Dies wird unter anderem mit der ermöglichten parallelisierten Entwicklung, Sicherheitsaspekten und allgemein einer Trennung der Verantwortlichkeiten begründet.[33]

In der monolithischen Softwareentwicklung ist die Anwendung des *MVC-Patterns* weit verbreitet. Das Model-View-Controller-Entwurfsmuster sieht vor, dass bei einer Anfrage an den *Controller* dieser die angeforderten Aktionen ausführt, ein *Model* mit Daten populiert und diese in eine *View* rendert. Die Konzeption als Monolith hat den Vorteil, dass Erweiterungen des Funktionsumfangs der Software im Durchstich-Verfahren ermöglicht werden. Hierbei

[2] `https://github.com/yetanalytics/datasim-ui` – Zugriff: 28.01.2022
[3] Schlagworte: xapi simulation tool -datasim
[4] `https://de.mathworks.com/products/simulink.html` – Zugriff: 28.01.2022
[5] `https://sourceforge.net/projects/avida/` – Zugriff: 28.01.2022
[6] `https://en.wikipedia.org/wiki/GPSS` – Zugriff: 28.01.2022

wird die komplette Funktion als Einheit betrachtet und in *Frontend* und *Backend* in enger zeitlicher Konjunktion implementiert[21].

3.3. Entwicklungsaufgaben

Zunächst werden die formellen Voraussetzungen für die Anwendungsentwicklung und Integration von DATASIM festgestellt.

DATASIM wurde in *Clojure*[7] programmiert. Da *Clojure*-Anwendungen üblicherweise zur Nutzung in einer *Java Virtual Machine* kompiliert werden, können die beim Bauen der Anwendung entstehenden Artefakte auch als Programmbibliothek in beliebige Java-Anwendungen eingebunden werden. Um einer zukünftigen Integration von DATASIM als Programmbibliothek nicht im Wege zu stehen, wird folgende Entscheidung getroffen:

Festlegung 1 (Java)**.** Das *Backend* der zu erstellenden Anwendung wird in Java programmiert.

Die Simulationsanwendung DATASIM kann ebenfalls als singulärer oder verteilter Server betrieben werden, mit welchem über eine REST-Schnittstelle kommuniziert wird[7]. Um größere Datenmengen verarbeiten zu können, soll optional die Möglichkeit geschaffen werden, mit einem separaten DATASIM-*Cluster* kommunizieren zu können. Als Anforderung wird deshalb definiert:

Entwicklungsaufgabe 1 (DATASIM per REST)**.** Das Tool DATASIM ist über die vorhandene REST-API in die zu erstellende Anwendung einzubinden. Optional soll es in der zu erstellenden Anwendung die Möglichkeit geben, die Berechnungen zur Datensimulation auf externe DATASIM-Server auszulagern.

Die Konfiguration des DATASIM-Backendservers kann vorerst als instanzspezifisch und damit als statisch zu konfigurieren betrachtet werden (vgl. Festlegung 5). Dies ist bei der Bereitstellung zu berücksichtigen (vgl. Entwicklungsaufgabe 4).

3.3.1. Lösung der nicht-funktionalen Anforderungen

Während des Informatikstudiums an der Technischen Universität Dresden wird in der verpflichtenden Lehrveranstaltung „Softwaretechnologie-Projekt“ gegen eine Kombination aus *Spring Boot*[8] (Framework), *Spring MVC* (Webkomponenten), *Spring JPA* (Datenpersistenz), *Thymeleaf*[9] (Templating) und *Hibernate*[10] (Datenbank) entwickelt. Um Interoperabilität mit potenziellen zukünftigen Weiterentwickelnden zu gewährleisten (vgl. Anforderung 6), wird der Technikstack wie folgt gewählt:

Festlegung 2 (Anwendungsstack)**.** Die Anwendung soll aufbauend auf *Spring Boot* gegen *Spring MVC*, *Spring JPA* und *Thymeleaf* entwickelt werden. Als Datenbanksystem ist eine relationale Datenbank zu integrieren.

Aus diesem Grund wird auch eine Separation von *Frontend* und *Backend* (vgl. Abschnitt 3.2) vermieden und stattdessen ein Monolith unter Berücksichtigung des *MVC-Patterns* entwickelt.

[7] `https://clojure.org/` – Zugriff: 28.01.2022
[8] `https://spring.io/` – Zugriff: 28.01.2022
[9] `https://www.thymeleaf.org/` – Zugriff: 28.01.2022
[10] `https://hibernate.org/` – Zugriff: 28.01.2022

Zur Lösung von Anforderung 10 wird festgelegt:

Festlegung 3 (Sprache)**.** Alle Entwicklungsdateien sind in englischer Sprache zu verfassen. Die Benutzeroberfläche muss mindestens auf Englisch verfügbar sein.

Weiterhin wird bezüglich Anforderung 8 und Anforderung 5 die folgende Aufgabe formuliert:

Entwicklungsaufgabe 2 (Dokumentation)**.** Die Anwendungskomponenten und deren Zusammenhänge sind in einem kurzen Entwicklerhandbuch auf Englisch zu beschreiben. Für die Nutzenden der Anwendung ist ein kurzes Handbuch auf Deutsch oder Englisch zu verfassen.

Damit wird auch der Grundstein für Erweiterbarkeit gelegt.

Die Sicherheit im Anwendungskontext (Anforderung 11) soll mindestens durch Lösung folgender Aufgabe implementiert werden:

Entwicklungsaufgabe 3 (HTTP Basic Authentication)**.** Zur Nutzung der Anwendung ist eine statisch konfigurierbare Nutzername-Passwort-Kombination nach RFC7617 verpflichtend zu machen. Es ist keine Rechteverwaltung geplant.

Bedingt durch die planmäßige Einbindung als internes oder lokal ausführbares Tool sollte dieser Schutz vorerst genügen. Eine TLS-Terminierung ist wegen Entwicklungsaufgabe 4 über *Reverse Proxies* (wie beispielsweise *Trœfik*[11]) möglich. Die Konfiguration hierfür erfolgt separat und ist nicht Teil dieser Arbeit.

Um nicht zur Nutzendenverwirrung beizutragen, wird folgende Entwicklungsentscheidung bezüglich Fehlertoleranz gefällt:

Festlegung 4 (Fehlermanagement)**.** Für alle vorhersehbaren Ausführungsfehler ist ein entsprechendes Fehlermanagement vorzusehen. Insbesondere betrifft dies auch Kommunikationsfehler mit externen Komponenten.

Dies hat als Nebeneffekt, dass die Navigation in der Anwendung im Fehlerfall besser gesteuert werden kann.

Zur Konfiguration der Anwendung (vgl. Anforderung 7) gilt:

Festlegung 5 (Umgebungsvariablen)**.** Für alle statischen Anwendungsparameter (insbesondere zur Datenbankverbindung und Sicherheit) sind Umgebungsvariablen vorzusehen.

Schlussendlich soll die Auslieferung der Anwendung wie folgt durchgeführt werden:

Entwicklungsaufgabe 4 (Docker)**.** Die Anwendung soll in ein *Docker Image* gepackt werden. Die Zusammenstellung und Konfiguration einer Installation erfolgt durch ein *Docker Compose*-Projekt. Hierfür sind Beschreibungsdateien auszuformulieren.

Docker[12] ist ein Werkzeug zur Bereitstellung von Server- und Client-Applikationen. Anwendungen werden hier (meist) idempotent in durch ihre Beschreibungsdatei, das *Dockerfile*, definierte Abbilder gepackt. Diese enthalten die vollständige Ausführungsumgebung, sind leicht verteilbar und können bei ihrer Instanziierung durch Umgebungsvariablen parametrisiert werden (vgl. Festlegung 5).

[11] `https://www.traefik.io/traefik/` – Zugriff: 28.01.2022
[12] `https://www.docker.com/` – Zugriff: 28.01.2022

3.3.2. Lösung der funktionalen Anforderungen

Um die funktionalen Anforderungen 1 und 3 lösen zu können, muss eine Datensenke implementiert werden:

Entwicklungsaufgabe 5 (Senden von Statements)**.** Es muss eine Funktion zum Senden von xAPI-Statements geschrieben werden. Sie nimmt eine nicht-leere Liste von *Statements* an und sendet diese an einen variabel festlegbaren Learning Record Store.

Daraus folgt Entwicklungsaufgabe 10.

Für den Datenabruf (Anforderung 4) muss implementiert werden:

Entwicklungsaufgabe 6 (Auslesen von Statements)**.** Es muss eine Methode zum Auslesen von xAPI-Statements geschrieben werden. Das Modul nimmt einen Verweis auf einen Learning Record Store entgegen und hat als Rückgabewert eine (möglicherweise leere) Liste von xAPI-Statements.

Weiterhin erfordert Anforderung 1:

Entwicklungsaufgabe 7 (Import von Statements)**.** Nutzende des Systems müssen die Möglichkeit erhalten, eine Datei mit xAPI-Statements hochzuladen oder eine Liste von *Statements* in die Nutzeroberfläche hineinkopieren zu können. Über diesen Mechanismus müssen die *Statements* dann abgesendet werden können (vgl. Entwicklungsaufgabe 5).

Zur Lösung von Anforderung 2 gibt es nachstehenden Bedarf:

Entwicklungsaufgabe 8 (Parametrisierung von Simulationen)**.** Eine Möglichkeit, die durch DATASIM benötigten Parameter in die Anwendung einzutragen und diese von vorherigen Durchläufen abrufen zu können, muss geschaffen werden.

Aus diesen Daten folgt dann schlussendlich zur Lösung von Anforderung 3:

Entwicklungsaufgabe 9 (Durchführung von Simulationen)**.** Es muss eine Funktion implementiert werden, mit welcher aus einem Satz von Parametern eine DATASIM-Simulation gestartet wird. Die resultierenden xAPI-Statements sind zum Download sowie zur Eintragung in einen definierten Learning Record Store bereitzustellen. Weiterhin sind die Ergebnisse zusammen mit der Beschreibung der Simulation zu persistieren.

3.3.3. Datentypanalyse

> Programming Is About Code, But
> Programs Are About Data
>
> David Thomas
> Andrew Hunt
> in: The Pragmatic Programmer

Für den Verbindungsaufbau zu einem Learning Record Store werden Parameter benötigt. Diese umfassen laut Standard den Endpunkt, an welchen die *Statements* gesendet werden müssen, sowie ein Nutzername-Passwort-Paar (unter der Einschränkung, dass der LRS das *Basic Authentication*-Verfahren unterstützt)[26]. Die Verbindungsparameter sind dynamisch und damit zur Laufzeit festlegbar zu machen. Da diese Daten im Klartext (*Base64*-kodiert) übertragen werden müssen, können sie nicht vertraulich behandelt werden. Eine verschlüsselte Persistenz ist deshalb nicht zielführend.

Entwicklungsaufgabe 10 (Datenstruktur für LRS)**.** Die Datenstruktur für das Abspeichern von Verbindungsparametern für LRS muss angelegt werden. Sie umfasst Informationen über den Endpunkt, einen Benutzernamen und ein Passwort. Die Instanzen dieser Verbindungsparameter müssen durch einen Prozess über die Nutzeroberfläche angelegt werden können. Anlegen, Ändern und Löschen muss persistiert werden. In der Dokumentation ist auf die Datensicherheit hinzuweisen.

Zur Kommunikation mit DATASIM muss ein Beschreibungsobjekt für jede zu erstellende Simulation geschaffen werden. Eine Simulationsbeschreibung besteht gemäß Anleitungsdokument aus einer Menge von xAPI-Profilen (Schemata, welche die Menge der möglichen xAPI-Statements festlegen), einer Liste von Personae (Gruppen von xAPI-*Actors*, siehe Abschnitt 2.4), den Simulationsparametern (frühester Beginn der simulierten *Statements*, spätestes Ende, Anzahl der zu generierenden *Statements*, Zeitzone für Beginn und Ende und einem Startwert für die Simulation), sowie Alignments, welche die Interaktionswahrscheinlichkeiten von einer Persona mit einem Profilbestandteil abbilden[7].

Entwicklungsaufgabe 11 (Datenstruktur für DATASIM)**.** Eine Datenstruktur zur Abbildung der DATASIM-Simulationsbeschreibung muss angelegt werden. In dieser Entität müssen Personae, Alignments, Simulationsparameter und das zu verwendende xAPI-Profil miteinander verknüpft werden. Diese Datentypen sind (wegen transitiver Abhängigkeit) ebenfalls zu implementieren. Ihre Instanzen sollen über die Benutzeroberfläche verwaltet werden können und unveränderlich (bis auf Löschen) persistiert werden. Das Simulationsobjekt soll einen menschenlesbaren Bezeichner im Sinne eines Kommentars erhalten.

Die Personae sind außerdem simulationsübergreifend vorzuhalten und können auch außerhalb von Simulationen erstellt werden (vgl. Abschnitt 3.4).
Als optionale Aufgabe wird weiterhin definiert:

Entwicklungsaufgabe 12 (xAPI-Profile)**.** Es ist zu untersuchen, welches xAPI-Profil aus der DATASIM-Distribution den Eingabedaten aus dem Forschungsprojekt entspricht.

Zur Analyse des Datentyps „xAPI-Statement“ sei an dieser Stelle auf Unterabschnitt 2.4.1 verwiesen.

3.4. Skizze der Benutzeroberfläche

Die Benutzeroberfläche soll alle Interaktionen abdecken, die durch die Lösungen der funktionalen Anforderungen (siehe Unterabschnitt 3.3.2) vorgegeben sind.

Dabei soll gemäß der nicht-funktionalen Anforderung 5 (Zugänglichkeit) auf Nutzerfreundlichkeit und Intuitivität geachtet werden. Orientiert an bestehenden Anwendungen (vgl. Abschnitt 3.2) wird deshalb für die globale Sicht auf das System eine Menü-basierte Nutzeroberfläche implementiert (vgl. Abbildung 3.1). Über dieses Menü können Einstellungsseiten für unabhängige Entitäten (wie beispielsweise die Kommunikationsparameter für Learning Record Stores aus Entwicklungsaufgabe 10) erreicht werden, sowie *Workflows* gestartet werden.

Über den Inhaltscontainer werden alle Seiten des Workflows dargestellt. Um eine klare Struktur zu bieten werden dort bei Darstellung mehrerer Entitäten *Karten*[13] verwendet. Wird eine Abfrage dargestellt, sollen Formulare verwendet werden.
Workflows bilden die Erstellung komplexer Entitäten (hier der Simulationsbeschreibungen) ab. Auf die assistierte Komposition der Simulationsparameter wird zurückgegriffen, da ihre Bestandteile (vgl. Entwicklungsaufgabe 11) voneinander unabhängig und ausreichend komplex sind. Die Schritte zur Erstellung einer Simulation werden wie folgt angeordnet:

[13] `https://getbootstrap.com/docs/5.0/components/card/` – Zugriff: 28.01.2022

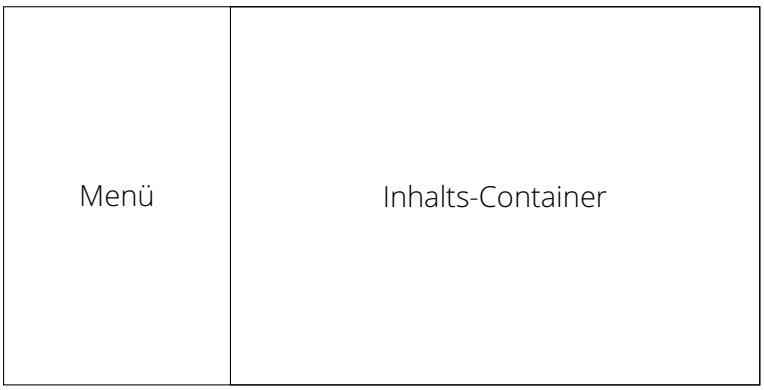


Abbildung 3.1.: Grundsätzlicher Aufbau der Nutzeroberfläche (nicht maßstabsgetreu)

1. Auswahl eines xAPI-Profils, welchem die generierten *Statements* genügen sollen
2. Erstellung und Auswahl von Personae für die Simulation
3. Definition von Alignments (Wichtungsfaktoren für die Interaktionshäufigkeiten)
4. Festlegung der weiteren Simulationsparameter
5. Durchführung der Simulation (wartend)
6. Bereitstellung des Simulationsergebnisses, optional Auswahl des LRS für *Upload*

Auch nach Abschluss einer Simulation sollen ihre Parameter und Ergebnisse verfügbar bleiben (wie in Entwicklungsaufgabe 11 festgelegt) und für einen erneuten Simulationslauf zur Verfügung stehen. Weiterhin soll es eine Möglichkeit geben, Simulationen zu projektieren.

Definition 1 (Projektierung)**.** Eine Simulation heißt projektiert, wenn ihre Parameter festgelegt wurden, aber die Simulation noch nicht berechnet wurde. Solange eine Simulation projektiert, aber noch nicht berechnet wurde, sind ihre Parameter veränderbar. Projektierte Simulationen sind grundsätzlich löschbar.

Definition 2 (Abschluss)**.** Eine Simulation heißt abgeschlossen, wenn sie berechnet wurde. Eine abgeschlossene Simulation kann aus dem System nur gemeinsam mit den generierten Simulationsdaten gelöscht werden. Ihre Parameter sind nicht veränderbar.

Die Menüstruktur kann damit wie folgt festgelegt werden:

- Workflows
  1. Simulationen *ausklappend wenn aktiv:*
     a) Schritte des Workflows …
  2. Statement Transfer
- Einstellungen
  - LRS-Verbindungen
  - DATASIM-Verbindung (mögliche Erweiterung)
- Dienstzustände
  - Liste aktiver externer Dienste mit Verbindungsstatus …

Die erste Navigationsebene soll immer ausgeklappt sein, um den Überblick über die zur Verfügung stehenden Funktionen zu wahren.

Die einzelnen Datenerfassungen und -darstellungen sollen durch (möglicherweise nur lesbare) Formulare realisiert werden. Bei der Navigation muss berücksichtigt werden, dass keine Daten verloren gehen.

Entwicklungsaufgabe 13 (Navigationswarnung)**.** Bei der Navigation in der Anwendung soll eine Warnung erscheinen, wenn nicht-gespeicherte Daten verloren gehen könnten und dem Nutzenden die Möglichkeit gegeben werden, diese zu speichern.

3.5. Validierungsplan

> "select" Isn't Broken
>
> David Thomas
> Andrew Hunt
> in: The Pragmatic Programmer

Es ist davon auszugehen, dass die Frameworks (*Spring*, *Hibernate*, *Thymeleaf*) und DATASIM selbst nicht getestet werden müssen, da diese nicht eigenständig implementiert wurden.

Da die Anwendung hauptsächlich externe Dienste integriert, sind primär Integrationstests zu entwickeln. Hierbei soll untersucht werden, ob beispielsweise die Kommunikation mit einem Learning Record Store korrekt funktioniert. Ebenfalls ist überprüfenswert, ob aus den *Workflow*-Eingaben im *Frontend* eine korrekte Simulationsbeschreibung entsteht. Automatisiert kann getestet werden, dass das *Handling* bei Erfolg oder Misserfolg einer an DATASIM gesendeten Simulationsaufgabe den Erwartungen entspricht. Auch die Erfüllung der Sicherheitsanforderungen kann mit Testfällen abgedeckt werden.

Alles in allem wird an dieser Stelle aus Zeitgründen auf extensive Test-Entwicklung verzichtet werden müssen, um die nicht-funktionalen Anforderungen so gut wie möglich implementieren zu können.

Festlegung 6 (Validierung)**.** Die Validierung der Anwendung erfolgt aus technischer Sicht nach den folgenden Kriterien:

1. Alle an externe Dienste ausgegebenen Datenstrukturen sind valide.
2. Es können maximal viele sinnvolle Parametrisierungen vorgenommen werden.
3. Kein Interaktionspfad führt zu einem ungültigen Anwendungs- oder Datenzustand.
4. Für Fehleingaben und dem Entwickler bekannte Interaktionsfehler mit externen Diensten werden sinnvolle Fehlermeldungen angezeigt.

Es wird nicht validiert, ob die externen Dienste ihre (Schnittstellen-)Spezifikation korrekt erfüllen.

Die Validierung der Nutzersicht erfolgt primär durch die Betreuenden dieser Arbeit.

3.6. Verworfene Konzepte

Nicht alle Gedanken des Autors führten zu einer integren Sicht auf die Gesamtanwendung. Einige der nicht umgesetzten Ideen werden im Folgenden diskutiert.

Verwendung von DATASIM als Programmbibliothek (*Library*)

Zu Beginn der Konzeption sollte die externe Anwendung DATASIM sowohl als Programmbibliothek, als auch als externer Dienst verwendet werden können. Dies ist prinzipiell möglich, da DATASIM in *Clojure* programmiert wurde und damit durch eine *Java Virtual Machine* ausführbar ist. Nach Betrachtung der Kompilate fiel jedoch die dynamische Typisierung von Parametern (alles ist vom Typ `java.util.Object`) und die mangelnde Dokumentation des Quelltextes schwer ins Gewicht, sodass dieser Ansatz verworfen werden musste.

Weiterhin würde durch die Ermöglichung der Integration als *Library* und als externe Anwendung unnötiger Projekt-*Overhead* entstehen. Wäre die Entscheidung unter diesem Gesichtspunkt zugunsten der Verwendung als Programmbibliothek gefallen, so wäre außerdem die Anwendung nur noch schwer zu skalieren gewesen. Die Skalierbarkeit ist in DATASIM durch den Betrieb als *Cluster* vorgesehen[7].

Oberflächenelemente

Für das Grundgerüst der Nutzeroberfläche sah das Konzept eine Titelleiste vor, wie in Abbildung 3.2 dargestellt.

