Ein auf BPMN[1] und REST[2] basierendes Framework zur Umsetzung des OAIS-Referenzmodells: Die fachlichen und technischen Anforderungen der digitalen Archivierung mit einem auf BPMN und REST basierenden Framework meistern

Folie 1:

Liebe Kolleginnen und Kollegen,

bevor ich mit meinem Vortrag beginne, möchte ich darauf aufmerksam machen, dass ich Ihnen heute das theoretische Fundament eines seit September 2020 im Landesarchiv Sachsen-Anhalt verfügbaren und produktiv einsetzbaren Frameworks zeigen werde.

Innerhalb eines Vierteljahres habe ich mit dem von mir entwickelten Framework, das als Appliance verfügbar ist, exemplarisch eine Workflow-basierte Integration der Bayrischen Access-Komponente, einen xDomea-Ingest und das Preservation realisiert. Am Vortragsende blende ich meine Kontaktdaten ein. Schreiben Sie mir, wenn Sie in einem Online-Seminar die praktische Benutzung ausführlich vorgestellt haben möchten.

Die eben erwähnten exemplarischen Umsetzungen von OAIS-Referenzmodellfunktionen habe ich natürlich ausführlich im DIMAG-DAN-Archivverbund vorgestellt. Es war offensichtlich, dass fachliche Anforderungen innerhalb kürzester Zeit umsetzbar sind.

Folie 2:

Mein Name ist Frank Obermeit und ich arbeite seit 2011 im Landesarchiv Sachsen-Anhalt als Informatiker. Seit 2017 widme ich mich intensiv dem Thema Digitale Archivierung, veranlasst durch die Mitarbeit im DIMAG-Entwicklungsverbund. Davor habe ich 35 Jahre in Rechenzentren gearbeitet.

Folie 3:

Im August 2017 fragte ich mich:

Ist es möglich das OAIS-Referenzmodell software-technisch umzusetzen?

Die ersten Erkenntnisse waren, dass die Prozesse mit dem OMG-Standard BPMN beschrieben und die Schnittstellen auf Daten und Funktionen über REST gekapselt werden können. REST ist ein de facto Standard der im Bereich der Webservices dem XML-basierten SOAP[3] den Rang abgenommen hat.

[1] Business Process Model and Notation - https://de.wikipedia.org/wiki/Business_Process_Model_and_Notation
[2] Representational State Transfer - https://de.wikipedia.org/wiki/Representational_State_Transfer
[3] Simple Object Access Protokol - https://de.wikipedia.org/wiki/SOAP

Damit sind wir beim Thema des Vortrages.

Folie 4:

Den grundsätzlichen Ansatz habe ich ausführlich auf der AUdS-Tagung 2019 in Prag vorgestellt und damals die Kernaussage des Vortrages in folgendem Satz formuliert.

Die Kernaussage lautete:

Die Archival Storages werden mit REST gekapselt, die Prozessmodellierung und Prozessverarbeitung erfolgt mit BPMN, alles wird über eine webbasierte Benutzeroberfläche dem Nutzer zur Verfügung gestellt und die vorhandene Rechteverwaltung wird integriert.

Folie 5:

Ich werde Ihnen heute zeigen, wie vom OAIS-Referenzmodell ausgehend, über ein Software-Architektur-Modell ein Software-Framework entstand, welches die Umsetzung der Anforderungen des OAIS-Referenzmodells ermöglicht.

Folie 6:

Die Ausgangsfrage lautet:

Wie kann das OAIS-Referenzmodell software-technische strukturiert werden?

Folie 7:

Es gibt die Entitäten SIP, AIP und DIP.

Die AIPs werden im Archival Storage verwaltet und dauerhaft aufbewahrt.

SIPs und DIPs benötigen ebenfalls Storages. Vergleichen wir die Anforderungen zwischen AIP einerseits und SIP und DIP andererseits, dann besteht der wesentliche Unterschied in der Aufbewahrungsfrist. AIPs werden langfristig aufbewahrt und SIPs und DIPs mehr oder weniger kurzfristig.