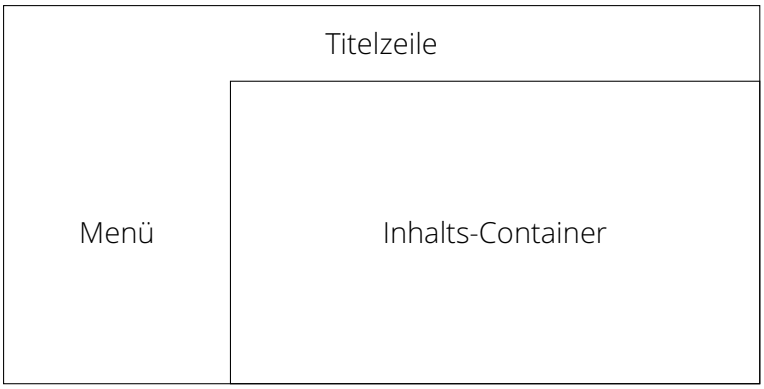


Abbildung 3.2.: Grundsätzlicher Aufbau der Nutzeroberfläche (nicht maßstabsgetreu)

Aufgrund der Beschränkung der Darstellungsgröße auf handelsüblichen Computerbildschirmen mit *FullHD*-Auflösung wurde von der Titelleiste abgesehen, da die dort vorgesehenen Informationen (Versionsnummer) entweder keinen Mehrwert für die Nutzenden bieten oder in der Seitenleiste (Dienstzustände, aktiver *Workflow*) untergebracht werden konnten.

Außerdem stellten erste Implementierungsversuche Daten in Tabellen dar. Durch die steigende Parameteranzahl war dies nicht mehr überall möglich, weshalb sich für das generalistische *Karten*-Schema[14] entschieden wurde.

[14] `https://getbootstrap.com/docs/5.0/components/card/` – Zugriff: 28.01.2022

Editierbarkeit von xAPI-Profilen

Die sogenannten xAPI-Profile sind für DATASIM eine zentrale Komponente zur Generierung von xAPI-Statements. Da im Kontext der Forschungsgruppe „VerDatAs“ *H5P* zur Generierung von *Statements* eingesetzt wird und H5P wiederrum *Statements* aus dem cmi5-Profil erzeugt und ein solches Profil mit DATASIM ausgeliefert wird, wurde diese Anforderung schlussendlich verworfen. Stattdessen werden alle mit DATASIM ausgelieferten Profile in der Anwendung zur Auswahl zur Verfügung gestellt.

4. Implementierung der Anwendung

Anhand der Entscheidungen und Entwicklungsaufgaben aus dem Konzept wurde die Implementierung durchgeführt. Hierbei wurden zuerst abhängige und unabhängige Aufgaben differenziert und die abhängigen Aufgaben dann in dem in Abbildung 4.1 dargestellten Abhängigkeitsbaum sortiert.

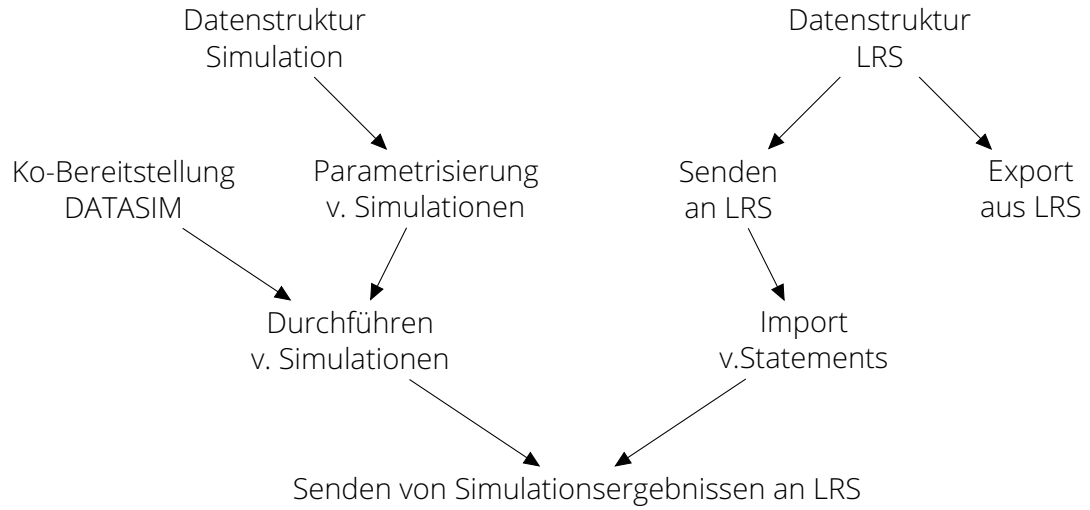


Abbildung 4.1.: Abhängige Entwicklungsaufgaben

Unabhängige Entwicklungsaufgaben umfassen unter anderem:

- Entwicklungsaufgabe 3 (HTTP Basic Authentication)
- Entwicklungsaufgabe 4 (Docker)

Diese wurden zuerst implementiert, um die Anwendung schneller in einen testbaren Zustand versetzen zu können. Danach wurden die Datenstrukturen für LRS (Entwicklungsaufgabe 10) und DATASIM (Entwicklungsaufgabe 11) implementiert. Nach der Implementierung der Parametererfassung (Entwicklungsaufgabe 8) wurde das Senden von Simulationsdaten an DATASIM erprobt (Entwicklungsaufgabe 9) und dann die Kommunikation mit einem Learning Record Store implementiert und getestet. Schlussendlich wurde noch der Export aller Daten aus einem LRS (Entwicklungsaufgabe 6) ermöglicht.

4.1. Methodik und Implementierungsdetails

> Build End-to-End, Not Top-Down or Bottom Up
>
> David Thomas
> Andrew Hunt
> in: The Pragmatic Programmer

Die Anwendung wurde im Durchstich-Verfahren entwickelt, das heißt, dass für jede Funktionseinheit die Nutzeroberfläche und deren zugehöriges *Backend* gleichzeitig implementiert wurden. Nachdem das Grundgerüst der Anwendungsabhängigkeiten (wie in Festlegung 2 konstatiert) über den *Spring Initializr*[1] importiert wurde, konnten die Infrastrukturaufgaben (insbesondere *Continous Integration*) durch eine *Pipeline*, welche automatisiert Tests ausführt, ein *Docker Image* baut und dieses in eine *Docker Registry* lädt, gelöst werden. *Spring Boot* verfügt über eine Funktion (im *Maven*[2]-Sprech auch *Target*), welches ein *Docker Image* unter Einbezug der *Best Practices* bezüglich Sicherheit und Performance erstellen kann. Diese wurde in hier genutzt. Weiterhin wurde *Maven* so konfiguriert, dass verschiedene Profile zur Anwendungsausführung in Abhängigkeit vom Ausführungskontext genutzt werden können. Diese Profile steuern unter anderem die Datenpersistenz (*In-Memory* für die Entwicklung, relationale Datenbank im produktiven Einsatz), das *Seeding* (das Anlegen von Beispieldaten) und den Zugriff auf Entwicklungstools wie beispielsweise die Datenbankkonsole. Außerdem können so Standardwerte für Test-Sitzungen übergeben werden.

Begonnen wurde die Implementierungsphase mit der Persistenz von Verbindungsparametern für Learning Record Stores, da deren Komplexität sehr gering ist:

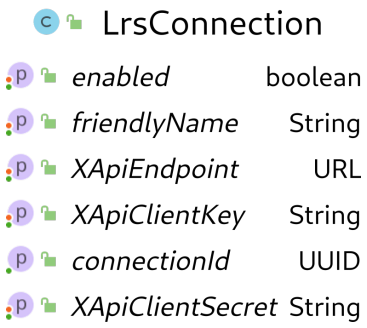


Abbildung 4.2.: Klassendiagramm für LRS-Verbindungsparameter

Über die Anforderungen aus Entwicklungsaufgabe 10 hinaus wurden ein Identifikator für die Persistenzschicht und ein Zustandsflag eingefügt, welches anzeigt, ob eine Verbindung zum LRS möglich sein soll. Zur Entkopplung der Nutzerschicht von der Persistenzschicht wurde ein Transferobjekt mit integriertem *Mapper*, wie in Abbildung 4.3 gezeigt, implementiert.

Danach wurde (im Durchstich-Verfahren) direkt die nötige (nicht-finale) Benutzeroberfläche entwickelt. Hierbei wurde die folgende Entscheidung getroffen:

Festlegung 7 (Bootstrap)**.** Die Benutzeroberfläche wird mit *Bootstrap 5*[3] gebaut.

Bootstrap ist ein freies *HTML5-CSS3-Javascript-Framework* zur Entwicklung von Benutzeroberflächen.

[1] `https://start.spring.io` – Zugriff: 28.01.2022
[2] Java Build Tool, `https://maven.apache.org/` – Zugriff: 28.01.2022
[3] `https://getbootstrap.com/docs/5.1/getting-started/introduction/` – Zugriff: 29.01.2022

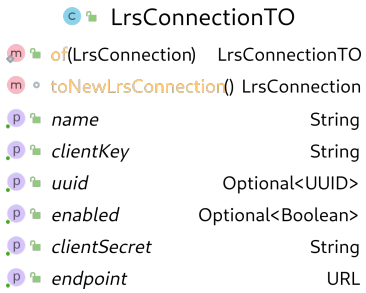


Abbildung 4.3.: Klassendiagramm für das LRS-Verbindungsparameter-Transferobjekt. Die Mapper-Methoden sind hervorgehoben.

Im Rahmen des Aufbaus der Ordnerstruktur wurde ein Basis-*Template* für *Thymeleaf* (vgl. Festlegung 2) angelegt. Zur Reduktion von *Overhead* beim *Templating* der Nutzeroberfläche wurde das *HTML*-`head`-Tag parametrisiert ausgelagert, sodass dieses als Grundlage für alle Seiten der Benutzeroberfläche eingebunden werden kann.

Im Anschluss daran wurde die Grundlage für das zu erreichende Ziel gebaut: Ein Verbinder (engl. *Connector*, eine Variante des *Glue-Patterns*) für DATASIM (dargestellt in Abbildung 4.4). Ziel dieses Konstrukts ist es, die Verbindung zu DATASIM über HTTP (vgl. Entwicklungsaufgabe 1) zu abstrahieren und *Health Checks* bereitzustellen. Die Verbindungsparameter werden durch eine Annotation (`@Value`) zur Laufzeit der Software auf Basis von Kontext-spezifischen Parametern bezogen. Im Regelfall entstammen sie den Umgebungsvariablen `XAPITOOLS_SIM_BACKEND_BASE_URL` (Host-Adresse), `XAPITOOLS_SIM_BACKEND_USERNAME` und `XAPITOOLS_SIM_BACKEND_PASSWORD` (Zugangsdaten nach RFC7617). Der Konnektor unterstützt neben dem Abruf von Verfügbarkeitsinformationen das Senden von Simulationsbeschreibungen (`DatasimSimulationTO`) im Austausch für eine Liste von xAPI-Statements, welche generisch als `JsonNode` typisiert wurden.

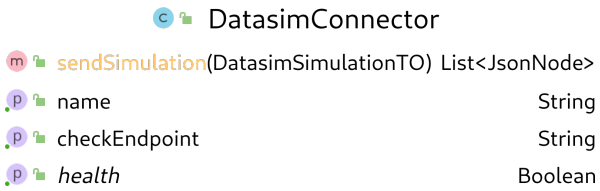


Abbildung 4.4.: Klassendiagramm für den DATASIM-Konnektor.

Bei der Entwicklung des Konnektors fiel auf, dass DATASIM kein gültiges JSON-Dokument als Rückgabewert übermittelt. Obwohl die Struktur einer Liste ähnelt (die *Delimiter* sind eckige Klammern), werden die Einträge der Liste nur durch einen Zeilenumbruch getrennt, nicht – wie es der Standard RFC8259, Section 5 vorsieht – durch Kommata. Hierfür musste ein manueller *Workaround* in der Methode `DatasimConnector.sendSimulation` implementiert werden, welcher bei einer Behebung des Fehlers in DATASIM zurückgenommen werden muss.

Bevor der Konnektor fertiggestellt wurde, musste die Datenstruktur für die Simulationsbeschreibung gemäß der Vorgaben aus dem kurzen Nutzerleitfaden von DATASIM (*README*, siehe Entwicklungsaufgabe 11) abgebildet werden. Hierbei wurde die erwartete Struktur seitens des Herstellers für die Persistenzschicht nur geringfügig adaptiert (vgl. Abbildung 4.5):

- Personae gehören immer zu mindestens (persistenzspezifisch; nach Implementierung

genau) einer Gruppe. Dies wurde so umgesetzt, da Alignments laut Dokumentation auch auf Gruppenebene konstruiert werden können. Aktuell ist dies jedoch in der entwickelten Anwendung nicht vorgesehen. Die Zugehörigkeit zu einer Gruppe wurde dennoch als sinnvolle Erweiterung angesehen und deshalb mit modelliert.

- Profile werden nicht als Dokument in der Datenbank abgespeichert, sondern als Verweise auf Dateien. Hierfür wurde ein *Seeder* entwickelt, welcher den Klassenpfad unter `/xapi/profiles` nach gültigen Profildokumenten durchsucht und die entsprechenden Verweise in die Datenbank speichert. Dadurch können neue Profile mit geringem Aufwand einkompiliert werden. Ein Nachteil dieser Implementierungsvariante ist, dass Profile nicht editierbar sind. Bei einer zukünftigen dahingehenden Weiterentwicklung müssten jedoch nur die betreffende Persistenzeinheit (`DatasimProfile`) und das zugehörige Transferobjekt angepasst werden. Dass DATASIM die Eingabe mehrerer Profile erlaubt, wurde ebenfalls bereits berücksichtigt, jedoch in der Benutzeroberfläche nicht verfügbar gemacht, um die für Nutzende wahrnehmbare Komplexität zu verringern.

- Für alle Persistenzeinheiten wurden Identifikatoren hinzugefügt. Zur Umsetzung von Definition 2 (Abschluss) wurde die Hauptklasse (`DatasimSimulation`) um ein *Flag* erweitert.

- Alignments (Zuneigungen und Abneigungen bezüglich Profilelementen) werden als Wertepaare persistiert. Die Umwandlung zu der nativen DATASIM-Struktur erfolgt im zugehörigen Transferobjekt.

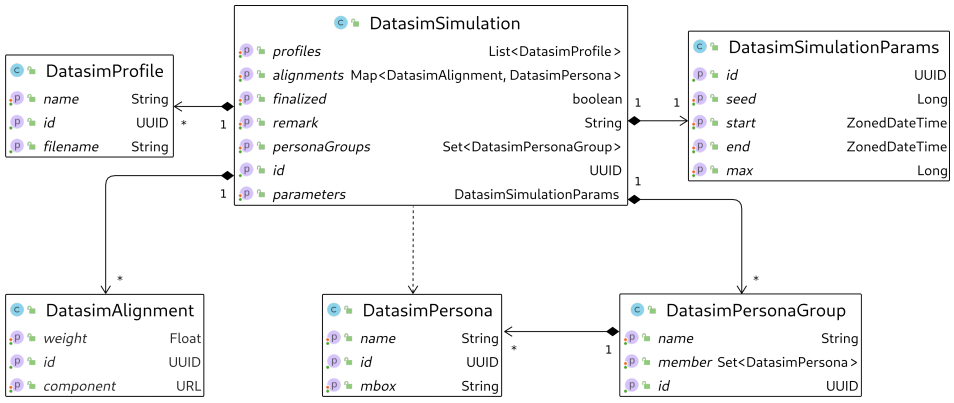


Abbildung 4.5.: Klassendiagramm für die Persistenz von Simulationsbeschreibungen

Im Gegensatz zur xAPI-Spezifikation sind im kurzen Nutzerleitfaden zu DATASIM nur E-Mail-Adressen als Identifikatoren beschrieben. Deshalb wurde in der Anwendung auch nur ein solches Identifikationsmuster umgesetzt. Da beim Testen auffiel, dass es aufwändig ist, sich E-Mail-Adressen auszudenken, wurde ein Zufallsgenerator für solche Eingaben in die Benutzeroberfläche integriert.

Um nun die Persistenzeinheiten in eine valide DATASIM-Simulationsbeschreibung (vgl. *README* in [7]) umzuwandeln (und die Entkopplung und Anwendungssicherheit zu verbessen) wurden Transferklassen angelegt. Durch *Factory*-Methoden und *Mapper* können die Datentypen passend konvertiert, sowie mit Standardwerten versehene Instanzen erstellt werden. Ein Beispiel hierfür ist die statische Methode `DatasimPersonaGroupTO.empty(String)`, welche eine leere Gruppe von Personae anlegt und mit einem Kommentar benennt. Außerdem

implementieren die Transferobjekte jeweils eine Methode `forExport()`, welche das jeweilige Transferobjekt für den Transfer an DATASIM vorbereitet. Dabei werden rekursiv nicht in der Datenspezifikation von DATASIM erfasste Felder entfernt, ermöglicht durch die Verwendung von `Optional` als Datentyp in Konjunktion mit korrekten *Jackson*-Annotationen. Die Serialisierung des Profils kann durch eine *Jackson*-Annotation sogar so erfolgen, dass das Transferobjekt sein zugrundeliegendes Profildokument als Repräsentation annimmt. Weitere Serialisierungsaspekte können unter anderem durch die geschickte Wahl von Feldnamen und Feldtypen gelöst werden.

Quelltext 4.1: Transferobjekt-Aufbereitung für DATASIM

```
public class DatasimSimulationTO {
    // Entferne Feld, wenn das Optional leer ist
    @JsonInclude(JsonInclude.Include.NON_ABSENT)
    private Optional<UUID> id;
    ...
    // Benenne das Feld um
    @JsonProperty("personae-array")
    private Set<DatasimPersonaGroupTO> personaGroups;
    ...
    // Leert rekursiv die Felder, die nicht vorhanden sein sollen
    // Die Serialisierung muss auf dem Wert dieser Methode erfolgen.
    public DatasimSimulationTO forExport() {
        return new DatasimSimulationTO(
                // Leeren
                Optional.empty(),
                ...
                // Rekursion
                this.parameters.forExport(),
                // Profile werden nicht angepasst,
                // da sie In-Place serialisiert werden.
                this.profiles,
                ...
        )
    }
}
public class DatasimProfileTO {
    // Der Serialisierungswert dieser Klasse entspricht dem Wert dieser
       Methode.
    @JsonValue
    public JsonNode getProfileContent() {
        ...
        // Gebe den Dateiinhalt aus
        return objectMapper.readTree(
            new ClassPathResource("xapi/profiles/" + this.filename)
                .getFile()
            );
    }
}
```

Um ungültige Datenzustände zu vermeiden, wurden Validatoren entwickelt. Diese überprüfen in der Verarbeitungsebene (den *Services*), ob Entitäten gültige Zustände haben. Beim

Handling von Verbindungen zu Learning Record Stores beispielsweise wird überprüft, ob diese aktiv sind. Simulationsbeschreibungen sind so nur bearbeitbar, wenn sie nicht als finalisiert markiert wurden; respektive sind sie nur an DATASIM versendbar, wenn dieses *Flag* den booleschen Wert `Wahr` aufweist.

An dieser Stelle soll auch auf eine Besonderheit in der Benutzerführung eingegangen werden. Ohne dass dies offensichtlich dokumentiert wurde, ist der Wertebereich für Alignments in DATASIM derzeit sehr eingeschränkt[1]. Damit keine ungültigen Werte ins System gelangen, wurde ein Parser entwickelt, welcher valide Passungskomponenten[4] aus dem Simulationsprofil extrahieren kann. Aus diesen kann dann eine Menge von Alignments erstellt werden. Da diese mit dem Ziel der Aufwandsreduktion für jeden Nutzer beim Hinzufügen neutral (das heißt mit einem Gewicht von 0) angelegt werden, entsteht so sehr schnell eine hohe Anzahl von konfigurierbaren Parametern. Zur Erhöhung der Übersichtlichkeit werden diese deshalb in einem sogenannten *Accordion*[5] nach Komponenten gruppiert dargestellt (siehe Abbildung 4.6). Im Körper dieses *Accordions* befinden sich dann Schieberegler, mit welchen die Affinität einer Persona zur umschließenden Komponente im gültigen Wertebereich $\mathbb{R}_{-1 \leq x \leq 1}$ gewichtet werden kann.

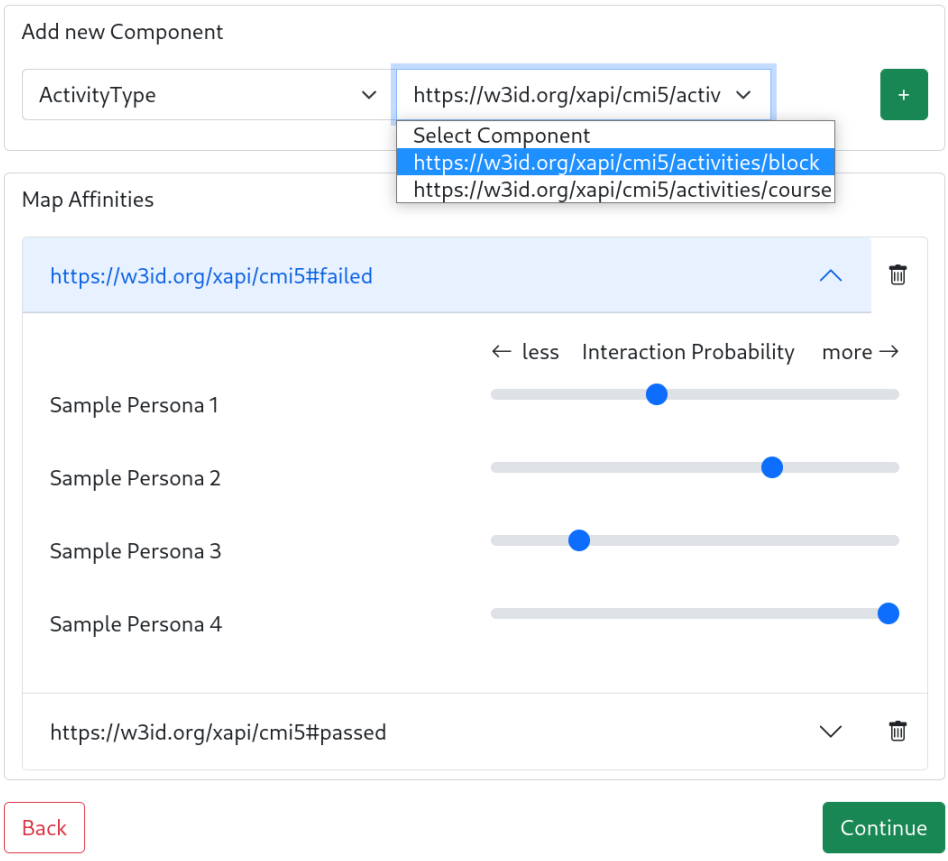


Abbildung 4.6.: Nutzerführung zur Definition von Alignments

Nachdem nun Simulationen für DATASIM parametrisiert und serialisiert wurden, und der zugehörige Konnektor diese im Austausch für generierte xAPI-Statements versenden kann, soll eine Funktion geschaffen werden, um das Simulationsergebnis in einen Learning Record

[4] Derzeit: *Pattern*, *ActivityType* und *StatementTemplate*
[5] `https://getbootstrap.com/docs/5.1/components/accordion/` – Zugriff: 29.01.2022

Store zu laden. Im Rahmen dessen wurde ein Konnektor für die Kommunikation mit einem Learning Record Store implementiert. Dieser stellt ebenfalls *Health Checks* bereit.

Nun soll die Anwendung nicht nur mit einem einzigen LRS interagieren können. Wegen der regelmäßigen *Health Checks* muss der Konnektor aber über einen Lebenszyklus verfügen. Im Vergleich zum `DatasimConnector`, von welchem genau eine Instanz zur Laufzeit des Programms von *Spring* gebunden wird, entsteht hier das Problem, dass der Konnektor für LRS durch Verbindungsdetails parametrisierbar sein muss, und von diesen Verbindungsdetails können mehrere zeitgleich im System existieren (vgl. Entwicklungsaufgabe 10). Deshalb wurde für die Klasse `LrsConnector` ein *LifecycleManager* implementiert, welches die dynamische Erzeugung und Destruktion von Objekten dieser Klasse ermöglicht. Das funktioniert durch die parametrisierte Instanziierung des Konnektors zu einer Spring Bean[6] (vgl. Abbildung 4.7). Die Existenz eines solchen Konnektors wird (entgegen dem *Design Pattern* „Inversion of Control", welches bei *Spring* üblich ist) durch einen *Service* in Abhängigkeit des aktiviert-Zustandes einer `LrsConnection` gesteuert.

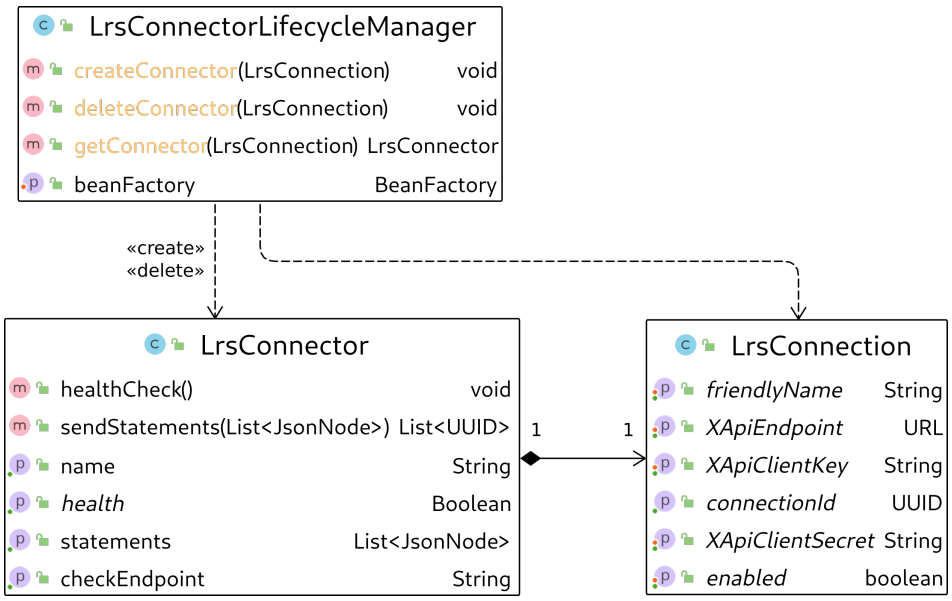


Abbildung 4.7.: Steuerung des Lebenszyklus von LRS-Konnektoren

Es stellt sich sicherlich die Frage, ob diese Komplexität gerechtfertigt ist. Zusätzlich zur Konzeption wurde ermöglicht, *Health Checks* für externe Dienste durchzuführen, wofür die dynamische Erzeugung und Destruktion von Konnektoren erfordert wird. Die Gesundheitszustände sollten auch den Nutzenden der Anwendung angezeigt werden. Durch die Instanziierung je einer Konnektor-*Bean* pro externem *Service* (also auch für die Verbindung zu DATASIM) und durch *Springs ApplicationContext* können dann alle Konnektoren im zugehörigen *Controller* für die weitere Nutzung gesammelt werden. Hierfür muss lediglich ein gemeinsames *Interface*, hier `IExternalService`, implementiert werden (vgl. Abbildung 4.8). Da diese Konnektoren zur Laufzeit instanziierbar sind, muss die Abfrage aus dem Anwendungskontext ebenfalls zur Laufzeit erfolgen. Im Gegensatz dazu können nicht-laufzeitabhängige Komponenten (hier: `IUIFlow` und `IUIManagementFlow` – dazu folgend mehr) direkt durch *Springs Inversion of Control*-Mechanismen gebunden werden.

[6] `https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#spring-core` – Zugriff: 29.01.2022

Quelltext 4.2: Dynamisches Binden von Anwendungskomponenten

```
@Controller
public class BasepageMavController {
    // Statisch durch IOC gebunden
    private final List<IUIFlow> flows;
    private final List<IUIManagementFlow> managementFlows;
    // Erlaubt direkte Suche nach Beans
    private final ApplicationContext context;
    ...
    @GetMapping("")
    public ModelAndView showHome() {
        // Binden von externen Diensten zur Laufzeit
        List<IExternalService> externalServices =
            this.context.getBeansOfType(IExternalService.class).values()
                .stream()
                // Aufbereitung zur Anzeige
                .sorted(Comparator.comparing(IExternalService::getName))
                .toList();
        ...
    }
}
```

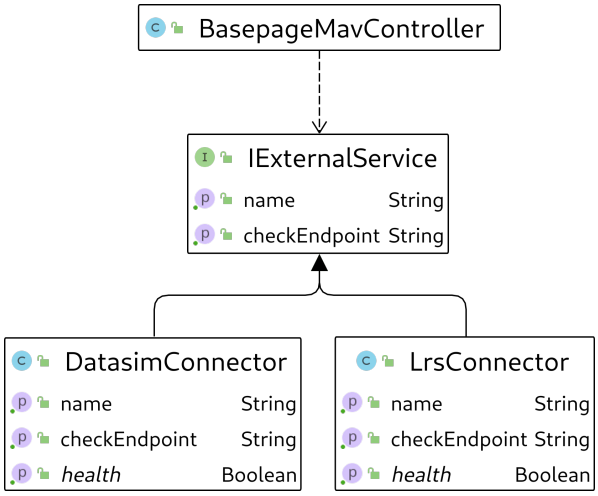


Abbildung 4.8.: Alle Konnektoren implementieren ein gemeinsames Interface

Da nun die externen Dienste erfolgreich auf der Nutzeroberfläche angezeigt werden, soll darauf eingegangen werden, wie die anderen Elemente zur Anzeige gelangen. Der einfachste Weg wäre, die Anzeigeelemente (*Links*) statisch im *HTML*-Quelltext des Menüs (vgl. Abbildung 3.1) zu hinterlegen. Unter Betrachtung von Anforderung 6 (Erweiterbarkeit) mit der Kenntnis über das *Inversion-of-Control-Pattern* wäre es jedoch wünschenswert, wenn bei einer Erweiterung der Anwendung an der Basis-Nutzeroberfläche nichts geändert werden muss. Aus diesem Grund wurden die *Interfaces* (vgl. Abbildung 4.9) `IUIFlow` für Unteranwendungen und `IUIManagementFlow` für Einstellungen entworfen. Da Unteranwendungen Teilschritte aufweisen können, welche auf der Nutzeroberfläche dargestellt werden sollen, wurde zusätzlich das *Interface* `IUIStep` hinzugefügt. Die Hervorhebung des aktuell aktiven Schrittes erfolgt hierbei durch die Überprüfung, ob der Pfad der aktuell angezeigten Seite dem regulären Ausdruck `IUIStep.getPathRegex()` genügt.

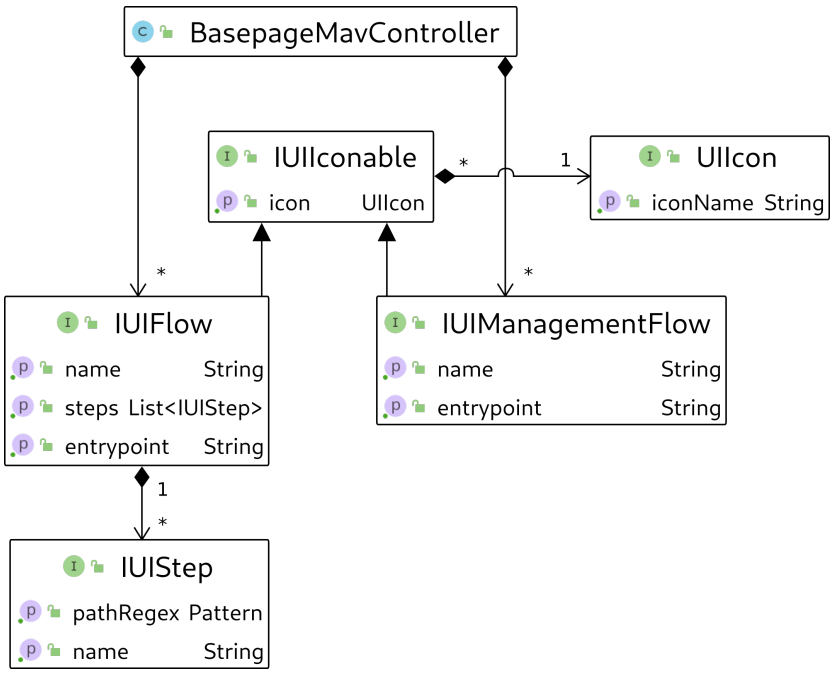


Abbildung 4.9.: Interfaces zum Binden der Benutzeroberflächenelemente

Durch die Implementierung von `UIIcon` können *Framework*-Spezifika (wie beispielsweise das Präfix `bi-` für *Bootstrap Icons*[7]) abstrahiert werden (vgl. Quelltext 4.3). Auf diese Art und Weise kann auch eine gewisse Erweiterbarkeit, beispielsweise zur Integration von weiteren *Icon-Frameworks*, sichergestellt werden. Möchte man beispielsweise die *Icons* von *Font Awesome*[8] verwenden, so kann ein weiteres *Enum* mit dieser Zuständigkeit `UIIcon` implementieren. Eine mögliche Erweiterung wäre es, die Verweise auf die benötigten *CSS*-Dateien im implementierenden *Enum* zu hinterlegen, dieses ebenfalls im `BasepageMavController` zu binden und so den `head`-Tag der *HTML*-Dateien automatisiert mit den Abhängigkeiten zu populieren.

Quelltext 4.3: Abstraktion von Icon-Framework-Spezifika

```
@AllArgsConstructor(access = AccessLevel.PRIVATE)
public enum BootstrapUIIcon implements UIIcon {
    ARROW_LR("arrow-left-right"),
    ...
    private final String iconName;
    private static final String PREFIX = "bi-";
    ...
    public String getIconName() {
        return PREFIX + this.iconName;
    }
}
```

[7] `https://icons.getbootstrap.com/` – Zugriff: 30.01.2022
[8] `https://fontawesome.com/` – Zugriff: 30.01.2022

4.2. Benutzeroberfläche

Informationen zum *User Interface* und dessen Design und Aufbau können der Nutzerdokumentation (engl.) (Abschnitt A.3) und der Entwicklerdokumentation (engl.) (Abschnitt A.4) im Anhang entnommen werden. Diese Dokumente sollen als Hilfe bei der Bedienung und Weiterentwicklung der Anwendung verstanden werden.

4.3. Details zur Bereitstellung

Nach dieser technischen Betrachtung soll nun auf die Administratorensicht der Anwendung eingegangen werden.

Zur Bereitstellung der Anwendung wird ein *Docker Image* gepackt, welches bei Aktualisierungen des Anwendungsquelltextes ebenfalls neu erstellt wird. So können Weiterentwicklungen und Fehlerbehebungen ihren Weg schnell in die Ausführungsumgebung finden. Die Abhängigkeiten und Konfigurationen werden dabei in einer sogenannten *Docker Compose-Datei*[9] (vgl. Quelltext A.1 im Anhang) abgebildet. Dadurch ist es – beispielsweise durch den Einsatz eines Tools wie *Watchtower*[10] – möglich, automatisch Anwendungsupdates einzuspielen.

Nach dem *Deployment* ist die Anwendung per HTTP unter der IP-Adresse des Servers erreichbar, in Standardkonfiguration auf Port 80. Aufgrund des geplanten Einsatzes als internes Tool und den damit verbundenen Einschränkungen wurde auf eine Absicherung mit TLS verzichtet. Dies muss bei der Verarbeitung von personenbezogenen Daten berücksichtigt werden, da einige Kommunikationsparameter (inklusive Passwörter) auf der Nutzeroberfläche einsehbar sind und Daten ungefiltert aus externen Diensten (hier: Learning Record Stores) abgerufen werden können.

Gemäß Festlegung 5 (Umgebungsvariablen) wurden Parametrisierungen für externe Tools ermöglicht. Die Variablen und deren Semantik sind in Tabelle 4.1 erläutert.

Tabelle 4.1.: Umgebungsvariablen der Anwendung

Variable	Bedeutung
`XAPITOOLS_SEC_USERNAME`	Benutzername für HTTP Basic Authentication
`XAPITOOLS_SEC_PASSWORD`	Passwort für HTTP Basic Authentication
`XAPITOOLS_DB_CONNECTION_STRING`	JDBC-Verbindungsstring (type://host:port/database)
`XAPITOOLS_DB_CONNECTION_USER`	Benutzername für Datenbank
`XAPITOOLS_DB_CONNECTION_PASSWORD`	Passwort für Datenbank
`XAPITOOLS_SIM_BACKEND_BASE_URL`	DATASIM-URL (scheme://host:port)
`XAPITOOLS_SIM_BACKEND_USERNAME`	Benutzername für DATASIM
`XAPITOOLS_SIM_BACKEND_PASSWORD`	Passwort für DATASIM
`XAPITOOLS_SIM_STORAGE_DIR`	Cache-Verzeichnis für Simulationsergebnisse

[9] `https://docs.docker.com/compose/` – Zugriff: 03.02.2022
[10] `https://containrrr.dev/watchtower/` – Zugriff: 03.02.2022

5. Validierung und Bewertung der Anwendung

5.1. Nutzertests

In diesem Abschnitt soll ein Testplan zur Untersuchung der Nutzererfahrung festgehalten werden.

5.1.1. Eingrenzung

Die Tests dienen der Überprüfung, ob bei der Anwendungsausführung Fehler auftreten, der gewünschte Funktionsumfang vorhanden ist und die Nutzererfahrung positiv ist. Mit diesen Tests soll primär die Benutzeroberfläche getestet werden, sowie weiterhin die über die Nutzeroberfläche erreichbaren Serverfunktionen. Außerdem werden Integrationen externer Dienste durch die Testfälle mit abgedeckt.

5.1.2. Testgegenstand

Die zu testende Anwendung entspricht dem *Maven*-Artefakt `de.tudresden.inf.verdatas/xapitools` Version `1.0.0`, welches im *Docker Image* mit dem Tag `nexus.galaxion.de:5050/de.tudresden.inf.verdatas/xapitools:1.0.0`[1] am 06.02.2022 um 22:44 Uhr bereitgestellt wurde. Dieser Stand basiert auf dem *Commit* `6108900` aus dem zu dieser Arbeit gehörenden *Git-Repository*, welches auf *GitHub*[2] unter der *MIT*-Lizenz[3] der Öffentlichkeit zugänglich gemacht wurde. Zur Auslieferung wurde die *Docker Compose-Datei* aus Quelltext A.1 mit angepassten Passwörtern verwendet. Das letzte *Deployment* erfolgte auf einen von den Betreuenden dieser Arbeit festgelegten virtuellen Server am 06.02.2022 um 22:50 Uhr. Die für den Zugriff nötigen Daten wurden am 01.02.2022 um 22:55 Uhr den Testenden übermittelt. Für Interaktionstests wurde ein leerer Learning Record Store in die Anwendung eingetragen. Der verwendete DATASIM-*Service* entstammt dem offiziellen *Git-Repository*[4] des Herstellers und wurde zur schnelleren und zuverlässigeren Integration in ein *Docker Image* gepackt.

[1] SHA256: 255caf24333467692e02512e2f835bcfea0c04a3ac0c23a7a1fd397484b0dd71
[2] `https://github.com/GaLaXy102/xapi-toolkit` – Zugriff: 06.02.2022
[3] `https://choosealicense.com/licenses/mit/` – Zugriff: 06.02.2022
[4] `https://github.com/yetanalytics/datasim` – Zugriff: 06.02.2022, Stand d55f2409

5.1.3. Zu testende Funktionen

Für die folgenden Anforderungen sind Testfälle zu definieren:

- Anforderung 1 (Statementimport)
- Anforderung 2 (Simulationsparametrisierung)
- Anforderung 3 (Simulationsausführung)
- Anforderung 4 (Statementexport)
- Anforderung 7 (Konfigurierbarkeit) (nur eingeschränkt, da globale Systemparameter nicht über die Nutzeroberfläche zu verändern sind)
- Anforderung 9 (Fehlertoleranz)
- Anforderung 11 (Sicherheit)

Außerdem soll die folgende (nicht durch Anforderungen abgedeckte) Entwicklungsaufgabe getestet werden:

- Entwicklungsaufgabe 13 (Navigationswarnung)

Nach der Bearbeitung der Testfälle soll die Intuitivität der Anwendung, die erstellte Dokumentation (Anforderung 8) und die englischsprachige Umsetzung der Nutzeroberfläche (Anforderung 10) separat bewertet werden.

5.1.4. Nicht zu testende Funktionen

Die folgenden nicht-funktionalen Anforderungen werden nicht getestet:

- Anforderung 5 (Zugänglichkeit), da die Barrierefreiheit bei der Anwendungsentwicklung nicht im Vordergrund stand (vgl. Tabelle 3.1) und im Forschungskontext derzeit nicht relevant ist;
- Anforderung 6 (Erweiterbarkeit), weil dieser Punkt nur durch eine Weiterentwicklung der Anwendung getestet werden kann und dies außerhalb des Nutzerfokus ist;
- Anforderung 7 (Konfigurierbarkeit) teilweise, denn das *Deployment* soll nicht durch Nutzende geändert werden;
- Anforderung 12 (Auslieferung), da das *Deployment* bereits erfolgte und die Anforderung selbst mit dem Packen als *Docker Image* (vgl. Entwicklungsaufgabe 4) als gelöst gilt.

5.1.5. Ansatz

Die Testenden erhalten nur durch ihr bestehendes Domänenwissen Unterstützung bei der Anwendungsnutzung. Anhand der detaillierten Schritt-für-Schritt-Tests sollen die Testfälle bearbeitet werden. Testdatensätze werden vom Nutzenden beim Durchlaufen der Tests in Reihenfolge selbst angelegt oder mit den Testfällen bereitgestellt. Ein Testfall gilt als bestanden, wenn der Testende das Testsystem durch die im Testfall benannten Schritte in den gewünschten Zustand versetzen konnte und darüber hinaus alle im Testfall definierten Erwartungen erfüllt wurden. Fehlgeschlagene Tests sollen in Unterabschnitt 5.1.8 aufbereitet werden. Die Tests dürfen unterbrochen und fortgesetzt werden, wenn eine Konkurrenz mit

anderen Nutzenden des Systems ausgeschlossen werden kann. Der Testentwickler empfiehlt, die Tests konsekutiv auszuführen. Sollten unerwartete Fehler (also solche, die nicht im Testfall definiert wurden) auftreten, so ist der gesamte Testfall abzubrechen und der Testentwickler unter Angabe des *Tracing Codes* zu verständigen.

5.1.6. Testfälle

Für die Bearbeitung der nun folgenden Testfälle kann die Vorlage aus Abschnitt A.1 genutzt werden.

Testfall 1 (Anmeldung)**.**

Anforderung	Anforderung 11 (Sicherheit)
Startzustand	Der Testende ist abgemeldet (beispielsweise durch Nutzung eines privaten Tabs)
Parametrisierungen	leer: Zugangsdatenabfrage wird durch Testenden abgebrochen ungültig: Nutzername: `test`, Passwort: `test` gültig: bekanntgegebene Zugangsdaten
Endzustand	Der Testende ist angemeldet.