Die Entitäten (SIP, AIP, DIP) werden durch das funktionale Modell bewegt und dabei be- bzw. verarbeitet.

- *SIP - submission information package*
- *AIP - archival information package*
- *DIP - dissemination information package*

Folie 8:

SIP, AIP und DIP können mit UUIDs adressiert und URIs lokalisiert werden.

Folie 9:

An dieser Stelle soll das Akronym REST eingeführt werden.

REST (seltener ReST) steht für Representational State Transfer und bezeichnet ein Programmierparadigma für verteilte Systeme, insbesondere für Webservices.

REST wird, neben SOAP (Simple Object Access Protocol), vorrangig für die Maschine-Maschine-Kommunikation eingesetzt.

Der Begriff RESTful oder RESTful HTTP taucht häufig auf und verdeutlicht lediglich, dass REST unter Ausnutzung der Möglichkeiten des Application Protocol HTTP umgesetzt wird.

Mit REST können die eigentlichen Daten lokalisiert werden, aber auch Dienste respektive Services bereitgestellt werden.

Folie 10:

REST-Webservices können für die Verwaltung von Ressourcen genutzt werden.

Hier ein Beispiel für einen REST-Webservice, der neben GET auch die Methoden PUT, POST und DELETE nutzt.

***POST**: Fügt eine neue (Sub-)Ressource unterhalb der angegebenen Ressource ein. Da die neue Ressource noch keinen URI besitzt, adressiert der URI die übergeordnete Ressource. Als Ergebnis wird der neue Ressourcenlink dem Client zurückgegeben.*

***POST** kann im weiteren Sinne auch dazu verwendet werden, Operationen abzubilden, die von keiner anderen Methode abgedeckt werden.*

***GET**: Wird für Reads benutzt.*

***PUT**: Die angegebene Ressource wird angelegt. Wenn die Ressource bereits existiert, wird diese geändert.*

*Im Zusammenhang mit REST wird der Begriff **Idempotent** benutzt und dies bedeutet, dass laut HTTP-Spezifikation für die Methoden GET, HEAD, PUT und DELETE gilt:*

*Das mehrfache Absenden der gleichen Anforderung wirkt sich **nicht** anders als ein einzelner Aufruf aus (siehe wiki[4]).*

Folie 11:

Über REST-Webservices können auch Daten-ver-und-be-arbeitende Operationen bereitgestellt werden.

Hier sehen Sie exemplarisch eine auf DROID basierende Formaterkennung, die als REST-Webservice bereitgestellt wird. Dabei kann die Ressource als URI oder als URL lokalisiert werden.

[4] https://de.wikipedia.org/wiki/Representational_State_Transfer

Folie 12:

Gehen wir wieder zurück zum OAIS-Referenzmodell.

Hinter den Funktionseinheiten:

- Archival Storage,
- Ingest,
- Preservation Planing,
- Data Management,
- Access und
- Administration

Folie 13:

verbergen sich Dienste und Funktionen, aber auch Workflows.

SIP, AIP und DIP können mit REST-Webservices lokalisiert werden und Dienste und Funktionen darüber bereitgestellt werden.

Diese Funktionseinheiten stellen Dienste und Funktionen bereit, aber letztendlich verbergen sich Workflows dahinter.

Hinweis: Dienste beziehen sich auf Referenzmodelle für offene Systemumgebungen z.B. Betriebssystem-Dienste, Netzwerkdienste, Sicherheitsdienste.[5](Seite 35 Abschnitt 4.1.1)

Folie 14:

An dieser Stelle soll das Akronym BPMN eingeführt werden.

BPMN ist die Abkürzung für „Business Process Model and Notation". Mit BPMN können Workflows modelliert werden. Wenn BPMN genannt wird, dann muss auch DMN „Decision Model and Notation" erwähnt werden, denn dahinter verbergen sich Entscheidungsregeln, ein wichtiges BPMN-Modellierungselement.

Das Case Management Model and Notation (CMMN[6]) spielt momentan keine Rolle, da es für die Modellierung nicht so klar spezifizierbarer Prozessabläufe genutzt wird.

[5] http://www.mnm-team.org/pub/Diplomarbeiten/stri98/HTML-Version/node36.html
[6] https://en.wikipedia.org/wiki/CMMN

Folie 15:

Die sich hinter den Funktionseinheiten verbergenden Workflows können mit BPMN modelliert werden – sprich, die Dienste und Funktionen werden miteinander zu Workflows verknüpft. Dies möchte ich Ihnen an einem fiktiven Ingest einer XDOMEA-503-Nachricht verdeutlichen.

XDOMEA-Beispiel mit BPMN. Der fiktive Ingest-Workflow beginnt

- mit der XML-Validierung der 503-Nachricht,
- bei nicht erfolgreicher Validierung, wird eine E-Mail an den Archivar versandt,
- bei erfolgreicher Validierung werden die Daten-Dateien der 503-Nachricht ermittelt,
- bei jeder Datei wird deren Existenz und Hashcode geprüft und deren Format ermittelt,
- im Fehlerfall erhält der Archivar eine E-Mail,
- für jede Datei werden deren Prüfergebnisse zusammengefasst und
- nachdem alle Dateien verarbeitet wurden, werden die Prüfergebnisse der gesamten 503-Nachricht zusammengefasst.

Die Daten-Datei-Prüfungen können parallel durchgeführt werden.

Beim Validieren und bei der Formatbestimmung werden REST-Webservices eingesetzt.

Das fachliche BPMN-Modell sieht dann so aus.

Folie 16: *Übergang vom OAIS-Referenzmodell zum Architekturmodell*

Die software-technische Strukturierung haben wir durch die Kapselung der Daten und Funktionen mittels REST-Webservices und deren Verknüpfung zu Workflows mit BPMN abgeschlossen.

Wir können uns jetzt der Software-Architektur-Modellierung zuwenden, um letztendlich darauf aufbauend ein flexibles und wartbares Framework zu implementieren.

Folie 17:

Eine Software-Architektur beschreibt die grundlegenden Komponenten und deren Zusammenspiel innerhalb eines Softwaresystems und bezieht sich dabei typischerweise auf die größeren Strukturen. Sie befasst sich mit der Art und Weise, wie mehrere Software-Prozesse zur Erfüllung ihrer Aufgaben zusammenarbeiten.

In der Software-Architektur werden nicht-funktionale Entscheidungen gefällt und von den funktionalen Anforderungen getrennt. Im Software-Design erfolgt die Beschreibung der funktionalen Anforderungen.

Warum bedienen wir uns einer Software-Architektur?

Die Software-Architektur

- dient als Blaupause für ein System,
- bietet eine Abstraktion, um die Systemkomplexität zu verwalten,
- etabliert einen Kommunikations- und Koordinationsmechanismus zwischen den Komponenten,
- deckt die Struktur des Systems auf,
- blendet aber die Details der Implementierung aus.

Im Gegensatz dazu, bietet das Software-Design einen Designplan, der die Elemente eines Systems beschreibt sowie die Art und Weise, wie sie zusammenpassen und zusammenarbeiten, um die Anforderungen des Systems zu erfüllen, und fungiert als Blaupause für den Entwicklungs-prozess.

Folie 18:

Ein Software-Architektur-Modell dient nicht nur der Strukturierung als Vorstufe zur technischen Umsetzung, sondern soll auch ein gemeinsames Verständnis der Gesamtarchitektur bei allen fachlich, technisch und organisatorisch Verantwortlichen vermitteln.

Das Modell muss intuitiv erfassbar sein. Archivare und Informatiker sollen es erklären können und keine Frage soll auf diesem Abstraktionsniveau unbeantwortet bleiben.