Vorgehen:

1. Der Testende ruft die Web-Anwendung mit seinem Browser unter der mitgeteilten URL auf.
   Erwartung 1: Das Passwortabfragefenster des Webbrowsers erscheint.
2. Die Parametrisierung `leer` wird durch Abbruch der Abfrage durchgeführt.
   Erwartung 2: Ein Fehlertext mit dem Statuscode 401 wird angezeigt.
3. Schritt 1 wird wiederholt.
   Erwartung 3: Das Passwortabfragefenster des Webbrowsers erscheint.
4. Die Parametrisierung `ungültig` wird in die Abfragemaske eingetragen und bestätigt.
   Erwartung 4: Das Passwortabfragefenster des Webbrowsers erscheint erneut oder ein Fehlertext mit dem Statuscode 401 wird angezeigt.
5. Schritt 1 wird wiederholt.
   Erwartung 5: Das Passwortabfragefenster des Webbrowsers erscheint.
6. Die Parametrisierung `gültig` wird in die Abfragemaske eingetragen und bestätigt.
   Erwartung 6: Die Benutzeroberfläche der Anwendung wird geladen und zeigt einen flächigen Text `Welcome to xAPI Toolkit!` an.
   Erwartung 7: Die URL der Browser-Anwendung hat den Pfad `/ui`.

Testfall 2 (Parametrisierung der LRS-Verbindung)**.**

Anforderung	Anforderung 7 (Konfigurierbarkeit)
Startzustand	Der Testende ist angemeldet (vgl. Testfall 1).
Parametrisierungen	unvollst.: Ein beliebiges Feld wird durch den Testenden leer gelassen. ungültiger Server: Name: `Test inv` Endpoint: `http://foo.example.org/bar` Client Key: `test` Client Secret: `test` ungültiges Passwort: Name: `Test2` Endpoint: `https://ba-lrs.galaxion.de/data/xAPI` Client Key: `24880bc900758f118e6ade00e639edf0f6b54557` Client Secret: `invalid` gültig: (alternativ: beliebige andere gültige Verbindungsparameter) Name: `Testverbindung` Endpoint: `https://ba-lrs.galaxion.de/data/xAPI` Client Key: `24880bc900758f118e6ade00e639edf0f6b54557` Client Secret: `d2553e87856286966c77cb0aa5338aebc05a572f`
Endzustand	Es existiert eine gültige LRS-Verbindung.

Vorgehen:

1. Der Testende klickt in der Seitenleiste auf `LRS Connections`.
   Erwartung 1: Die angeklickte Schaltfläche wird hervorgehoben.
   Erwartung 2: Der Seiteninhalt zeigt eine Karte mit dem Knopf `Create new Connection` an.
   Variable n: Auf der Seite werden $n \in \mathbb{N}_{\geq 0}$ Verbindungskarten angezeigt.

2. Der Testende klickt auf den Körper der Karte `Create new Connection`.
   Erwartung 3: Ein Formular zur Abfrage der LRS-Verbindungsdetails wird angezeigt.

3. Die Parametrisierung `unvollst.` wird eingetragen und mit `Save` versucht zu senden.
   Erwartung 4: Eine Browser-seitige Fehlermeldung erscheint und beschreibt den ungültigen Formularinhalt.

4. In das noch offene Fenster aus Schritt 3 wird die Parametrisierung `ungültiges Passwort` eingetragen und mit `Save` gesendet.
   Erwartung 5: Die Anwendung akzeptiert die Eingabe.
   Erwartung 6: Der Testende wird auf die Übersichtsseite mit den LRS-Verbindungen weitergeleitet.
   Erwartung 7: Auf der angezeigten Seite werden $n + 1$ Verbindungskarten angezeigt.
   Variable n': Auf der Seite werden $n' \in \mathbb{N}_{\geq 1}$ Verbindungskarten angezeigt.

5. Der Testende klickt auf der mit `Test` (vgl. Parametrisierung `ungültiges Passwort`) betitelten Karte auf `Edit`.

Erwartung 8: Ein mit den Daten aus der Parametrisierung gefülltes Formular zur Abfrage der LRS-Verbindungsdetails wird angezeigt.

6. Die Parametrisierung `gültig` wird eingetragen und mit `Save` gesendet.
   Erwartung 9: Die Anwendung akzeptiert die Eingabe.
   Erwartung 10: Der Testende wird auf die Übersichtsseite der LRS-Verbindungen weitergeleitet.
   Erwartung 11: Auf der angezeigten Seite werden n' Verbindungskarten angezeigt.
7. Schritt 2 wird wiederholt.
8. Das Formular wird mit den Daten aus Parametrisierung `ungültiger Server` ausgefüllt und mit `Save` abgeschickt.
   Erwartung 12: Die Anwendung akzeptiert die Eingabe.
   Erwartung 13: Der Testende wird auf die Übersichtsseite der LRS-Verbindungen weitergeleitet.
   Erwartung 14: Auf der angezeigten Seite werden $n' + 1$ Verbindungskarten angezeigt.
9. Der Testende lädt die Anwendung (beispielsweise mit `F5`) neu.
   Erwartung 15: In der Seitenleiste unten werden unter anderem Statusindikatoren mit den Titeln `Testverbindung` (Indikator grün) und `Test inv` (Indikator rot) angezeigt.
10. Schritt 1 wird wiederholt.
    Variable n'': Auf der Seite werden $n'' \in \mathbb{N}_{\geq 2}$ Verbindungskarten angezeigt.
11. Der Testende klickt auf der mit `Test inv` betitelten Karte auf `Deactivate`.
    Erwartung 16: Es werden $n'' - 1$ Verbindungskarten angezeigt.
12. Schritt 9 wird wiederholt.
    Erwartung 17: In der Seitenleiste unten wird der Statusindikator mit dem Titel `Test inv` nicht mehr anzeigt.

Testfall 3 (Statementtransfer)**.**

Anforderung	Anforderung 1 (Statementimport) Anforderung 4 (Statementexport)
Startzustand	Der Testende ist angemeldet (vgl. Testfall 1). Es existiert eine gültige LRS-Verbindung (vgl. Testfall 2).
Parametrisierungen	leere Datei: empty_statements.json gültige Datei: test_statements.json
Endzustand	Der Test-LRS enthält Daten.

Vorgehen:

1. Der Testende klickt in der Seitenleiste auf `Statement Exchange`.
   Erwartung 1: Die Nutzeroberfläche zeigt zwei Karten, `Import from JSON` und `Export to JSON`.
2. In der Karte mit dem Titel `Export to JSON` wählt der Nutzende den in Testfall 2 mit der Parametrisierung `gültig` erstellten LRS (`Testverbindung`) im Dropdown aus.
3. Der Testende klickt auf `Retrieve`.
   Erwartung 2: Eine JSON-Datei wird heruntergeladen.
   Variable n: Die heruntergeladene Datei enthält $n \in \mathbb{N}_{\geq 0}$ xAPI-Statements. (Bei Verwendung eines leeren LRS gilt $n = 0$.)

4. In der mit `Import from JSON` betitelten Karte wird der LRS aus Schritt 2 ausgewählt und die Datei aus der Parametrisierung `leere Datei` wird ausgewählt. Der Vorgang wird mit einem Klick auf `Send` bestätigt.
   Erwartung 3: Eine Meldung mit dem Beleg über das Senden von 0 xAPI-Statements wird angezeigt.

5. Schritte 2 und 3 werden wiederholt.
   Erwartung 4: Die neu heruntergeladene Datei enthält genau $n+0 = n$ xAPI-Statements.

6. Schritt 4 wird mit der Parametrisierung `gültige Datei` wiederholt.
   Erwartung 5: Das erfolgreiche Laden von 210 xAPI-Statements wird bestätigt.

7. Schritte 2 und 3 werden wiederholt.
   Erwartung 6: Die exportierte Datei enthält n + 210 xAPI-Statements.

Testfall 4 (Fehlermeldungen).

Anforderung	Anforderung 9 (Fehlertoleranz) Anforderung 7 (Konfigurierbarkeit)
Startzustand	Der Testende ist angemeldet (vgl. Testfall 1). Die Test-LRS-Verbindungen wurden angelegt (vgl. Testfall 2).
Parametrisierungen	leere Datei: empty_statements.json
Endzustand	unverändert

Vorgehen:

1. Der Testende klickt in der Seitenleiste auf `LRS Connections`.
   Variable n: Auf der Seite werden $n \in \mathbb{N}_{\geq 0}$ Verbindungskarten angezeigt.

2. Auf der linken Karte mit dem Inhalt `Create new Connection` wird der Knopf `Also show inactive Connections` angezeigt und ausgewählt.
   Erwartung 1: Es werden n + 1 Verbindungskarten angezeigt. (Falls nicht, wurde der Testfall 2 nicht korrekt ausgeführt. Die Verbindung für die Parametrisierung `ungültig` muss hinzugefügt werden. Es kann dann mit Schritt 4 fortgefahren werden.)

3. Auf der neu erschienen Verbindungskarte mit dem Namen `Test inv` (oder wie vorher parametrisiert) klickt der Testende auf `Reactivate`.

4. Die Anwendung wird durch den Testenden (beispielsweise durch Betätigen der Taste `F5`) neu geladen.
   Erwartung 2: In der Seitenleiste unten wird unter anderem ein Statusindikator mit dem Titel `Test inv` (Indikator rot) angezeigt.

5. Die Funktion `Statement Exchange` wird durch Klicken der entsprechenden Schaltfläche in der Seitenleiste gestartet.

6. Auf der Karte mit dem Titel `Import from JSON` wird der LRS `Test inv` und die Datei aus der Parametrisierung `leere Datei` ausgewählt.

7. Die Anfrage wird durch Klick auf die Schaltfläche `Send` abgeschickt.
   Erwartung 3: Eine Fehlermeldung wird angezeigt. (Status: `503 Service unavailable`, *Tracing Code*, Grund: `No connection . . .`).

8. Der Testende stellt durch die Navigation `LRS Connections` → `Test inv: Deactivate` und anschließendes Neuladen der Anwendung (beispielsweise mit `F5`) den Ursprungszustand wieder her.

Testfall 5 (Simulationsparametrisierung)**.**

Anforderung	Anforderung 2 (Simulationsparametrisierung) Entwicklungsaufgabe 13 (Navigationswarnung)
Startzustand	Der Testende ist angemeldet (vgl. Testfall 1).
Parametrisierungen	gültig: Remark: TestSim xAPI Profile: yetanalytics-cmi5 Personae: - Persona Name: Test1 Persona Mail: test1@example.org - Persona Name: Test2 Persona Mail: test2@example.org Alignments: - Type: StatementTemplate Component: https://w3id.org/xapi/cmi5#failed Test1: Minimum (-1), Regler ganz links Test2: Maximum (+1), Regler ganz rechts Number of Statements: 1000 Simulation Seed: 2521 Simulation Start: 08.02.2022 12:36:28 Simulation End: 15.02.2022 12:36:28
Endzustand	Eine gültige Simulationsbeschreibung wurde angelegt.

Vorgehen:

1. Der Testende klickt in der Seitenleiste auf `Simulations`.
   Erwartung 1: Es wird eine Seite mit mindestens der Karte `Create new Simulation` angezeigt.
   Variable *n*: Auf der Seite werden $n \in \mathbb{N}_{\geq 0}$ Simulationskarten angezeigt.
2. Die Karte `Create new Simulation` wird angeklickt.
   Erwartung 2: Der Simulationsassistent beginnt mit der Abfrage eines Titels (`Remark`).
3. Der `Remark` aus der Parametrisierung `gültig` wird in das Formular eingetragen.
4. Der Testende klickt auf die Schaltfläche `Statement Exchange` in der Seitenleiste.
   Erwartung 3: Eine Warnung wird angezeigt. Diese wird mit Abbrechen/Cancel o.ä. quittiert.
5. Das Formular wird mit Klick auf `Continue` abgesendet.
   Erwartung 4: Das für die Simulation zu verwendende xAPI-Profil wird nun abgefragt.
6. Das xAPI-Profil der Parametrisierung `gültig` wird ausgewählt und mit `Continue` abgesendet.
   Erwartung 5: Der Assistent fragt die zu verwendenden Personae ab.
7. Nacheinander werden die Persona im Formular `Add new Persona` wie von der Parametrisierung `gültig` vorgegeben eingetragen und durch Klick auf + der Simulation hinzugefügt.

Erwartung 6: Im Formular `Select Personae for Simulation` sind die Personae `Test1` und `Test2` ausgewählt.

8. Der Testende klickt auf `Continue`.
   Erwartung 7: Die Formulare `Add new Component` und `Map Affinities` werden angezeigt.

9. Im Formular `Add new Component` wird Type und Component des Alignments aus der Parametrisierung `gültig` eingetragen und mit + zur Simulation hinzugefügt.
   Erwartung 8: Das Formular `Map Affinities` enthält genau die zu alignende Komponente.

10. Die Komponente `https://w3id.org/xapi/cmi5#failed` wird im Formular `Map Affinities` ausgeklappt (Symbol: ∨).
    Erwartung 9: Die Schieberegler der Personae sind mittig ausgerichtet (neutral).

11. Die angezeigten Schieberegler werden gemäß Parametrisierung `gültig` ausgerichtet und der Vorgang mit `Continue` übermittelt.
    Erwartung 10: Es erscheint das Formular `Simulation Parameters`.

12. Das Formular `Simulation Parameters` wird gemäß Parametrisierung `gültig` ausgefüllt und mit `Continue` abgesendet.
    Erwartung 11: Eine Seite, welche eine Karte enthält, die die gegebene Parametrisierung zusammenfasst, wird angezeigt.
    Erwartung 12: Es werden für alle parametrisierten Simulationsbestandteile Stifte angezeigt. Diese können zum Editieren (gemäß Definition 1 (Projektierung)) genutzt werden, welches jedoch nicht getestet wird, da dies keine formulierte Anforderung war.

13. Der Testende klickt auf die Schaltfläche `Finalize` am unteren Ende der angezeigten Simulationskarte.
    Erwartung 13: Die Editier-Stifte werden nicht mehr angezeigt. Stattdessen erscheint am Kartenende eine Schaltfläche `Perform`.

14. Der Testende klickt auf die Schaltfläche `Copy`.
    Erwartung 14: Eine wieder editierbare Kopie der Simulationskarte wird angezeigt.
    Erwartung 15: Die Schaltfläche `Perform` wird nicht mehr angezeigt, stattdessen erscheint am Kartenende eine Schaltfläche `Finalize`.

15. Am Ende der Simulationskarte wird auf die Schaltfläche `Delete` geklickt.
    Erwartung 16: Die Benutzeroberfläche zeigt $n + 1$ Simulationskarten an.

Testfall 6 (Simulationsdurchführung)**.**

Anforderung	Anforderung 3 (Simulationsausführung)
Startzustand	Der Testende ist angemeldet (vgl. Testfall 1). Es existiert eine gültige LRS-Verbindung (vgl. Testfall 2), der zugehörige Statusindikator in der linken Seitenleiste ist grün. Eine gültige Simulationsbeschreibung wurde angelegt (vgl. Testfall 5). Es besteht eine Verbindung zu DATASIM (der zugehörige Statusindikator in der linken Seitenleiste ist grün).
Parametrisierungen	∅
Endzustand	Die Simulation wurde durchgeführt.

Vorgehen:

1. Der Testende klickt in der Seitenleiste auf `Simulations`.
2. An der Simulationskarte mit dem Titel `TestSim` (vgl. Parametrisierung `gültig` aus Testfall 5) klickt der Testende auf `Retrieve`.
   Erwartung 1: Eine JSON-Datei mit einer Liste von xAPI-Statements wird heruntergeladen.
   Variable *n*: Die Datei enthält $n \in \mathbb{N}_{\geq 0}$ xAPI-Statements (die gegebene Parametrisierung ergibt $n = 415$).
3. An der Simulationskarte wird die Schaltfläche `Push` angeklickt und der in Testfall 2 erstellte LRS mit dem Namen `Testverbindung` ausgewählt.
   Erwartung 2: Eine Bestätigung über den Versand der *Statements* wird angezeigt.

5.1.7. Ergebnisse des Testausführung

Die Tests wurden am 09.02.2022 von Dr.-Ing. Tommy Kubica und Robert Schmidt ausgeführt. Die Protokolle gemäß Abschnitt A.1 (Vorlage für die Durchführung der Nutzertests) befindet sich im Anhang in Unterabschnitt A.2.1 und Unterabschnitt A.2.2.
Im Folgenden werden die Ergebnisse des Testplans diskutiert.

Testfall 1 (Anmeldung)

Der Testfall wurde kommentarlos bestanden.

Testfall 2 (Parametrisierung der LRS-Verbindung)

Es wurde angemerkt, dass die Benutzeroberfläche einige Fehler bei der Ladeanzeige aufweist (vgl. Unterabschnitt 5.2.7). Die Formulare der Benutzeroberfläche könnten klarer zwischen Formularfeldern und Formulartiteln differenziert angezeigt werden. Für die Knöpfe der Seitenleiste wären *Hover*-Effekte dem Nutzererlebnis dienlich. Es gibt einen Fehler in der Definition des Testfalls, der Erwartung 6 fehlschlagen lässt: Der zum Testen verwendete Learning Record Store *Learning Locker* dedupliziert eingehende xAPI-Statements, weshalb das wiederholte Absenden der gleichen *Statements* (diese sind hier wegen der selben Simulationsparameter identisch) zum gleichen Zustand des LRS führt.

Aus technischer Sicht wurde der Test bestanden.

Testfall 3 (Statementtransfer)

Auch hier wurde angemerkt, dass die Ladeindikatoren bei Ablehnung der Eingabe nicht richtig funktionieren. Weitere Einwände wurden nicht mitgeteilt; der Testfall ist also bestanden.

Testfall 4 (Fehlermeldungen)

Negativ fiel auf, dass die Liste der Statusindikatoren nicht automatisiert neu geladen wird, wenn sich die externen Dienste verändern. Als Abhilfe wurde nahe gelegt, einen Bestätigungstoast anzuzeigen.

Alle Erwartungen wurden erfüllt, was ein Bestehen des Tests impliziert.

Testfall 5 (Simulationsparametrisierung)

Bei diesem Test wurde eine Erwartung, das Warnen bei der Navigation mit ungespeicherten Eingaben (3), nicht erfüllt. Außerdem funktioniert diese Warnung bei der Profilauswahl nicht.

Bezüglich der `User Experience` wurde die Eingabe des Titels als uneinheitlich kritisiert. Ebenfalls wird der Bezeichner für das E-Mail-Feld bei der Erstellung einer Persona nicht korrekt formatiert, sobald das Eingabefeld fokussiert wird.

Bei der Wichtung von Alignments wurde gewünscht, den Schieberegler zusätzlich mit Zahlen zu beschriften.

Es wurde angemerkt, dass die verwendeten *Icons* teilweise nicht selbsterklärend sind und deshalb mit Erklärtexten versehen werden könnten.

Da die fehlgeschlagene Erwartung die korrekte Funktion der Anwendung als Ganzes nicht beeinträchtigt, wurde der Testfall als bestanden markiert.

Testfall 6 (Simulationsdurchführung)

Ein Fehler bei der Definition des Testfalls ist aufgefallen: Bevor das Simulationsergebnis bereitgestellt werden kann, muss die Simulation durch Klick auf `Perform` durchgeführt werden. Da je nach Rechenleistung diese Berechnung etwas Zeit in Anspruch nehmen kann, wurde der Vorschlag unterbreitet, die ungefähre Simulationsdauer mit anzuzeigen.

Auch dieser Testfall wurde erfolgreich abgeschlossen.

Freie Kommentare

Einige Eingabefelder sind nicht korrekt beschriftet (insbesondere `Remark`, wenn es sich um einen Titel handelt). Außerdem wurden die Regeln der englischen Sprache bezüglich Groß- und Kleinschreibung teilweise nicht eingehalten.

Als Wünsche für zusätzliche Funktionen wurden die Implementierung der Editierbarkeit von Personae und das Löschen inaktiver LRS-Verbindungen geäußert.

5.1.8. Erkannte Probleme und deren Behebung

Zur Lösung der Probleme bei der Nutzererfahrung werden folgende Vorschläge eingebracht:

- Delegation der Ladeanzeigen nur an nicht browserseitig validierte Knöpfe, um bei Validierungsfehlern diese nicht unendlich lang anzuzeigen
- Formatieren von Formulartiteln als Überschrift
- Hinzufügen von *Hover*-Effekten für die Navigationsknöpfe der Seitenleiste
- Implementieren einer Bestätigungsmeldung für das Reaktivieren von Verbindungen zu Learning Record Stores
- Erweitern der Navigationswarnung auf *Dropdowns*
- Koppeln der Navigationsknöpfe der Seitenleiste an den Anwendungszustand (derzeit an den Klick gekoppelt)
- Überprüfen der Beschriftung aller Formularfelder und der Groß-/Kleinschreibung
- Anpassen der Formatierung des E-Mail-Felds für Personae
- Hinzufügen von Alternativtexten für *Icons*
- Berechnen und Anzeigen der erwarteten Simulationsdauer

Die Bearbeitung dieser Vorschläge ist für die Version 1.1.0 der Anwendung geplant, deren Veröffentlichung für den 16.02.2022 terminiert ist.

Die Implementierung des automatischen Neuladens bei Statusindikatoren für externe Dienste ist derzeit strukturell mit einem Neuladen der Anwendung verbunden. Es könnte geprüft werden, einen Knopf hierfür einzufügen, wodurch jedoch die jeweilige Nutzernavigation verloren ginge.