Folie 19:

In dem Buch „Langlebige Software-Architekturen" von Frau Dr. Carola Lilienthal (2017) findet sich eine sehr intuitiv verständliche Vorlage für die Darstellung der Software-Architektur. Die Vorlage basiert u.a. auf den Ausarbeitungen von Eric Evans zum Domain-Driven Design und Gernot Starke zu effektiven Software-Architekturen.

Der Modellierungsvorschlag sprach mich und unseren für die digitale Archivierung zuständigen Archivar sofort an.

Schreiten wir auf dem Weg vom OAIS-Referenzmodell zum Framework voran.

Folie 20:

Die sich hinter dem Modell verbergenden Grundgedanken sind:

- Die Software-Architektur-(Modell)-Vorlage ist zweidimensional aufgebaut und enthält eine fachliche und eine technische Dimension.
- Höhere Schichten dürfen darunterliegende Schichten verwenden.
- Das verwendete Modell enthält die Präsentations-, Applikations- und Fachdomänen-Schichten.
- Jeder Schicht werden Aufgaben zugeordnet und damit eine semantische Anreicherung nach technischen Gesichtspunkten herbeigeführt.
- Die fachliche Schichtung ist ein probates Mittel zur Hierarchisierung und Aufteilung der Fachlichkeit.
- Jedes fachliche Modul enthält Anteile der technischen Schichten.
- Früher wurde eine Infrastrukturschicht als zusätzliche unterste technische Schicht modelliert, doch daraus ergab sich das Dilemma, dass nicht direkt darüber liegende Schichten diese auch nicht direkt verwenden durften.
- Daher wird die bisher übliche Infrastrukturschicht als eine schichtenübergreifende Komponente modelliert.

Resultat: „Höhere Schichten, sowohl in der technischen als auch in der fachlichen Dimension der Schichtung, dürfen die weiter untenliegenden Schichten verwenden. Ein klares und einheitliches Muster!“ (siehe Seite 99 in Carola Lilienthal, 2017).

- Jede Komponente besitzt eine Präsentationsschicht, dennoch soll sich das Gesamtsystem dem Benutzer als eine homogene Einheit präsentieren. Diese Aufgabe übernimmt eine integrierende Benutzeroberfläche.

Folie 21:

Übertragen wir die bisherigen Strukturierungsergebnisse auf die Architektur-Modell-Vorlage.

- Die fachlichen Schichten bezeichnen wir als Module und bisher haben wir die drei fachlichen Module SIP, AIP und DIP identifiziert. Diese kapseln Daten und Dienste bzw. Funktionen und stellen „nur" die Be- und Verarbeitungsfunktionen bereit.
- Alle Funktionseinheiten enthalten Workflows und organisatorische Anweisungen bzw. Regeln, die der Workflow-Komponente zugeordnet werden.
- Es gibt Funktionen, die in allen fachlichen Komponenten benötigt werden und in die Infrastruktur-Komponente ausgelagert werden, z.B. XML-XSD-Validierung, Virenprüfung und Formatanalyse.
- Dem Thema Datensicherheit muss das Modell auch gerecht werden. *Eine Erkenntnis aus Softwareprojekten, bei denen komplexe Anforderungen umzusetzen sind, ist: Es ist wie bei einer Bettdecke, egal an welcher Stelle Sie ziehen, Sie haben immer die ganze Decke an der Hand. Auf Software-Projekte bezogen heißt dies, Sie müssen sich immer mit allen Themen beschäftigen. Und eines der schwierigsten Themen ist und bleibt die Datensicherheit. Kurzum,* es wird ein Identity and Access Management Modul benötigt, mit dem die Schutzbedarfskategorien normal, hoch und sehr hoch innerhalb eines digitalen Archivs umgesetzt werden können.
- Bei all diesen technischen Strukturierungen entscheidet sich die Akzeptanz dennoch am Komfort der Benutzeroberfläche, über die das Gesamtsystem für den Archivar greifbar wird. Der Nutzer möchte nicht auf Entdeckungsreise gehen, sondern das System benutzen und seine Arbeit erledigen. Es wird, wie bereits erwähnt, eine integrierende Oberfläche benötigt.

Damit wäre das Modell vollständig und alle Anforderungen des OAIS - Referenzmodells umsetzbar.

Modell und Standards

Die Komponenten des Modells basieren auf Standards. Diese möchte ich nur grob andeuten.

- Alle Module arbeiten UUIDs,
- die Ressourcen der fachlichen Module werden mit URIs lokalisiert,
- alle Komponenten bieten REST-Webservices an bzw. die integrierende Benutzeroberfläche verarbeitet diese,
- OpenAPI wird für die Dokumentation der REST-Webservices genutzt,
- die Workflow-Modellierung basiert auf BPMN und DMN und
- beim Identity and Access-Management werden die Standards OAuth2 für die Autorisierung und OpenID Connect für die Authentifizierung angewendet.

UUID *- UUID sind als Teil des Standards ISO/IEC 11578:1996 "Information technology – Open Systems Interconnection – Remote Procedure Call (RPC)" und als separater Standard ISO/IEC 9834-8:2005 dokumentiert. Die IETF hat das auf UUID basierende RFC 4122 veröffentlicht.*[7]

URI *- Der aktuelle Stand 2016 ist als RFC 3986 publiziert. Der aktuelle Stand 2016 ist als RFC 3986 publiziert.*[8]

REST *– Ist ein etabliertes Programmierparadigma.*[9]

BPMN *- Die aktuelle Version des BPMN-Standards, BPMN 2.0, wurde im Januar 2011 von der OMG verabschiedet.*[10]

DMN *- Decision Model and Notation (kurz DMN) ist ein offizieller Notationsstandard für Entscheidungsregeln im Geschäftsprozessmanagement, der von der Object Management Group (OMG) definiert wurde.*[11]

OAuth2 *- OAuth 2.0 wurde in RFC 6749 und RFC 6750 publiziert.*[12]

OpenID Connect *- OpenID Connect (OIDC) ist eine Authentifizierungsschicht, die auf dem Autorisierungsprotokoll OAuth 2.0 basiert. Der Standard wird durch die OpenID Foundation überwacht.*[13]

OpenAPI *- Die OpenAPI Specification (vormals Swagger Specification) ist ein Standard zur Beschreibung von REST-konformen Programmierschnittstellen (API).*[14]

JSON *- Die JavaScript Object Notation ist ein kompaktes Datenformat in einer einfach lesbaren Textform und dient dem Zweck des Datenaustausches zwischen Anwendungen. JSON wurde ursprünglich von Douglas Crockford spezifiziert. Aktuell wird es durch zwei konkurrierende Standards spezifiziert – RFC 8259[2] sowie ECMA-404.*[15]

Folie 22:

[7] https://de.wikipedia.org/wiki/Universally_Unique_Identifier
[8] https://de.wikipedia.org/wiki/Uniform_Resource_Identifier
[9] https://de.wikipedia.org/wiki/Representational_State_Transfer
[10] https://de.wikipedia.org/wiki/Business_Process_Model_and_Notation
[11] https://de.wikipedia.org/wiki/Decision_Model_and_Notation
[12] https://de.wikipedia.org/wiki/OAuth#OAuth_2.0_und_OpenID_Connect
[13] https://de.wikipedia.org/wiki/OpenID_Connect
[14] https://de.wikipedia.org/wiki/OpenAPI
[15] https://de.wikipedia.org/wiki/JavaScript_Object_Notation

Bevor ich Ihnen die software-technische Umsetzung des Frameworks vorstelle, möchte ich auf nicht offensichtliche Aspekte des Modells eingehen.

Folie 22a:

Beginnen wir mit den fachlichen Modulen. Diese bestehen aus den drei technischen Schichten und dabei gilt, dass die eigentlichen Daten über UUIDs adressiert und über URIs lokalisiert werden.