Die unter *Freie Kommentare* genannten Funktionswünsche könnten später als Erweiterung der Anwendung implementiert werden.

5.2. Technische Tests

Hier findet sich der Testplan für die in Festlegung 6 benannten Aspekte.

5.2.1. Eingrenzung

Mit diesen Tests soll die Anwendung aus technischer Sicht auf korrektes Verhalten geprüft werden. Genauer soll beispielsweise betrachtet werden, ob die Schnittstellenspezifikationen bei der Kommunikation mit externen Diensten eingehalten werden. Außerdem soll das Verhalten bei provozierten Kommunikationsfehlern betrachtet werden.

5.2.2. Testgegenstand

Die Tests beziehen sich auf den gleichen Gegenstand wie in Unterabschnitt 5.1.2; die Instanz wird jedoch lokal betrieben.

5.2.3. Zu testende Entwicklungen

Gemäß Festlegung 6 werden die Tests die folgenden Entwicklungsaufgaben abdecken:

- Entwicklungsaufgabe 1 (DATASIM per REST)
- Entwicklungsaufgabe 5 (Senden von Statements), darin eingeschlossen Entwicklungsaufgabe 10 (Datenstruktur für LRS)
- Entwicklungsaufgabe 6 (Auslesen von Statements), darin eingeschlossen Entwicklungsaufgabe 10 (Datenstruktur für LRS)
- Entwicklungsaufgabe 9 (Durchführung von Simulationen), dort insbesondere auch Entwicklungsaufgabe 11 (Datenstruktur für DATASIM)
- Entwicklungsaufgabe 8 (Parametrisierung von Simulationen), insbesondere mit Überprüfung auf den maximalen Abschluss
- Fehlermeldungen (vgl. Festlegung 4 (Fehlermanagement))

Außerdem soll durch Brute-Force-Techniken geprüft werden, ob über die Nutzeroberfläche ein ungültiger Anwendungszustand erreicht werden kann.

5.2.4. Nicht zu testende Entwicklungen

Die hier aufgezählten Aufgaben werden nicht technisch getestet:

- Entwicklungsaufgabe 2 (Dokumentation), weil das nicht technisch testbar ist;
- Entwicklungsaufgabe 3 (HTTP Basic Authentication), da diese Aufgabe durch Testfall 1 abgedeckt ist;
- Entwicklungsaufgabe 4 (Docker), denn die Existenz eines *Deployments* aus dem generierten *Docker Image* beweist die Erfüllung;
- Entwicklungsaufgabe 7 (Import von Statements), weil externe Schnittstellenspezifikationen (hier: die Gültigkeit der Eingabedaten) nicht getestet werden soll;
- Entwicklungsaufgabe 12 (xAPI-Profile), da diese Aufgabe aufgrund ihrer Komplexität nur subjektiv bewertet werden kann;
- Entwicklungsaufgabe 13 (Navigationswarnung), weil Testfall 5 diese Aufgabe mit abdeckt.

5.2.5. Ansatz

Die Tests werden als *Proof-of-Work* durch den Entwickler durchgeführt. Für den *Brute-Force*-Test zum Erreichen illegaler Zustände leisteten Alexander Petrenz (M.Sc.) und Willi Sontopski (M.Sc.) zusätzliche Unterstützung. Für die Tests werden zufällige Beispieldatensätze durch die Anwendung angelegt.

5.2.6. Testfälle

Testfall 7 (DATASIM-Datenstruktur)**.**

Anforderung	Entwicklungsaufgabe 9 (Durchführung von Simulationen) Entwicklungsaufgabe 11 (Datenstruktur für DATASIM)
Startzustand	Die Anwendung wurde im `dev`-Profil gestartet. Der Testende ist angemeldet (vgl. Testfall 1).
Parametrisierungen	∅
Endzustand	unverändert

Vorgehen:

1. Der Testende klickt in der Seitenleiste auf `Simulations`.

2. An einer zufällig ausgewählten Simulationskarte klickt der Testende auf den `Download`-Knopf in der Kopfzeile.
   Erwartung 1: Die Simulationsbeschreibung wird als JSON-Dokument heruntergeladen.
   Erwartung 2: Die bereitgestellte Datei genügt dem DATASIM-Schema „*Simulation Inputs > Simulation Specification*“[5].

Testfall 8 (DATASIM-REST-Schnittstelle)**.**

Anforderung	Entwicklungsaufgabe 1 (DATASIM per REST)
Startzustand	Die Anwendung wurde im `dev`-Profil gestartet. Der Testende ist angemeldet (vgl. Testfall 1). Testfall 7 (DATASIM-Datenstruktur) ist gültig
Parametrisierungen	∅
Endzustand	unverändert

Vorgehen:

1. Der Testende klickt in der Seitenleiste auf `Simulations`.

2. Die Anzahl zu generierender *Statements* einer zufälligen Simulation wird (gemäß Bedienungsanleitung) auf 10 gesetzt.

3. An der zufällig ausgewählten Simulationskarte klickt der Testende auf den `Finalize`-Knopf.

4. Die Aufzeichnung des von der Anwendung ausgehenden Traffics (beispielsweise mit Wireshark) beginnt.

5. An ebendieser Simulationskarte klickt der Testende auf den `Perform`-Knopf.

6. Der Testende wartet auf das Neuladen der Seite.

7. Die Aufzeichnung des von der Anwendung ausgehenden Traffics endet.
   Erwartung 1: Der beobachtete Traffic entspricht der DATASIM-API-Spezifikation[6].

[5] `https://github.com/yetanalytics/datasim#simulation-inputs` – Zugriff: 09.02.2022
[6] `https://github.com/yetanalytics/datasim#post-apiv1generate` – Zugriff: 09.02.2022

8. An der zuvor ausgewählten Simulationskarte klickt der Testende auf den `Retrieve`-Knopf.
   Erwartung 2: Das Simulationsergebnis wird als JSON-Dokument heruntergeladen.
   Erwartung 3: Die bereitgestellte Datei genügt dem xAPI-Schema. (Zur Überprüfung dieser Erwartung kann die Bibliothek *xAPI-Schema*[7] von *Yet Analytics* genutzt werden.)

Testfall 9 (LRS-Kommunikation sendend).

Anforderung	Entwicklungsaufgabe 5 (Senden von Statements) Testfall 7 (DATASIM-Datenstruktur) ist gültig.
Startzustand	Die Anwendung wurde im `dev`-Profil gestartet. Der Testende ist angemeldet (vgl. Testfall 1). Testfall 8 wurde erfolgreich ausgeführt.
Parametrisierungen	http-insecure: `http://ba-lrs.galaxion.de/data/xAPI`
Endzustand	unverändert

Vorgehen:

1. Der Testende klickt in der Seitenleiste auf `LRS Connections`.
2. Die URL des LRS `Galaxion LRS` wird (gemäß Bedienungsanleitung) auf die Parametrisierung `http-insecure` gesetzt.
3. Die Anwendung wird (beispielsweise mit `F5`) neu geladen.
4. Der Testende klickt in der Seitenleiste auf `Simulations`.
5. Die Aufzeichnung des von der Anwendung ausgehenden Traffics (beispielsweise mit Wireshark) beginnt.
6. An der ausgeführten Simulationskarte (aus Testfall 8) klickt der Testende auf den `Push`-Knopf und wählt den LRS `Galaxion LRS` aus.
   Erwartung 1: Der Statementtransfer wird bestätigt.
7. Die Aufzeichnung des von der Anwendung ausgehenden Traffics endet.
   Erwartung 2: Der beobachtete Traffic entspricht der xAPI-Spezifikation, Teil Kommunikation, Abschnitt „*POST Statements*“[8].

[7] `https://github.com/yetanalytics/xapi-schema#plain-ol-javascript` – Zugriff: 09.02.2022
[8] `https://github.com/adlnet/xAPI-Spec/blob/1.0.3/xAPI-Communication.md#212-post-statements` – Zugriff: 09.02.2022

Testfall 10 (LRS-Kommunikation empfangend)**.**

Anforderung	Entwicklungsaufgabe 6 (Auslesen von Statements)
Startzustand	Die Anwendung wurde im `dev`-Profil gestartet. Der Testende ist angemeldet (vgl. Testfall 1). Der Testende hat Zugriff auf die Nutzeroberfläche des LRS.
Parametrisierungen	∅
Endzustand	unverändert

Vorgehen:

1. Auf der Benutzeroberfläche der LRS überprüft der Testende, wie viele *Statements* im gewählten LRS existieren.
   Variable n: Im LRS sind $n \in \mathbb{N}_{\geq 0}$ *Statements* gespeichert.

2. Der Testende klickt in der Seitenleiste auf `Statement Exchange`.

3. Auf der Karte `Export to JSON` wählt der Testende einen gültigen LRS aus und klickt auf `Retrieve`.
   Erwartung 1: Ein JSON-Dokument wird heruntergeladen.
   Erwartung 2: Das Dokument enthält n Einträge.

4. Ein Export aller *Statements* wird über die Nutzeroberfläche des vorher gewählten LRS angefordert.
   Erwartung 3: Das vom LRS erstellte Dokument und das in Schritt 3 heruntergeladene Dokument sind identisch.

Testfall 11 (Parametrisierung)**.**

Anforderung	Entwicklungsaufgabe 8 (Parametrisierung von Simulationen)
Startzustand	Die Anwendung wurde im `dev`-Profil gestartet. Der Testende ist angemeldet (vgl. Testfall 1).
Parametrisierungen	∅
Endzustand	unverändert

Vorgehen:

1. Der Testende setzt sich mit der Eingabedaten-Spezifikation von DATASIM[9] auseinander.

2. Es wird in der Seitenleiste auf `Simulations` geklickt und dann durch Auswahl der Karte `Create new Simulation` eine neue Simulation angelegt.

3. Beim Iterieren der Formulare überprüft der Testende, ob alle Bestandteile der Datenspezifikation konfiguriert werden können. Für nicht konfigurierbare Bestandteile legt der Testende eine Bewertung (*Gravity*, auf einer Skala von 0% (absolut unnötig) über 50% (dringend benötigt) bis 100% (Anwendung nicht nutzbar)) fest, wie sehr sich die fehlende Konfigurierbarkeit auf das Resultat dieser Arbeit auswirkt.
   Erwartung 1: Es gibt keine Parameter mit einer *Gravity* von größer als 50%, die nicht konfiguriert werden können.
   Erwartung 2: Es sind mehr als 80% der Parameter aus der Spezifikation konfigurierbar.

[9] `https://github.com/yetanalytics/datasim#simulation-inputs` – Zugriff: 09.02.2022

Testfall 12 (Fehlerbehandlung)**.**

Anforderung	Festlegung 4 (Fehlermanagement)
Startzustand	Die Anwendung wurde im `dev`-Profil gestartet. Die externe Abhängigkeit `DATASIM` wird ausgeschaltet. Der Testende ist angemeldet (vgl. Testfall 1).
Parametrisierungen	Einzelstatement: one_statement.json Syntaxfehler: bad_statement_in_list.json
Endzustand	unverändert

Vorgehen:

1. Der Testende klickt in der Seitenleiste auf `Statement Exchange`.
2. Auf der Karte `Export to JSON` wählt der Testende einen ungültigen LRS (roter Statusindikator) aus und klickt auf `Retrieve`.
   Erwartung 1: Eine Fehlerseite erscheint mit der Meldung, dass der LRS nicht erreichbar ist.
   Hinweis: Diese Meldung erscheint hier aus technischen Gründen in einem neuen Tab, weshalb der Zurück-Knopf nicht funktioniert. Der Tab muss manuell geschlossen werden.
3. Über die Karte `Import from JSON` lädt der Testende die Datei aus der Parametrisierung `Einzelstatement` in einen gültigen LRS hoch.
   Erwartung 2: Eine Fehlermeldung erscheint mit der Meldung, dass die Datei falsch formatiert ist.
4. Über die Karte `Import from JSON` lädt der Testende die Datei aus der Parametrisierung `Syntaxfehler` in einen gültigen LRS hoch.
   Erwartung 3: Die Anwendung meldet einen Fehler bei der Verarbeitung der *Statements* durch den LRS.
5. Der Testende klickt in der Seitenleiste auf `Simulations`.
6. Eine beliebige Simulationskarte wird mit `Finalize` und `Perform` an DATASIM gesendet. (Erinnerung: DATASIM ist inaktiv)
   Erwartung 4: Die Nichterreichbarkeit von DATASIM wird dem Nutzenden mitgeteilt.
7. Eine zufällige Unterseite wird über die Adressleiste aufgerufen (hier: `/foo`).
   Erwartung 5: Eine Fehlerseite, welche auf das Nichtvorhandensein der angeforderten Seite hinweist, wird angezeigt.

Testfall 13 (Ungültige Navigationen)**.**

Anforderung	-
Startzustand	Die Anwendung wurde im `dev`-Profil gestartet. Der Testende ist angemeldet (vgl. Testfall 1).
Parametrisierungen	∅
Endzustand	undefiniert

Vorgehen:

1. Der Testende versucht ausgiebig, eine Fehlermeldung zu erzeugen, welche nicht den HTTP-Zustand 503 (`Service Unavailable`) anzeigt.
   Erwartung 1: Keine Fehlermeldung kann provoziert werden.

5.2.7. Ergebnisse der Testausführung

Die Ergebnisse der Tests sind hier nach Testfällen aufgeschlüsselt.

Testfall 7 (DATASIM-Datenstruktur)

Über die Benutzeroberfläche konnte eine Simulationsbeschreibung heruntergeladen werden.
Laut der angegebenen Quelle soll die heruntergeladene Datei folgender Syntax genügen:

```
;; Reihenfolge aller Variablen irrelevant
;; Whitespace kann zwischen den Symbolen vorhanden sein
<DOCUMENT>     :== '{' <PROFILES> ',' <PARAMETERS> ',' <PERSONAE> ','
    <ALIGNMENTS> '}'

<PROFILES>     :== '"profiles":' '[' <PROFILE> (',' <PROFILE>)* ']'
<PROFILE>      :== <JSON-LD-DOCUMENT>  ;; hier nicht spezifischer

<PARAMETERS>   :== '"parameters":' '{' <STARTTS> ',' <ENDTS> ',' <MAXSTM>
    ',' <TZ> ',' <SEED> '}'
<STARTTS>      :== '"start":' <ISO8601-TIMESTAMP-QUOTED>
<ENDTS>        :== '"end":' <ISO8601-TIMESTAMP-QUOTED>
<TZ>           :== '"tz":' <TZ-DATABASE-NAME-QUOTED>
<MAXSTM>       :== '"max":' <POSITIVE-INTEGER>
<SEED>         :== '"seed":' <POSITIVE-INTEGER>

<PERSONAE>     :== '"personae-array":' '[' <PGROUP> (',' <PGROUP>)* ']'
<PGROUP>       :== '{' <NAME> ',' <OBJTYPE> ',' <GMEMBERS> '}'
<NAME>         :== '"name":' <STRING>
<OBJTYPE>      :== '"objectType":' '"Group"'
<GMEMBERS>     :== '"member":' '[' <GMEMBER> (',' <GMEMBER>)* ']'
<GMEMBER>      :== '{' <NAME> ',' <MBOX> '}'
<MBOX>         :== '"mbox":' '"mailto:'<EMAIL-ADDRESS>'"'

<ALIGNMENTS>   :== '"alignments":' '[' (<ALIGNMENT> (',' <ALIGNMENT>)*)? ']'
<ALIGNMENT>    :== '{' <ACTORID> ',' <ACTORTYPE> ',' <SALIGNMENTS> '}'
;; folgendes muss einem Eintrag <MBOX> entsprechen
<ACTORID>      :== '"id":' '"mbox::mailto:'<EMAIL-ADDRESS>'"'
<ACTORTYPE>    :== '"type": "Agent"'
<SALIGNMENTS>  :== '"alignments":' '[' <SALIGNMENT> ']'
<SALIGNMENT>   :== '{' <COMPONENT> ',' <WEIGHT> '}'
<COMPONENT>    :== '"component":' <URL-QUOTED>
<WEIGHT>       :== '"weight":' <FLOAT-IN--1-1>
```

Die Datei entspricht nach einer manuellen Überprüfung der angegebenen Syntax, der Test ist also bestanden.

Testfall 8 (DATASIM-REST-Schnittstelle)

Der während des Tests aufgezeichnete HTTP-*Request* ist in Unterabschnitt A.2.3 angehangen. Es wurde geprüft, dass die Elemente der *Encapsulated multipart parts* den Ausgaben von Testfall 7 entsprechen.

Offensichtlich genügt die Anfrage der DATASIM-API-Spezifikation[10], welche diese Struktur erwartet.

Leider konnte das Schema zur Validierung nicht kompiliert werden. Da Testfall 6 gültig ist, kann jedoch von einer Gültigkeit der empfangenen *Statements* ausgegangen werden.

Der Testfall ist somit bestanden.

Testfall 9 (LRS-Kommunikation sendend)

Die Bestätigung wurde angezeigt. Der beobachtete Internetverkehr ist verkürzt in Unterabschnitt A.2.4 abgebildet.

Er genügt der angegebenen Spezifikation, weshalb der Test bestanden ist.

Testfall 10 (LRS-Kommunikation empfangend)

Laut Weboberfläche enthält der Test-Learning Record Store $n = 3368$ xAPI-Statements. Der über die zu testende Anwendung durchgeführte Export enthält ebenfalls *n Statements*. Leider unterstützt der angeschlossene LRS (Learning Locker[11] v7.1.1) keinen direkten Export von *Statements*, weshalb Erwartung 3 nicht überprüft werden kann. Empirische Stichproben konnten eine Übereinstimmung der über die Anwendung exportierten *Statements* mit den Daten in der Weboberfläche des LRS feststellen.

Der Testfall ist bestanden.

Testfall 11 (Parametrisierung)

DATASIM sieht (laut `README` in [7]) folgende Parametrisierungen vor:

- xAPI-Profil
- Personae („*xAPI-Actors*“), welche in Gruppen organisiert werden und durch eine E-Mail-Adresse[12] identifiziert werden.
- Alignments zwischen Simulationskomponenten (nicht: simulierten Komponenten[1]) und xAPI-*Actors*[13] mit einem Gewicht zwischen -1 (Interaktion unwahrscheinlich) und 1 (Interaktion sehr wahrscheinlich).
- Simulationsparameter: Frühester und spätester Zeitstempel der generierten *Statements* sowie deren Zeitzone, maximale Anzahl zu generierender *Statements* und Startwert (*Seed*) für den Zufallsgenerator.

Die getestete Anwendung unterstützt lediglich die Verwendung vordefinierter xAPI-Profile. Da diese das standardisierte und verbreitete cmi5-Profil beinhalten, ist die Dringlichkeit zur Nachbesserung mit ca. 25% (sinnvolle Erweiterung) anzugeben. Personae können nicht Gruppen zugeordnet werden. Die weitere Verwendung der Gruppenzuordnung ist nicht dokumentiert, weshalb diese fehlende Funktion unter Vorbehalt mit 0% (absolut unbedeutend) zu bewerten ist.

[10] `https://github.com/yetanalytics/datasim#post-apiv1generate` – Zugriff: 09.02.2022
[11] `https://learningpool.com/solutions/learning-locker-community-overview/` – Zugriff: 11.02.2022
[12] Die Dokumentation lässt offen, ob andere Identifikatoren unterstützt werden.
[13] Die Dokumentation ist unklar, ob eine Gruppe von Personae ebenfalls ein *xAPI-Actor* sein kann.

Alle weiteren Einstellungen können vorgenommen werden, wobei technische Details den Nutzenden verborgen bleiben (unter anderem die Skalierung der Wichtung von Alignments).

Der Test ist bestanden.

Testfall 12 (Fehlerbehandlung)

Vier der fünf Erwartungen wurden erfüllt. Eine Fehlermeldung (Erwartung 3) ist ausbaufähig, da die Fehlermeldung des Learning Record Stores nicht korrekt propagiert wird (ist: `400 Bad Request`, angezeigt: `500 Internal Server Error`). Erwartung 5 ist fehlgeschlagen, da keine wohlformatierte Nachricht angezeigt wird, sondern eine Standardfehlerseite des unterliegenden *Frameworks*.

Der Test kann somit **nicht** als bestanden gewertet werden.

Testfall 13 (Ungültige Navigationen)

Während der Tests wurden viele atypische Eingaben getestet und der optische Gesamteindruck bewertet. Die Anwendung konnte nicht in einen ungültigen Zustand überführt werden und widerstand Versuchen der *SQL Injection*, jedoch waren ungültige Eingaben möglich. Es fiel insbesondere auf, dass zwei Felder bei den Simulationsparametern, `Number of Statements` und `Simulation Seed`, leere Eingaben akzeptieren, obwohl sie vom DATASIM-Schema erwartet werden, was beim Absenden der Simulation zu einem Fehler führt. Ebenfalls wird dort die Eingabe *Startzeitpunkt nach Endzeitpunkt* akzeptiert, welches in einer ungültigen Simulationsbeschreibung resultiert, die auch von DATASIM nicht zurückgewiesen wird (sondern dort einen Fehler provoziert). Außerdem haben die Freitexteingabefelder `Simulations > Remark`, `Simulations > Persona Name` und `LRS Connections > Name` keine Längenbeschränkung, wodurch es zu Fehlern bei der Persistierung langer Eingaben kommt. Durch das Löschen aller Personae einer Simulation oder deren Nichthinzufügen wird ebenfalls ein invalides Simulationsdokument erzeugt.

Es fiel negativ auf, dass die Ladeanzeigen (*Spinner*) bei Ablehnung der Eingabe nicht unsichtbar werden, sowie dass der Knopf `Back` unter `Simulations > Remark` und die *Breadcrumbs* im Seitenmenü keinen Effekt haben. Weiterhin werden die Identifikatoren einzelner Personae nicht mehr angezeigt.