Die Funktionen werden über die Applikationsschicht bereitgestellt.

Diese Grundsätze gelten auch für die Infrastruktur-Komponente.

Die Module besitzen eine Präsentationsschicht und können somit autark getestet und benutzt werden.

Diese Präsentationsschicht kann unterschiedlich gut ausgebaut sein, also im einfachsten Fall eine komfortable Oberfläche für die Benutzung der Applikationsschicht – sprich der REST-Webservices - bereitstellen oder im besten Fall eine komplette Web-Anwendung z.B. das DIMAG-Kernmodul.

In beiden Fällen erfolgt eine Integration an der Oberfläche.

Folie 23:

Die Workflow-Komponente muss diese Grundsätze ebenfalls erfüllen und die Standards BPMN und DMN abdecken.

An dieser Stelle möchte ich darauf hinweisen, dass BPMN die Modellierung von Transaktionen in verteilten Systemen ermöglicht. Das Rollback einer Transaktion wird als „Kompensation" modelliert, muss aber entsprechend implementiert werden.

BPMN verwendet das Konzept transaktionaler Subprozesse in Kombination mit Kompensierungsereignissen und -aktivitäten zur Transaktions-steuerung in verteilten Systemen[16].

Dieses möchte ich Ihnen an ausgewählten Fallbeispielen verdeutlichen.

Folie 23a:

Im ersten Fallbeispiel sehen Sie eine Service Task und die dazugehörende Kompensationstask. In der Service Task „check Data" werden die Daten geprüft und das Bewertungsergebnis wird im darauffolgenden Exklusive-Oder(XOR)-Gateway ausgewertet. Bei erfolgreicher Verarbeitung, wird der Workflow ohne Fehler beendet. Andernfalls wird über ein Ereignis die Kompensation der Task „Insert Data" ausgelöst.

[16] https://blog.holisticon.de/2011/05/transaktionen-bpmsoa/

Folie 23b:

Das zweite Fallbeispiel zeigt eine Transaktion (erkennbar an der doppelten Umrandung), die aus zwei Tasks mit Kompensation besteht. Die parallelen Striche am unten Rand der Transaktion deuten an, dass die Transaktion mehrfach ausgeführt wird. Die mehrfache Ausführung kann entweder parallel oder sequentiell erfolgen. Hier parallel. Mit dem Blitz an der Transaktion wird ein unterbrechendes Fehlerereignis modelliert. In diesem Fall wird nicht nur eine E-Mail versandt, sondern die Kompensationen für alle bisher erfolgreich durchgeführten Tasks der Transaktion ausgelöst.

Folie 23c:

Das dritte Fallbeispiel zeigt ein „Ereignisbasiertes Gateway". Der Prozessverlauf hängt davon ab, welches dem Gateway folgende Ereignis zuerst auftritt.

In diesem Beispiel: Die „verarbeite Daten"-Task wird ausgeführt, wenn eine E-Mail ankommt, oder der Archivar wird per E-Mail informiert, wenn die E-Mail nicht innerhalb eines bestimmten Zeitraumes eintrifft.

Folie 24: Animation

Folie 25:

Bei der IAM-Komponente möchte ich auf einen nicht offensichtlichen Aspekt aufmerksam machen.

Stellen Sie sich vor: Beim Ingest wird ein SIP verarbeitet und bearbeitet und als AIP in den Archival-Storage übernommen. Diese Übernahme wird als Workflow modelliert und Sie können sich vorstellen, dass viele Funktionen des SIP,- AIP,- und Infrastruktur-Moduls genutzt werden und über den Workflow miteinander verbunden werden. Viele Prozessschritte laufen im Backend-Bereich ab und es muss natürlich abgesichert werden, dass der Nutzer dafür auch autorisiert ist.