Der Testfall wurde **nicht** bestanden.

5.2.8. Erkannte Probleme und deren Behebung

Die bemängelten Fehlerbehandlungen aus Testfall 12 (Fehlerbehandlung) sollten korrigiert werden.

Die Navigationsprobleme aus Testfall 13 (Ungültige Navigationen) sind nicht strukturell und prinzipiell lösbar. Hierfür werden die folgenden Vorschläge unterbreitet:

- Hinterlegen einer gültigen URL für die `Back`-Schaltfläche bei der Vergabe eines Simulationstitel
- Delegation der Ladeanzeigen nur an nicht browserseitig validierte Knöpfe
- Setzen des Attributs `required` bei Eingabefeldern mit erforderlichem Wert
- Validierung der Zeitstempel der Simulationsparameter im Browser
- Beschränkung der Eingabelänge von Freitextfeldern auf 255 Zeichen[43]
- Verdunkelung der Schritt-*Breadcrumbs* durch Aktualisierung von *Bootstrap* auf Version 5.1 (vgl. Festlegung 7) und Änderung der *HTML*-Klasse von `text-white-50` auf `text-opacity-25`
- Anzeigen der Identifikatoren der Personae im Auswahlfeld
- Weiterleiten auf korrekte Fehlerseite für nicht vergebene Anwendungsrouten
- Korrektur der Propagierung von Fehlern bei der Kommunikation mit Learning Record Stores
- Überprüfen, dass eine Simulation ohne zugewiesene Personae nicht finalisiert werden kann

Die Lösung dieser Probleme ist für eine Version 1.1.0 der Anwendung vorgesehen, welche für den 16.02.2022 terminiert ist.

Später könnte ein Editor für xAPI-Profile der Anwendung hinzugefügt werden (vgl. Testfall 11).

5.3. Beurteilung der Umsetzung der Konzeption

Die Konzepte aus Kapitel 3 konnten mit den benannten Einschränkungen umgesetzt werden. Obwohl es an Zeit für das Schreiben von programmatischen Tests mangelte, wurde mit den in diesem Kapitel beschriebenen Testfällen eine hohe Testabdeckung erreicht. Dass die manuellen Tests positiv ausgefallen sind, bestätigt die Erfüllung der an die Anwendung formulierten Anforderungen.

Kleinere Nachbesserungen sind bei der *User Experience* möglich und werden – sofern zeitlich möglich – im Rahmen dieser Arbeit noch durchgeführt. Mögliche größere Erweiterungsvorhaben sowie weitere Nutzungsmöglichkeiten werden in Kapitel 6 diskutiert.

6. Zusammenfassung und Ausblick

Zum Abschluss dieser Arbeit soll konstatiert werden, welchen Nutzen die entwickelte Anwendung haben kann, wie sie weiterverwendet und -entwickelt werden kann sowie welche Einschränkungen sie derzeit aufweist.

6.1. Erreichte Ergebnisse

Die im Rahmen der Arbeit entwickelte Anwendung geht über den geforderten Prototypen hinaus, wie die Tests aus Kapitel 5 unterstreichen. Insbesondere die einfache Erweiterbarkeit durch die dynamische Zusammenstellung der Benutzeroberfläche eröffnet viele Möglichkeiten für die weitere Entwicklung. Nach Abschluss der Tests wurden viele der dort bemerkten Probleme und Wünsche[1] in einer Version 1.1.0 der Anwendung, welche am 16. Februar 2022 veröffentlicht wurde, behoben.

Auch die Konzeption der Anwendung kann erfolgreich bewertet werden, da nur wenige Ideen verworfen werden mussten (vgl. Abschnitt 3.6) und die durchgeführten Nutzertests sowie die technischen Tests alle weiteren Konzepte und gestellten Anforderungen abdecken.

Bei der entwickelten Software handelt es sich um die erste leicht bedienbare Nutzeroberfläche für DATASIM. Damit hat diese Arbeit eine Lücke in der Liste verfügbarer *Tools* im *E-Learning* geschlossen.

6.1.1. Einschränkungen und Offengebliebenes

Wie in den vorherigen Kapiteln festgestellt wurde, bietet DATASIM derzeit nicht die Möglichkeit, xAPI-Statements für vorgegebene xAPI-Aktivitäten (vgl. Unterabschnitt 2.4.1) zu erzeugen. Weiterhin ist es nicht möglich, Alignments für xAPI-Verben anzulegen.[1] Eine alternative Identifikationsmöglichkeit für xAPI-*Actors* (über E-Mail-Adressen hinaus) ist nicht dokumentiert[7].

Im Validierungsplan musste festgelegt werden, dass auf automatisierte Tests aus Zeitgründen weitestgehend verzichtet werden soll. Dadurch sind Regressionen bei der Fortentwicklung der Anwendung nicht auszuschließen.

Es konnte im Umfang dieser Arbeit nur implementiert werden, dass auf bereits existierende Simulationsprofile zurückgegriffen wird. Damit die generierten Szenarien etwas realitätsnäher oder gezielter gestaltet werden können wäre ein Editor sinnvoll.

[1] Insgesamt 18 von 20 Entwicklungsaufgaben

6.2. Nächste Schritte

Mit der hier umgesetzten Anwendung wurde eine solide Grundlage für die weitere Forschung und Entwicklung in diesem Bereich geschaffen. Im unmittelbaren Anschluss können die simulierten xAPI-Statements grafisch aufbereitet und/oder statistisch untersucht werden. Aus den (gegebenenfalls manuell beschriebenen) Ableitungen aus den Testdaten können Modelle für Lernassistenzsysteme berechnet werden, welche schlussendlich die Lernenden in ihrer individuellen Situation unterstützen können.

6.2.1. Visualisierung der erstellten Daten

In Kapitel 1 und Kapitel 2 wurden der Fokus und die Datenanforderungen von Lernassistenzsystemen an die Total Learning Architecture erörtert. Insbesondere wurde dort festgehalten, dass es *gelabelter* Datenmengen für das Training von *Machine Learning*-basierten Verfahren bedarf. Mit dieser Anwendung können im großen Stil *ungelabelte* Datensätze erzeugt werden, welche dann programmatisch oder manuell bewertet werden können.

Um dieses Vorhaben zu erleichtern und einen ersten Eindruck der simulierten Lernfortschritte zu erhalten, eignen sich Visualisierungen. Diese lassen sich auf verschiedene Art und Weise erzeugen, wie im Folgenden beschrieben wird.

LRS-interne Visualisierungstools

Der im Forschungsprojekt „VerDatAs“ und für die Tests verwendete Learning Record Store *Learning Locker*[2] bietet voreingestellte Möglichkeiten zur Datenvisualisierung, welche jedoch wenig Flexibilität aufweisen. So ist die statistische Auswertung der Daten in den mitgelieferten *Dashboards* nicht auf einen einzelnen Datenspeicher einschränkbar, sondern wird immer für den globalen Datenstand (inklusive gelöschter Daten) ausgeführt[3]. Ein solches *Dashboard* ist in Abbildung 6.1 dargestellt.

Außerdem bietet *Learning Locker* die Möglichkeit, eigene Diagrammtypen zu erzeugen[34]. Diese sind jedoch an eine spezifische Installation gebunden und nicht übertragbar.

DAVE

Für die ADL wurde von *Yet Analytics*, dem Entwickler von DATASIM, eine Anwendung zur Visualisierung von xAPI-Statements entwickelt. Diese weist viele Freiheitsgrade durch eine freie Konfiguration auf, sodass sie jede erdenkliche grafische Auswertung durchführen kann. Ylvi Sarah Bachmann setzt sich in ihrer derzeit in Bearbeitung befindlichen Bachelorarbeit „Prototypische Umsetzung und Evaluation eines Systems zur Visualisierung von Lernstands- und Lernverlaufsdaten“ mit einer Integration von DAVE in die hier entwickelte Anwendung auseinander.

6.2.2. Weitere Anwendungen

In einem Beitrag zur *International Conference on Education and New Learning Technologies* betrachten Forscherinnen der Universität Sofia diverse Möglichkeiten der Datenauswertung im E-Learning-Kontext. Dort werden insbesondere die Programme *Gismo* und *VisMOOC* hervorgehoben. Weiterhin referenzieren die Autorinnen das von der Europäischen Union geförderte Forschungsprojekt *weSPOT*.[36]

[2] `https://learningpool.com/solutions/learning-locker-community-overview/` – Zugriff: 11.02.2022
[3] Das Verhalten wurde empirisch festgestellt.

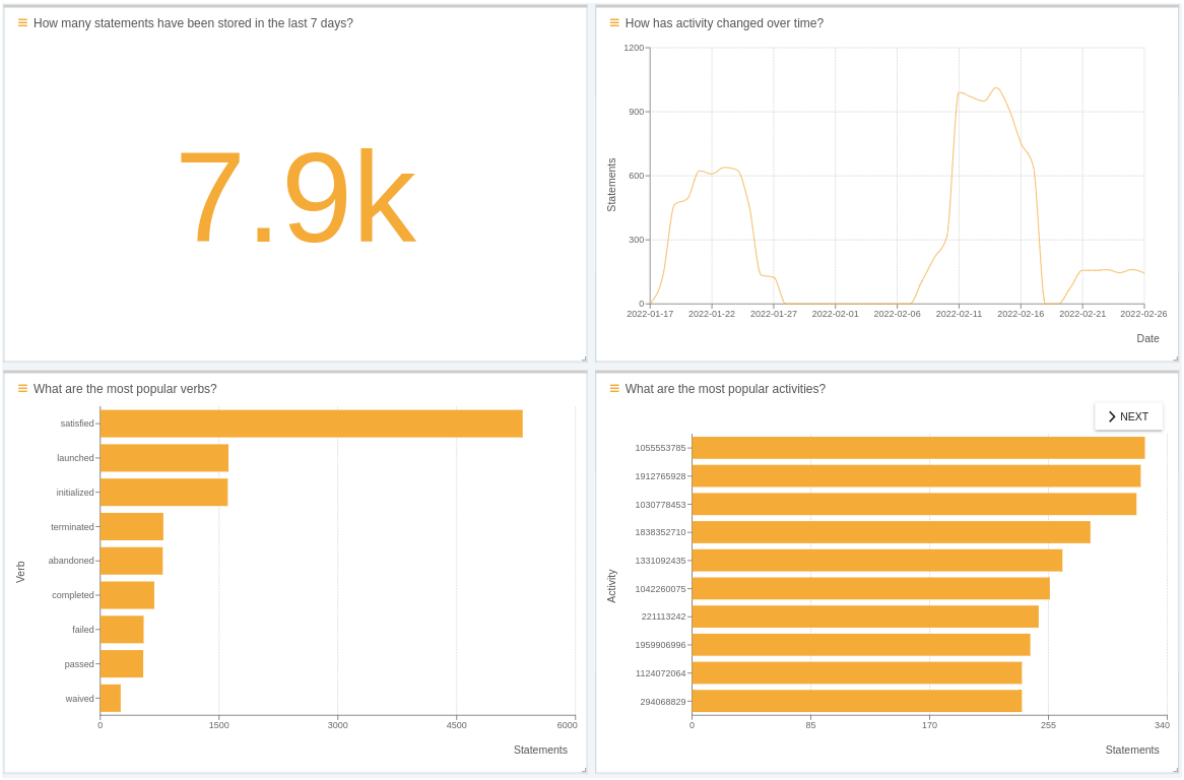


Abbildung 6.1.: Standard-Visualisierungsdashboard (Auszug) in Learning Locker

6.2.3. Weiterführende Themen

Aufbauend auf diese Arbeit könnten die folgenden Aufgabenstellungen näher untersucht werden:

- Implementierung eines Editors zur Erstellung und Anpassung von DATASIM-Profilen
- Analyse und Implementierung eines Werkzeugs zum Labeln von Trainingsdaten für Lernassistenzsysteme
  (Danach: Entwicklung eines *Machine Learning*-Modells für ein Lernassistenzsystem)

A. Anhang

Quelltext A.1: Compose-Datei für Deployment

```
---
version: "3"
networks:
  # Create custom Network Namespace
  xapitools0: {}
volumes:
  # Cache for Simulation results
  xapitools_simulations: {}

services:
  # Database
  mariadb:
    image: library/mariadb:10.7
    networks:
      - xapitools0
    environment:
      MARIADB_RANDOM_ROOT_PASSWORD: 1
      # Seed DB and user
      MARIADB_DATABASE: xapitools
      MARIADB_USER: xapi
      MARIADB_PASSWORD: toolkit
  # DATASIM
  datasim:
    restart: unless-stopped
    # Image is built by CI of DATASIM "mirror"
    image: nexus.galaxion.de:5050/yetanalytics/datasim:latest
    # The default is to run DATASIM in CLI mode.
    # We need to set this to run in Server mode.
    entrypoint: bin/server.sh
    networks:
      - xapitools0
    environment:
      credentials: foo:bar  # sic!
    expose:
      - "9090"
```

```
  permission_control:
    image: busybox
    # The Docker Volume is created with root access only.
    # The Spring Application will run with id 1000:1000,
    # so we have to enable access to the volume manually.
    command: /bin/sh -c 'touch /data/.initialized && chown -R 1000:1000
        /data'
    volumes:
      - xapitools_simulations:/data
  xapitools:
    depends_on:
      - mariadb
      - datasim
      - permission_control
    restart: on-failure
    # Image is built by CI of this project.
    image:
        nexus.galaxion.de:5050/de.tudresden.inf.verdatas/xapitools:latest
    networks:
      - xapitools0
    volumes:
      - xapitools_simulations:/data
    ports:
      # Map container port 8080 to local port 80
      - "80:8080"
    environment:
      XAPITOOLS_SEC_USERNAME: xapi
      XAPITOOLS_SEC_PASSWORD: tools
      # Docker-relative path
      XAPITOOLS_SIM_BACKEND_BASE_URL: http://datasim:9090
      XAPITOOLS_SIM_BACKEND_USERNAME: foo
      XAPITOOLS_SIM_BACKEND_PASSWORD: bar
      # Docker-relative path
      XAPITOOLS_SIM_STORAGE_DIR: /data
      # Docker-relative path
      XAPITOOLS_DB_CONNECTION_STRING: mariadb://mariadb:3306/xapitools
      XAPITOOLS_DB_CONNECTION_USER: xapi
      XAPITOOLS_DB_CONNECTION_PASSWORD: toolkit
      TZ: Europe/Berlin
```

A.1. Vorlage für die Durchführung der Nutzertests

Tabelle A.1.: Vorlage für die Durchführung der Nutzertests

Testfall	Bewertung		Kommentar
Testfall 1			
Erwartung 1	☐ OK	☐ NOK	
Erwartung 2	☐ OK	☐ NOK	
Erwartung 3	☐ OK	☐ NOK	
Erwartung 4	☐ OK	☐ NOK	
Erwartung 5	☐ OK	☐ NOK	
Erwartung 6	☐ OK	☐ NOK	
Erwartung 7	☐ OK	☐ NOK	
gesamt	☐ OK	☐ NOK	
Testfall 2			
Erwartung 1	☐ OK	☐ NOK	
Erwartung 2	☐ OK	☐ NOK	
Erwartung 3	☐ OK	☐ NOK	
Erwartung 4	☐ OK	☐ NOK	
Erwartung 5	☐ OK	☐ NOK	
Erwartung 6	☐ OK	☐ NOK	
Erwartung 7	☐ OK	☐ NOK	
Erwartung 8	☐ OK	☐ NOK	
Erwartung 9	☐ OK	☐ NOK	
Erwartung 10	☐ OK	☐ NOK	
Erwartung 11	☐ OK	☐ NOK	
Erwartung 12	☐ OK	☐ NOK	
Erwartung 13	☐ OK	☐ NOK	
Erwartung 14	☐ OK	☐ NOK	
Erwartung 15	☐ OK	☐ NOK	
Erwartung 16	☐ OK	☐ NOK	
Erwartung 17	☐ OK	☐ NOK	
gesamt	☐ OK	☐ NOK	

Fortsetzung in Tabelle A.2

Tabelle A.2.: Fortsetzung: Vorlage für die Durchführung der Nutzertests

Testfall	Bewertung		Kommentar
Testfall 3			
Erwartung 1	☐ OK	☐ NOK	
Erwartung 2	☐ OK	☐ NOK	
Erwartung 3	☐ OK	☐ NOK	
Erwartung 4	☐ OK	☐ NOK	
Erwartung 5	☐ OK	☐ NOK	
Erwartung 6	☐ OK	☐ NOK	
gesamt	☐ OK	☐ NOK	
Testfall 4			
Erwartung 1	☐ OK	☐ NOK	
Erwartung 2	☐ OK	☐ NOK	
Erwartung 3	☐ OK	☐ NOK	
gesamt	☐ OK	☐ NOK	
Testfall 5			
Erwartung 1	☐ OK	☐ NOK	
Erwartung 2	☐ OK	☐ NOK	
Erwartung 3	☐ OK	☐ NOK	
Erwartung 4	☐ OK	☐ NOK	
Erwartung 5	☐ OK	☐ NOK	
Erwartung 6	☐ OK	☐ NOK	
Erwartung 7	☐ OK	☐ NOK	
Erwartung 8	☐ OK	☐ NOK	
Erwartung 9	☐ OK	☐ NOK	
Erwartung 10	☐ OK	☐ NOK	
Erwartung 11	☐ OK	☐ NOK	
Erwartung 12	☐ OK	☐ NOK	
Erwartung 13	☐ OK	☐ NOK	
Erwartung 14	☐ OK	☐ NOK	
Erwartung 15	☐ OK	☐ NOK	
Erwartung 16	☐ OK	☐ NOK	
gesamt	☐ OK	☐ NOK	
Testfall 6			
Erwartung 1	☐ OK	☐ NOK	
Erwartung 2	☐ OK	☐ NOK	
gesamt	☐ OK	☐ NOK	

Fortsetzung in Tabelle A.3

Tabelle A.3.: Fortsetzung: Vorlage für die Durchführung der Nutzertests

Kriterium	Bewertung (Schulnoten)	Kommentar
Waren die Schaltflächenbezeichnungen für Sie intuitiv verständlich?		
Sind Ihnen sprachliche Fehler in der Anwendung aufgefallen?		
Wussten Sie während der Testausführung, welche Schritte durchlaufen werden?		
Wie logisch erscheint Ihnen die Schrittreihenfolge?		
Lesen Sie die Nutzerdokumentation. Wie verständlich ist es für Sie, wie die einzelnen Vorgänge der Anwendung benutzt werden können?		

Weitere Bemerkungen:

A.2. Testprotokolle

A.2.1. Testprotokoll von Tommy Kubica

A.1. Vorlage für die Durchführung der Nutzertests

Tabelle A.1.: Vorlage für die Durchführung der Nutzertests

Testfall	Bewertung		Kommentar
Testfall 1			
Erwartung 1	☑ OK	☐ NOK	
Erwartung 2	☑ OK	☐ NOK	
Erwartung 3	☑ OK	☐ NOK	
Erwartung 4	☑ OK	☐ NOK	
Erwartung 5	☑ OK	☐ NOK	
Erwartung 6	☑ OK	☐ NOK	
Erwartung 7	☑ OK	☐ NOK	
gesamt	☑ OK	☐ NOK	
Testfall 2			
Erwartung 1	☑ OK	☐ NOK	Ein hover-Effekt für die Buttons könnte hilfreich sein.
Erwartung 2	☑ OK	☐ NOK	
Erwartung 3	☑ OK	☐ NOK	Der Klick auf den Karten-Body ist unüblich, was jedoch nicht schlimm ist.
Erwartung 4	☑ OK	☐ NOK	Der Save-Button lädt ewig weiter, obwohl der Fehler angezeigt wurde.
Erwartung 5	☑ OK	☐ NOK	
Erwartung 6	☑ OK	☐ NOK	
Erwartung 7	☑ OK	☐ NOK	
Erwartung 8	☑ OK	☐ NOK	"Connection Parameters" könnte als Überschrift (h2/h3) dargestellt werden.
Erwartung 9	☑ OK	☐ NOK	
Erwartung 10	☑ OK	☐ NOK	
Erwartung 11	☑ OK	☐ NOK	
Erwartung 12	☑ OK	☐ NOK	
Erwartung 13	☑ OK	☐ NOK	
Erwartung 14	☑ OK	☐ NOK	
Erwartung 15	☑ OK	☐ NOK	
Erwartung 16	☑ OK	☐ NOK	Durch das Lade-Icon wird der Button "Deactive" auf zwei Zeilen umgebrochen.
Erwartung 17	☑ OK	☐ NOK	
gesamt	☑ OK	☐ NOK	

Fortsetzung in Tabelle A.2

Tabelle A.2.: Fortsetzung: Vorlage für die Durchführung der Nutzertests

Testfall	Bewertung		Kommentar
Testfall 3			
Erwartung 1	☑ OK	☐ NOK	
Erwartung 2	☑ OK	☐ NOK	
Erwartung 3	☑ OK	☐ NOK	Ohne Auswahl einer Datei lädt der "Send" Button wieder ewig nach Fehlerm.
Erwartung 4	☑ OK	☐ NOK	
Erwartung 5	☑ OK	☐ NOK	
Erwartung 6	☑ OK	☐ NOK	
gesamt	☑ OK	☐ NOK	
Testfall 4			
Erwartung 1	☑ OK	☐ NOK	
Erwartung 2	☑ OK	☐ NOK	Neuladen ein wenig unintuitiv (auch zuvor); Reconnect bei "Reactive" verwirrt. Evtl. eine Toast-Message ergänzen, dass reconnected wurde.
Erwartung 3	☑ OK	☐ NOK	
gesamt	☑ OK	☐ NOK	
Testfall 5			
Erwartung 1	☑ OK	☐ NOK	
Erwartung 2	☑ OK	☐ NOK	Die Art der Eingabe des Titels verwirrt. Inkonsistent zu vorherigen Inputs.
Erwartung 3	☐ OK	☑ NOK	Bei mir wird trotz Cancel der Button links gewechselt -> Ansicht weg Zudem wird z.B. im zweiten Schritt (xAPI Profile) etc. keine Warning gegeben
Erwartung 4	☑ OK	☐ NOK	
Erwartung 5	☑ OK	☐ NOK	Persona Mail Placeholder overflow bei geringer Weite Das "Verhalten" dieses Input-Felds ist anders (Placeholder nicht oben)
Erwartung 6	☑ OK	☐ NOK	
Erwartung 7	☑ OK	☐ NOK	
Erwartung 8	☑ OK	☐ NOK	
Erwartung 9	☑ OK	☐ NOK	
Erwartung 10	☑ OK	☐ NOK	Die Anzeige von Schritten bzw. eines Werts für die Regler wäre hilfreich.
Erwartung 11	☑ OK	☐ NOK	Die Icons könnten mit Tooltips versehen werden.
Erwartung 12	☑ OK	☐ NOK	
Erwartung 13	☑ OK	☐ NOK	
Erwartung 14	☑ OK	☐ NOK	
Erwartung 15	☑ OK	☐ NOK	
Erwartung 16	☑ OK	☐ NOK	
gesamt	☑ OK	☐ NOK	
Testfall 6			
Erwartung 1	☑ OK	☐ NOK	Vor "Retrieve" ist ein "Perform" notwendig. Bei Perform wäre die Angabe einer ungefähren Dauer hilfreich.
Erwartung 2	☑ OK	☐ NOK	
gesamt	☑ OK	☐ NOK	

Fortsetzung in Tabelle A.3

Tabelle A.3.: Fortsetzung: Vorlage für die Durchführung der Nutzertests

Kriterium	Bewertung (Schulnoten)	Kommentar
Waren die Schaltflächenbezeichnungen für Sie intuitiv verständlich?	2	z.B. "Set Remark", aber es ist laut Input der Title
Sind Ihnen sprachliche Fehler in der Anwendung aufgefallen?		Lediglich Auffälligkeiten beim Casing, z.B. "Also show inactive Connections" oder "Create new Connection"
Wussten Sie während der Testausführung, welche Schritte durchlaufen werden?	2	-
Wie logisch erscheint Ihnen die Schrittreihenfolge?	1	-
Lesen Sie die Nutzerdokumentation. Wie verständlich ist es für Sie, wie die einzelnen Vorgänge der Anwendung benutzt werden können?	1	-

Weitere Bemerkungen:

- Personaes können nicht gelöscht/editiert werden, wodurch Vertippen schlecht ist.
- ein refresh-Button könnte hilfreich sein.
- Tooltips für Icons hinzufügen, z.B. auch Download/Export oder Edit
- "inactive connections" sollten löschbar sein

A.2.2. Testprotokoll von Robert Schmidt

A.1. Vorlage für die Durchführung der Nutzertests

Tabelle A.1.: Vorlage für die Durchführung der Nutzertests

Testfall	Bewertung		Kommentar
Testfall 1			
Erwartung 1	☐ OK	☐ NOK	
Erwartung 2	☐ OK	☐ NOK	
Erwartung 3	☐ OK	☐ NOK	
Erwartung 4	☐ OK	☐ NOK	Passwortabfragefenster erscheint erneut
Erwartung 5	☐ OK	☐ NOK	
Erwartung 6	☐ OK	☐ NOK	
Erwartung 7	☐ OK	☐ NOK	
gesamt	☐ OK	☐ NOK	
Testfall 2			
Erwartung 1	☐ OK	☐ NOK	
Erwartung 2	☐ OK	☐ NOK	
Erwartung 3	☐ OK	☐ NOK	Save-Button hat ständiges Lade-Symbol
Erwartung 4	☐ OK	☐ NOK	
Erwartung 5	☐ OK	☐ NOK	
Erwartung 6	☐ OK	☐ NOK	
Erwartung 7	☐ OK	☐ NOK	
Erwartung 8	☐ OK	☐ NOK	
Erwartung 9	☐ OK	☐ NOK	
Erwartung 10	☐ OK	☐ NOK	
Erwartung 11	☐ OK	☐ NOK	
Erwartung 12	☐ OK	☐ NOK	
Erwartung 13	☐ OK	☐ NOK	
Erwartung 14	☐ OK	☐ NOK	
Erwartung 15	☐ OK	☐ NOK	
Erwartung 16	☐ OK	☐ NOK	
Erwartung 17	☐ OK	☐ NOK	
gesamt	☐ OK	☐ NOK	

Fortsetzung in Tabelle A.2

Tabelle A.2.: Fortsetzung: Vorlage für die Durchführung der Nutzertests

Testfall	Bewertung		Kommentar
Testfall 3			
Erwartung 1	☐ OK	☐ NOK	
Erwartung 2	☐ OK	☐ NOK	
Erwartung 3	☐ OK	☐ NOK	
Erwartung 4	☐ OK	☐ NOK	
Erwartung 5	☐ OK	☐ NOK	
Erwartung 6	☐ OK	☐ NOK	Anzahl bleibt gleich, immer 625 Statements
gesamt	☐ OK	☐ NOK	
Testfall 4			
Erwartung 1	☐ OK	☐ NOK	
Erwartung 2	☐ OK	☐ NOK	
Erwartung 3	☐ OK	☐ NOK	
gesamt	☐ OK	☐ NOK	
Testfall 5			
Erwartung 1	☐ OK	☐ NOK	
Erwartung 2	☐ OK	☐ NOK	
Erwartung 3	☐ OK	☐ NOK	
Erwartung 4	☐ OK	☐ NOK	
Erwartung 5	☐ OK	☐ NOK	
Erwartung 6	☐ OK	☐ NOK	
Erwartung 7	☐ OK	☐ NOK	
Erwartung 8	☐ OK	☐ NOK	
Erwartung 9	☐ OK	☐ NOK	
Erwartung 10	☐ OK	☐ NOK	
Erwartung 11	☐ OK	☐ NOK	
Erwartung 12	☐ OK	☐ NOK	
Erwartung 13	☐ OK	☐ NOK	
Erwartung 14	☐ OK	☐ NOK	
Erwartung 15	☐ OK	☐ NOK	
Erwartung 16	☐ OK	☐ NOK	
gesamt	☐ OK	☐ NOK	
Testfall 6			
Erwartung 1	☐ OK	☐ NOK	es muss ersteinmal auf Schaltfläche Perform geklickt werden
Erwartung 2	☐ OK	☐ NOK	
gesamt	☐ OK	☐ NOK	

Fortsetzung in Tabelle A.3

Tabelle A.3.: Fortsetzung: Vorlage für die Durchführung der Nutzertests

Kriterium	Bewertung (Schulnoten)	Kommentar
Waren die Schaltflächenbezeichnungen für Sie intuitiv verständlich?	1	Schaltflächen deutlich und klar beschriftet
Sind Ihnen sprachliche Fehler in der Anwendung aufgefallen?	1	keine sprachlichen Fehler in der Anwendung
Wussten Sie während der Testausführung, welche Schritte durchlaufen werden?	2	nicht immer deutliche Kommunikation der einzelnen Schritte
Wie logisch erscheint Ihnen die Schrittreihenfolge?	1	
Lesen Sie die Nutzerdokumentation. Wie verständlich ist es für Sie, wie die einzelnen Vorgänge der Anwendung benutzt werden können?	1	

Weitere Bemerkungen:

Rechtschreibfehler in Nutzerdokumentation: Seite 4, Exporting DATASIM simulation results: "to" -> "two"

A.2.3. Testfall 8 (DATASIM-REST-Schnittstelle)

Quelltext A.2: Mitschnitt des Sendens der Simulationsbeschreibung an DATASIM

```
Hypertext Transfer Protocol
    POST /api/v1/generate HTTP/1.1
    Authorization: Basic Zm9vOmJhcg==
    Accept: text/plain, application/json, application/*+json, */*
    Content-Type: multipart/form-data;boundary=<...>
    User-Agent: Java/17.0.1
    Host: localhost:9090
    Connection: keep-alive
    Content-Length: 21940

MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary:
   "<...>"
    [Type: multipart/form-data]
    First boundary: --<...>
    Encapsulated multipart part:  (application/json)
        Content-Disposition: form-data; name="personae-array"
        Content-Type: application/json
        JavaScript Object Notation: application/json
    Boundary: --<...>
    Encapsulated multipart part:  (application/json)
        Content-Disposition: form-data; name="profiles"
        Content-Type: application/json
        JavaScript Object Notation: application/json
    Boundary: --<...>
    Encapsulated multipart part:  (application/json)
        Content-Disposition: form-data; name="alignments"
        Content-Type: application/json
        JavaScript Object Notation: application/json
    Boundary: --<...>
    Encapsulated multipart part:  (application/json)
        Content-Disposition: form-data; name="parameters"
        Content-Type: application/json
        JavaScript Object Notation: application/json
    Last boundary: --<...>--
```

A.2.4. Testfall 9 (LRS-Kommunikation sendend)

Quelltext A.3: Mitschnitt des Sendens von xAPI-Statements an einen LRS

```
Hypertext Transfer Protocol
    POST /data/xAPI/statements HTTP/1.1
    Authorization: Basic <...>
    Accept: application/json, application/*+json
    Content-Type: application/json
    X-Experience-API-Version: 1.0.3
    User-Agent: Java/17.0.1
    Host: ba-lrs.galaxion.de
    Connection: keep-alive
    Content-Length: 491750

JavaScript Object Notation: application/json
    Array
        Object
            Member Key: id
            Member Key: actor
            Member Key: verb
            Member Key: object
            Member Key: context
            Member Key: timestamp
        Object
            ...
        ...
```

A.3. Nutzerdokumentation (engl.)

User's Guide

Project Title	Version	Date	Author
xAPI Toolkit - DATASIM	v1	08.02.2022	Konstantin Köhring (@Galaxy102)

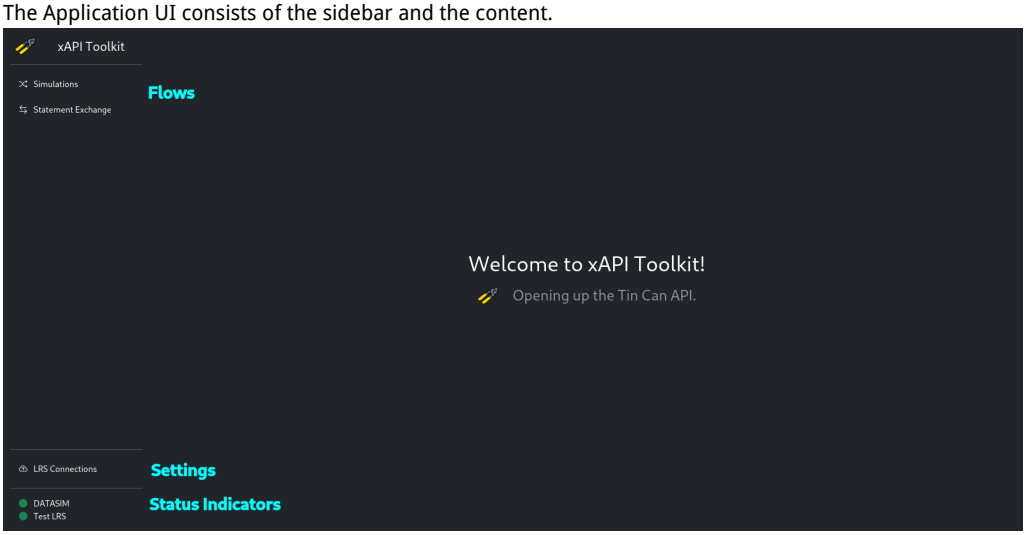
Intro

Hello there, thank you for your interest in using the xAPI Toolkit, which has been begun as a Bachelor Thesis project for the research group "VerDatAs" at the Technical University of Dresden.

To make your work as comfortable as possible, please use this manual as a reference guide when you don't find the application to be intuitive enough.

Application Overview

The Application UI consists of the sidebar and the content.



In the Sidebar, there are three sections (from top to bottom):

1. Flows
   They represent means to handle xAPI Statements.
2. Settings
   There you can manipulate e.g. connection parameters.
3. Status indicators
   At a quick glance, you can see which system components are healthy and which ones may need attention.
   A green indicator means that everything should be alright.
   Red indicates a remote server failure or a configuration mistake.
   Yellow means that there was an application error which can perhaps be fixed by reloading the application by hitting Refresh or F5.

You can access all Settings and Flows at any time by clicking the corresponding sidebar item.

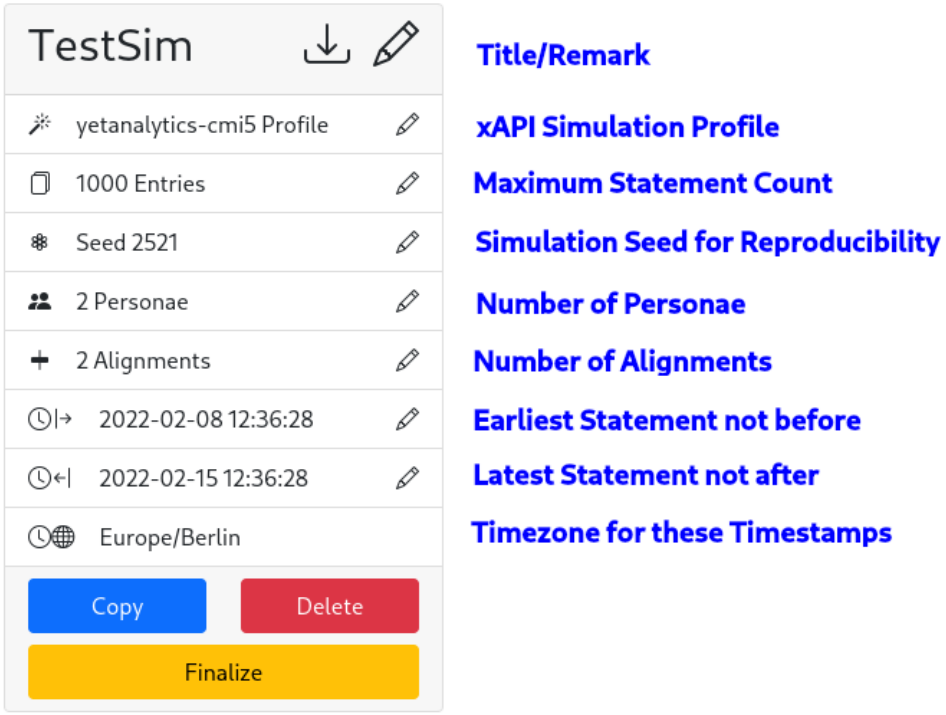
Step-by-Step Guides: Flows

Perform Statement Simulations using DATASIM

For all guides below, please select the appropriate flow by clicking `Simulations` in the sidebar.

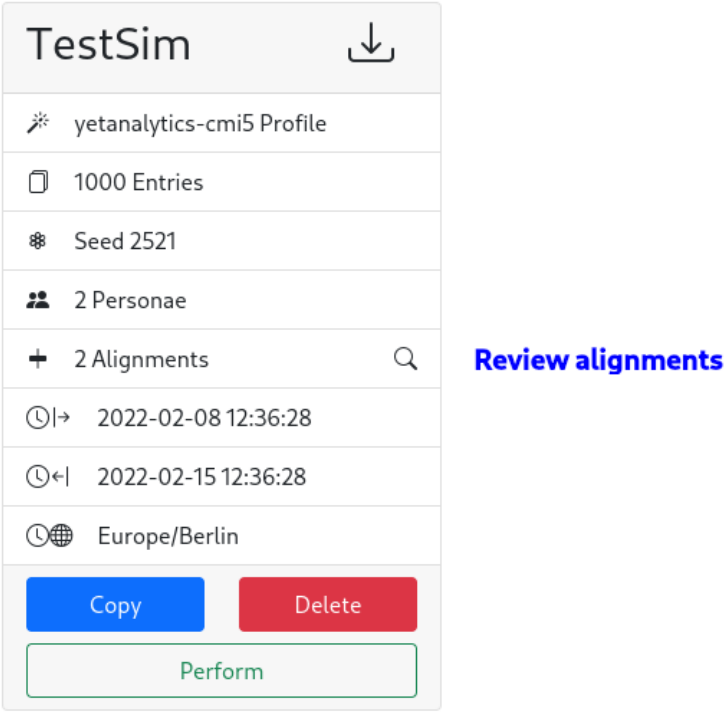
Anatomy of the Simulation UI and nice to know:

- Prepared Simulations
  These simulations' parameters can be edited by clicking the pencils on the corresponding items.



  By clicking `Finalize`, the simulation becomes immutable.
  To download the simulation description file (in DATASIM format), click the download icon near the title.

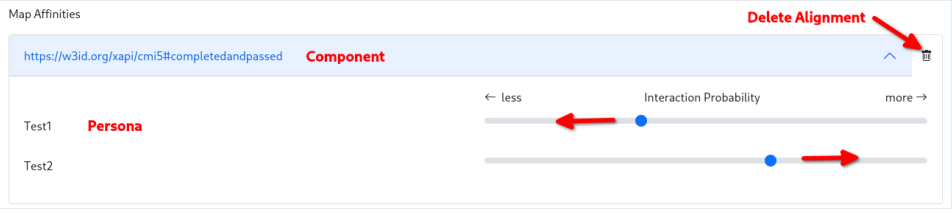
- Finalized Simulations
  These simulations have been made immutable. To adapt a simulation, click `Copy` to create a decoupled version.



Parametrize a DATASIM simulation

1. Click the `Create new Simulation` card.
2. Set a Simulation Title.
   It will be used for displaying the simulation and for naming the output files.
3. Set a Simulation Profile.
   The profiles shipped with the application have been written by Yet Analytics, the developers of DATASIM.
   They each represent a schema after which statements will be generated.
4. Create and select Personae.
   A persona has a name and an email address (used as unique identifier). To add a persona, insert those details into the `Add new Persona` form. If you're to lazy to imagine an email address, hit the "refresh"-ish button on the right-hand side of the input field. Press `+` to add the persona to the simulation.
   Personae used in the simulation can be selected in the bottom form. Multiple Personae can be selected by holding down `Shift` or `Ctrl` as usual.
5. Create alignments.
   Alignments can be used to control the affinity of personae to a specific simulation element. For now, DATASIM does not allow aligning to xAPI Verbs.
   First, add all components you want to align to by selecting their type and identifiers from the dropdown menus in the `Add new Component` form and hitting the `+` button.
   Afterwards, you can insert the affinities by expanding the component (`v` button) and moving the

sliders around as you wish.



6. Update the default simulation parameters.
   Adapt them as you wish.
   Number of statements is a maximum value. The seed is used to guarantee reproducibility. Simulation Start and End are the earliest and latest possible timestamp for generated xAPI Statements.

Editing a DATASIM simulation parametrization

1. On the card with the **non-finalized** simulation you want to edit, click the pencil icon on the part you want to change.
2. Proceed with the corresponding step of the section "Parametrize a DATASIM simulation".

Making a simulation description read-only

1. On the card with the simulation you want to finalize, click the `Finalize` button.

Creating a copy of a simulation description

1. On the card with the simulation you want to copy, click the `Copy` button.
   This will create a decoupled and non-finalized version of the simulation.

Performing a DATASIM simulation

1. Perform the steps in "Parametrize a DATASIM simulation" and "Making a simulation description read-only".
2. Click `Perform` on the card with the simulation you want to send to DATASIM.

Exporting DATASIM simulation results

1. After "Performing a DATASIM simulation",
2. You have to options to handle the result:
    1. You can retrieve the simulation result as JSON document by clicking `Retrieve` on the card of the simulation you want to get the result of.
    2. You can also send the result directly to an LRS by clicking `Push` and then selecting the destination LRS.

Deleting a simulation description

1. On the card with the simulation you want to delete, click the `Delete` button.
   This action can not be undone.

Import and Export LRS Data

For all guides below, please select the appropriate flow by clicking `Statement Exchange` in the sidebar.

Importing xAPI Statements

1. Select the LRS you want to send the statements to from the Dropdown in the `Import from JSON` card.
   It must be enabled.
2. Select one or more file(s) of xAPI Statements using the `Choose Files` form.
3. Hit the `Send` button.
   If the file was valid, you will receive a success message. Otherwise, an error will be displayed stating the issue with the upload.

Exporting xAPI Statements

1. Select the LRS you want to send the statements to from the Dropdown in the `Import from JSON` card.
   It must be enabled.
2. Initiate the download by clicking `Retrieve`.
   The process may take a few seconds depending on the amount of saved statements in the selected LRS.

Step-by-Step Guides: Settings

Managing LRS Connections

For all guides below, please access the appropriate setting by clicking `LRS Connections` in the sidebar.

Adding a connection

1. Click the Card `Create new Connection`.
2. Enter the Details of the LRS.
   They are usually shown to you by your Learning Record Store.
   The Name field will be used for the status indicator and to reference the LRS in flows.
3. Click `Save`.
4. To display the status indicator in the sidebar, reload the Application by hitting Refresh or `F5`.

Editing a connection

1. On the card of the LRS connection you want to edit, press the `Edit` button.
2. Update the Details of the LRS.
3. Click `Save`.
4. To update the status indicator in the sidebar, reload the Application by hitting Refresh or `F5`.
   The indicator may be flaky or yellow at first, but becomes stable after at most 2 minutes.

Deactivating a connection

1. On the card of the LRS connection you want to disable, press the `Deactivate` button.
2. To remove the status indicator from the sidebar, reload the Application by hitting Refresh or `F5`. Otherwise, the indicator will turn yellow after some time.

Reactivating a connection

1. Click `Also show inactive Connections` on the first card.
2. On the card of the LRS connection you want to re-enable, press the `Reactivate` button.
3. To display the status indicator in the sidebar, reload the Application by hitting Refresh or `F5`.

A.4. Entwicklerdokumentation (engl.)

Developer's Guide

Project Title	Version	Date	Author
xAPI Toolkit - DATASIM	v1	27.01.2022	Konstantin Köhring (@Galaxy102)

Intro

Hello there, thank you for your interest in maintaining or extending the xAPI Toolkit,
which has been begun as a Bachelor Thesis project for the research group "VerDatAs"
at the Technical University of Dresden.

To make your work as comfortable as possible, the POM should be able to cover all CI steps.
Just load it into your IDE (I have used IntelliJ for development), set the Application Profile to `dev`
and think of great values for the variables in application-dev.properties.

For my setup, I have chosen the following parameters:

Environment Variable	Value
XAPITOOLS_SEC_USERNAME	foo
XAPITOOLS_SEC_PASSWORD	bar
XAPITOOLS_SIM_BACKEND_BASE_URL	http://localhost:9090
XAPITOOLS_SIM_BACKEND_USERNAME	foo
XAPITOOLS_SIM_BACKEND_PASSWORD	bar
XAPITOOLS_SIM_STORAGE_DIR	/tmp/xapitools

Their semantic is explained in the README.

Apart from running and debugging your application from the IDE, you can build a Docker Image for it by
running `./mvnw (or your OS equivalent) spring-boot:build-image`.
Usually from your CI, by providing `-Ddocker.publish=true` you can also push the container image to
the registry specified in the POM.

The external service dependencies can be started with `docker-compose up -d (datasim|whatever)`,
see the Compose file for details.

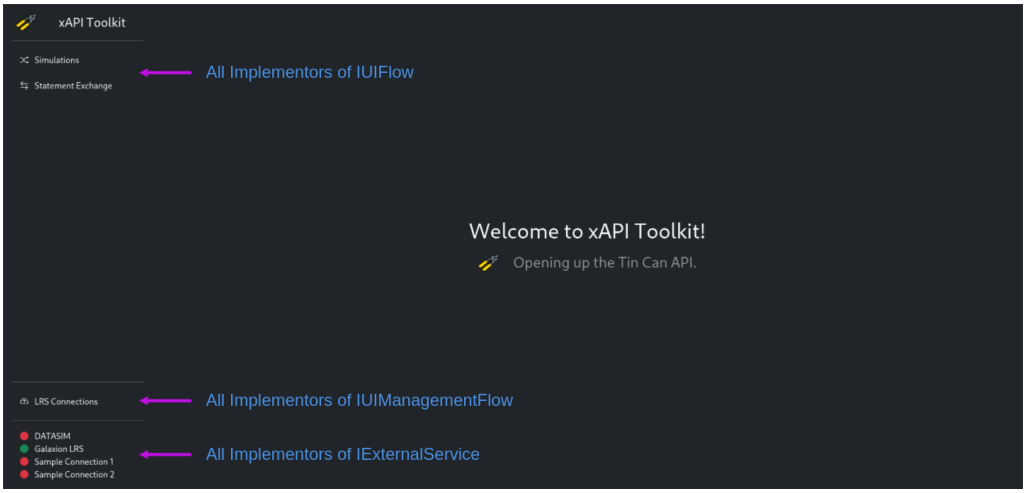
Application Overview

The Application consists roughly of a UI Controller (`ui.BasepageMavController`), a Security Handler
(`security` Package) and Sub-Applications.

Sub-Applications shall reside in their own packages.
The UI distinguishes Sub-Applications to either be a "Flow" (a real, standalone application) or a "Setting"
(perhaps Connection Settings).
I'd like future programmers to adhere to a separation of concerns:

Any "Flow" may access any "Setting", but "Settings" may not use any "Flows" and "Flows" shall use each other as little as possible.
An Example in this context would be:
I have some LRS connection (a "Setting").
It may be used by any "Flow" that needs to communicate with an LRS through a Connector (which is part of the Setting), but must not be mutated in any way outside the Setting.

Furthermore, the UI is bound automagically:



To learn about this in detail, you are advised to study the Javadoc (`./mvnw javadoc:javadoc`) and Spring Dependency Injection.

Step-by-Step Guides

Maintaining the Application

So you have found a Bug or want to do some routine maintenance.
You can probably navigate to the required code by searching the Logs for details (mind the exception markers aka "Tracing Codes").

1. Isolate the erroneous Class
   Find exactly the point in the Application where something goes wrong by examining logs and Data Flows (you would begin with the Controllers, then Services and Connectors and finally the Entities).
   By using semantic package and class names, you should be just fine to navigate as you think.
2. Fix whatever needs fixing
   Be as precise as possible when changing the Source Code and **comment your changes**, especially what did go wrong and -if you know- why.
   Try to find a way to inhibit invalid control paths (e.g. by adding Validators).
3. Give credits to yourself and your sources
   If you found the solution to your problem online, write a comment with your source. Feel free to add yourself to the Author section of the Class's Javadoc.
4. Bump the version
   I'd consider it good practice increasing the Version Number of the Project (in the POM).

Extending the Application

The application was designed with Extensibility in mind.

You can create your new sub-application independently of existing code with the only coupling points being the UI templates (optionally).

1. Write your Application as you would normally do.
   Make sure to seed Entities only in `dev`-Profile.
   Separate your Settings from your Main Program!
   To simplify the integration later, please also consider the following separation in your Controllers:
   - Separate RestControllers from UI Controllers
   - The Main UI ("Entrypoint") of your Application shall reside in one Standalone Controller.
   - (only non-Settings) All Sub-UIs (Forms, Detail Pages) shall have their very own Controller.
     Please also try to extract any external Logic (Application or Library Integrations) to standalone Connectors.
     If you want to use Bootstrap and jQuery for your UI, you are all set by replacing your HTML's `title` tag with
     `<head th:include="bootstrap/header.html :: scripted_head ('<YOUR PAGE TITLE HERE>')"></head>`
2. Make your Controllers implement the UI Magic-Binding Interfaces
   - The Main UI Controller of your Application should implement `IUIFlow`; your Settings Controller should implement `IUIManagementFlow` instead.
     Icons can be added to `ui.BootstrapUIIcon` or any other enum implementing `IUIIcon` as you need. Be sure to include the necessary sources when using a new Icon supplier.
   - All Subpage-Controllers of the Main UI should implement a Subinterface of `IUIStep`. Please abstract the base interface for your own dynamic binding to work.
     Perhaps examine `datasim.controllers` for a good example.
   - Any Connectors to Out-Of-Scope Services that will run alongside the Toolkit should implement `IExternalService` for a Health Check to be displayed.
3. Bump the Version Number
   For breaking changes or major new features, draft a new major Project Version in the POM.

Painful bugs encountered earlier

- The default HTML input `datetime-local` stepping is 60 seconds. Reduce it to 1 where needed.
- Hibernate does not handle UUIDs properly as ID in MariaDB. Use the `@Column(columnDefinition = "BINARY(16)")` annotation to fix this!

Deeper Dive: Design Patterns in use

- Model-View-Controller
  This Pattern is sourced directly from Spring. It dictates the following:
  - Your UI is filled by Controllers
  - which source or perform actions on a Model of Entities by using Services (a Backend)
  - and is finally rendered to a View (here: by a Templating Engine)
    The MVC Pattern is great to implement a separation of concerns.

- Transfer Objects
  Using Transfer Objects in the Application yields a lot of Flexibility when communicating with external Services.
  By using your mind (and Jackson Annotations), you can generate a JSON-Representation from your Entity without hassle.
  Plus point: By using TOs with the MVC pattern, you have guaranteed decoupling from your persistence layer, which also improves the applications overall security.
  To implement Mapping between Transfer Objects and Entities, all Transfer Objects have Factories (for themselves and their corresponding Entities).
- Component-Driven
  All Parts of this possibly big application can be used stand-alone (apart from the Main UI).
- Adapters
  All external dependencies are wrapped in Connectors which handle the communication with the external service or library.
- Inversion of Control/Dependency Injection
  This Pattern also originates in Spring.
  By annotating our Objects (i.e. Controllers, Services and other Components) correctly, Spring will automatically bind them when we need them.
- Java Bean Lifecycle Management
  This is not a Pattern per se, but an important design aspect for handling the possibility to have multiple external services using one kind of connector.
  The connector is templated depending on an Entity, and the Lifecycle manager (here: lrs.connector.LrsConnectorLifecycleManager) is responsible
  for creating, destroying and delivering the instance of the connector when needed.

Literatur

[1] @sagarsingari und Cliff Casey. *Alignment explanation (Ticket 41)*. https://github.com/yetanalytics/datasim/issues/41. Zugriff: 29.01.2022. 2020.

[2] Sam S. Adkins. *The 2016-2021 Worldwide Self-paced eLearning Market: Global eLearning Market in Steep Decline*. Techn. Ber. Ambient Insight, 2016, Seite 10.

[3] Advanced Distributed Learning Initiative. *Experience API (xAPI) Standard*. https://adlnet.gov/projects/xapi/. Zugriff: 04.12.2021; Absatz "xAPI Technical Details". 2021.

[4] Advanced Distributed Learning Initiative. *Experience API (xAPI) Standard*. https://adlnet.gov/projects/xapi/. Zugriff: 09.12.2021; Absatz "Overview". 2021.

[5] Advanced Distributed Learning Initiative. *Sharable Content Object Reference Model (SCORM®)*. https://adlnet.gov/projects/scorm/. Zugriff: 10.11.2021; Absätze Overview, SCORM History und SCORM Technical Details. 2021.

[6] Advanced Distributed Learning Initiative. *Total Learning Architecture*. https://adlnet.gov/projects/tla/. Zugriff: 04.12.2021; Absatz "What's the TLA?" 2021.

[7] Yet Analytics. *Data and Training Analytics Simulated Input Modeler (DATASIM)*. https://github.com/yetanalytics/datasim. 2021.

[8] Yet Analytics. *Data and Training Analytics Simulated Input Modeler (DATASIM) User Interface*. https://github.com/yetanalytics/datasim-ui. 2021.

[9] Bitkom Research GmbH im Auftrag des Bitkom und des VdTÜV. „Weiterbildung für die digitale Arbeitswelt". In: *Berlin: Bitkom* (2018), Seite 39.

[10] Peter Baumgartner, Hartmut Häfele und Kornelia Maier-Häfele. „Lernplattformen für das Corporate e-Learning". In: *Virtuelle Personalentwicklung: Status und Trends luKT-gestützten Lernens*. Hrsg. von Ulrike Hugl und Stephan Laske. Wiesbaden: Deutscher Universitätsverlag, 2004, S. 95–118. ISBN: 978-3-322-81837-9. DOI: 10.1007/978-3-322-81837-9_5. URL: https://doi.org/10.1007/978-3-322-81837-9_5.

[11] Peter Berking und Shane Gallagher. *Choosing a Learning Management System*. Advanced Distributed Learning Initiative, 2016, Seite 7.

[12] Stiftung Bertelsmann. „Monitor Digitale Bildung. Die Schulen im digitalen Zeitalter". In: *Gütersloh: Verlag Bertelsmann Stiftung* 7 (2017), Seiten 16 und 19.

[13] BPS-Support. *Support-Anfrage zur Verfügbarkeit von Datenschnittstellen in OPAL*. Aug. 2020.

[14] *Canvas LMS API Documentation*.
https://canvas.instructure.com/doc/api/index.html – Zugriff: 20.12.2021. 2021.

[15] IMS Global Learning Consortium. *Learning Tools Interoperability (LTI) Assignment and Grade Services Specification*. Standard 2.0.
https://www.imsglobal.org/spec/lti-ags/v2p0/. 2019.

[16] IMS Global Learning Consortium. *Learning Tools Interoperability Core Specification*. Standard 1.3. https://www.imsglobal.org/spec/lti/v1p3/. 2019.

[17] Philip Dodds (et al.) *The SCORM Content Aggregation Model*. Standard 1.2. Advanced Distributed Learning Initiative, 2001.

[18] Philip Dodds (et al.) *The SCORM Run-Time Environment*. Standard 1.2. Advanced Distributed Learning Initiative, 2001.

[19] Julia Fink u. a. „Learning Management Systeme (LMS) – Entwicklung und Funktionen von webbasierten Lernplattformen". In:
w.e.b.Square – Wissensmanagement und E-Learning unter Bildungsperspektive (Juli 2007).
http://websquare.imb-uni-augsburg.de/2007-07/2.html – Zugriff am 09.12.2021.

[20] Jerry Gordon u. a. *2019 Total Learning Architecture Report*. Techn. Ber. Advanced Distributed Learning Initiative, 2020.

[21] Ralph F. Grove und Eray Ozkan. „The MVC-web Design Pattern". In: *WEBIST*. 2011.

[22] Phil Hill. *State of Higher Ed LMS Market for US and Canada: Year-End 2020 Edition*.
https://philonedtech.com/state-of-higher-ed-lms-market-for-us-and-canada-year-end-2020-edition/. Zugriff: 20.12.2021. Feb. 2021.

[23] Chenn-Jung Huang Huang u. a. „A Learning Assistance Tool for Enhancing ICT Literacy of Elementary School Students". In: *Educational Technology & Society* 13 (Okt. 2010), S. 126–138.

[24] *ILIAS Feature Wiki: REST Service*.
https://docu.ilias.de/goto_docu_wiki_wpage_2121_1357.html. Zugriff: 20.12.2021, Version vom 27.05.2015 16:44. 2021.

[25] Advanced Distributed Learning Initiative. *Data Simulator for TLA (DATASIM)*.
https://adlnet.gov/projects/datasim/. Zugriff: 21.12.2021. 2021.

[26] Advanced Distributed Learning Initiative. *xAPI-Spec*. Technische Spezifikation 1.0.3. Online unter https://github.com/adlnet/xAPI-Spec. 2016.

[27] mmb Institut. *mmb-Trendmonitor 2020/21: Home-Office mischt die E-Learning-Branche auf*. Online unter https://www.mmb-institut.de/wp-content/uploads/mmb-Trendmonitor_2020-2021.pdf. Zugriff: 05.12.2021. 2021.

[28] Ludwig J. Issing und Dieter Baacke. *Medienpädagogik im Informationszeitalter*. 2., durchges. Aufl. Weinheim: Dt. Studien Verl., 1988, Seite 24. ISBN: 9783892710301. URL: https://katalog.slub-dresden.de/id/0-026991357.

[29] Marco Kalz u. a. „Systeme im Einsatz. Lernmanagement, Kompetenzmanagement und PLE". In: *Lehrbuch für Lernen und Lehren mit Technologien*. Hrsg. von Sandra Schön und Martin Ebner. Feb. 2011.

[30] Michael Kerres. *Mediendidaktik: Konzeption und Entwicklung mediengestützter Lernangebote*. Jan. 2013. ISBN: 9783486736038. DOI: 10.1524/9783486736038.

[31] Michael Kerres und Annabell Preußler. „Mediendidaktik". In: *EEO Enzyklopädie Erziehungswissenschaft Online* (2012), S. 1–18. ISSN: 2191-8325. DOI: 10.3262/EE018120258.

[32] Konstantin Köhring und Markus Windisch. *Befragung E-Learning im Unternehmenskontext*. Gespräch via Firmenchat. Dez. 2021.

[33] Manuel Kroiß. „From Backend to Frontend - Case study on adopting Micro Frontends from a Single Page ERP Application monolith". en. 2021. DOI: 10.34726/HSS.2021.85306. URL: https://repositum.tuwien.at/handle/20.500.12708/17595.

[34] *Learning Locker > Visualisations Overview*. https://ht2ltd.zendesk.com/hc/en-us/articles/360029525251-Visualisations-Overview – Zugriff: 12.02.2022. 2021.

[35] Cathy Li und Farah Lalani. „The rise of online learning during the COVID-19 pandemic". In: *World Economic Forum* (Apr. 2020). https://www.weforum.org/agenda/2020/04/coronavirus-education-global-covid19-online-digital-learning/ – Zugriff am 29.12.2021.

[36] Dafinka Miteva und Eliza Stefanova. „LET'S TAKE A LOOK AT BIG DATA: LEARNING ANALYTICS METHODS AND TOOLS FOR VISUALIZATION". In: *International Conference on Education and New Learning Technologies*. März 2017, S. 1613–1622. DOI: 10.21125/edulearn.2017.1348.

[37] *Moodle 3.11 > Externes Tool konfigurieren*. https://docs.moodle.org/311/de/Externes_Tool_konfigurieren. 2021.

[38] *Moodle 3.11 > Web service API functions*. https://docs.moodle.org/dev/index.php?title=Web_service_API_functions&oldid=58944 – Zugriff: 20.12.2021, Version vom 03.06.2021 07:01. 2021.

[39] Klaus North, Kai Reinhardt und Barbara Sieber-Suter. „Kompetent konkurrieren". In: *Kompetenzmanagement in der Praxis: Mitarbeiterkompetenzen systematisch identifizieren, nutzen und entwickeln Mit vielen Fallbeispielen*. https://doi.org/10.1007/978-3-8349-3696-7_1. Wiesbaden: Gabler Verlag, 2013, S. 21. ISBN: 978-3-8349-3696-7. DOI: 10.1007/978-3-8349-3696-7_1.

[40] Alicia Pack (Rustici Software). *An exciting time to watch xAPI and cmi5 adoption numbers*. Zugriff: 13.12.2021. Dez. 2020. URL: https://xapi.com/blog/an-exciting-time-to-watch-xapi-and-cmi5-adoption-numbers/.

[41] Robert Peters und Marc Bovenschulte. *Learning Analytics – Potenzial von KI-Systemen für Lehrende und Lernende*. Techn. Ber. Büro für Technikfolgen-Abschätzung beim Deutschen Bundestag (TAB), 2021. DOI: 10.5445/IR/1000131769.

[42] Hochschulrechenzentrum der Philipps-Universität Marburg. *ILIAS REST Plugin*. https://github.com/hrz-unimr/Ilias.RESTPlugin. 2018.

[43] Tarnum Java SRL. *Default Column Values in JPA*. https://www.baeldung.com/jpa-default-column-values. Zugriff: 09.02.2022. 2019.

[44] Florian Stahr. „Generierung von modellierten xAPI-Statements mit DATASIM". 2022.

Akronyme

ADL Advanced Distributed Learning Initiative. 3, 4, 8, 10, 11, 13, 17, 19, 60

API Application Programming Interface. 7, 8, 13

HTTP Hypertext Transfer Protocol (RFC7230-7235). 13

IRI Internationalized Resource Identifier. 11

KMS Kompetenzmanagementsystem. 1, 2, 13

LAS/TAS Lernassistenzsystem/tutorielles Assistenzsystem. 2, 13, 14, 17

LMS Lernmanagementsystem. 1, 2, 7–10, 12–14

LRS Learning Record Store. 2, 11, 13, 14, 17, 18, 22–24, 29, 30, 35, 42–44, 47, 48, 52–54, 56

LTI Learning Tools Interoperability. 7, 10, 13, 14

REST Representational State Transfer. 12–14, 17, 20

SCORM Sharable Content Object Reference Model. 7–10, 13

TLA Total Learning Architecture. 1, 4, 7, 13

UUID Universally Unique Identifier (RFC4122). 11

xAPI Experience API. 3, 7, 8, 10, 11, 14, 17–19, 23, 32, 52

Glossar

DATASIM Eine Anwendung zur Simulation von xAPI-Statements. 4, 17–20, 22–27, 29, 31–35, 39, 47, 51, 53, 54, 56, 57, 59–61

Experience API Ein Schnittstellenstandard zum Austausch von Lernzustands- und Lernfortschrittsmeldungen. 7, 10, 13, 17

Learning Record Store Server, welcher xAPI-Statements empfangen, speichern und bereitstellen kann [3]. 2–4, 11–14, 18, 22, 23, 25, 29, 30, 33, 34, 38, 39, 47, 49, 56–58, 60

Learning Tools Interoperability Ein Standard zur Kommunikation zwischen Lernprogrammen. 10

Lernassistenzsystem Eine Anwendung, welche Lernende durch einen Lerninhalt führt und Verständnisprobleme frühzeitig erkennen und durch Querverweise lösen kann [41]. 2, 4, 7, 9, 11, 13, 14, 17, 60, 61

Lernmanagementsystem Server-basiertes Softwaresystem zur Verwaltung und Bereitstellung von Lerninhalten (durch einen Browser) [11]. 1–4, 7–14

Lernplattform Ein Lernmanagementsystem ohne Einschränkung auf die serverseitige Datenhaltung. 1–3, 7–11, 13, 14

Representational State Transfer Ein Standard zur Übertragung von Entitäten und Zuständen über HTTP. 12

Sharable Content Object Reference Model Ein Standard zum Austausch von E-Learning-Inhalten. 8

Total Learning Architecture Forschungsprojekt der ADL. Umfasst Spezifikationen für die Integration unterschiedlicher Lerntechnologien in ein gemeinsames Ökosystem [6]. 1, 4, 7, 13, 15, 60

xAPI-Statement Datenaustauschobjekt der Experience API. Enthält Daten der Art *I did that* im JSON-Format.. 4, 11, 13, 14, 17–19, 22, 23, 27, 31, 34, 43, 44, 47, 56, 59, 60

Abbildungsverzeichnis

Liste der Festlegungen

Liste der Anforderungen

Liste der Entwicklungsaufgaben

Quelltextverzeichnis