Kurzgefasst, die Autorisierung des Nutzers muss an die beteiligten Backend-Funktionen weitergereicht werden. Dafür gibt es seit 2012 die technische Spezifikation OAuth2 und seit 2014 die darauf aufsetzende Schicht OpenID Connect 2.0. OAuth2 ist für die Autorisierung und OpenID Connect für die Authentifizierung zuständig.

Schauen wir uns zunächst die IAM-Komponente aus Sicht des Archivars an.

Nehmen wir wieder einen fiktiven Workflow:

Bei Web-Anwendungen sprechen wir vom Frontend- und vom Backend-Bereich. Der Nutzer mit dem Browser wird dem Frontend- und die Web-Services dem Backend-Bereich zugeordnet.

- Der Archivar authentifiziert sich einmalig und ist laut Rechte- und Rollenkonzept autorisiert, bestimmte Dienste auszuführen.
- Der Archivar bearbeitet ein SIP über den Browser,
- anschließend wird das SIP im Backend-Bereich verarbeitet,
- ein REST-Webservice wird genutzt,
- das AIP wird gebildet und
- abschließend wird der Archivar per E-Mail informiert.

Allgemein formuliert:

- Der Archivar meldet sich einmalig an (Single Sign On),
- die Authentifizierung erfolgt gegen LDAP,
- es wird ein Access-Token ausgestellt.
- Mit einem Access-Token werden die Berechtigungsinformationen an weitere im Workflow genutzte Backend-Services weitergereicht, wobei die ursprünglichen Authentifizierungsangaben, Nutzername und Password, nicht weitergegeben werden.

Bei verteilten Systemen interagieren Webservices (fachliche Komponenten), gesteuert über die Workflow-Komponente, miteinander. Die Workflows werden vom Nutzer direkt oder indirekt initiiert und es muss natürlich abgesichert werden, dass innerhalb des Workflows nur Services benutzt werden, für die der Nutzer auch autorisiert ist. Dieser Anforderung muss eine IAM-Komponente gerecht werden.

Die sich dahinter verbergenden Standards sind:

- OAuth2 und
- OpenID Connect

OAuth 2.0 und publizierte es in RFC 6749 und RFC 6750

RFC 6749: Das OAuth 2.0-Berechtigungskonzept

Das OAuth 2.0-Berechtigungs-Framework ermöglicht einem Dritten Anwendung, um begrenzten Zugriff auf einen HTTP-Dienst zu erhalten, entweder auf im Namen eines Ressourcenbesitzers durch Orchestrierung einer Genehmigungsinteraktion zwischen dem Ressourcenbesitzer und dem HTTP-Dienst, oder indem Sie den Antrag eines Dritten auf Zugang in seinem eigenen Namen. Dieser Spezifikation ersetzt und veraltet das beschriebene OAuth 1.0-Protokoll in RFC 5849.

RFC 6750: Das OAuth 2.0-Berechtigungskonzept: Verwendung von Inhaber-Token

Diese Spezifikation beschreibt die Verwendung von Inhaber-Tokens in HTTP Anträge auf Zugang zu geschützten OAuth 2.0-Ressourcen. Jede Partei die Besitzer eines Inhaber-Tokens (ein "Überbringer") ist kann diesen benutzen, um Zugang zu den damit verbundenen Ressourcen (ohne Nachweis des Besitzes eines kryptographischen

Schlüssels) zu erhalten. Um Missbrauch zu verhindern, müssen die Inhaber-Token vor der Offenlegung bei der Lagerung und beim Transport geschützt.

Schauen wir uns die IAM-Komponente aus Sicht des Benutzers an. Mit denselben Standards können die Authentifizierung des Nutzers und die Autorisierung für das Archiv durchgeführt werden.

Der Nutzer

- löst eine DIP-Bestellung über den Browser aus,
- im Backend-Bereich wird das DIP aus dem AIP erzeugt und
- anschließend wird der Nutzer darüber per E-Mail informiert.

Allgemein formuliert:

- Der Nutzer meldet sich einmalig an (Single Sign On),
- die Authentifizierung erfolgt aber nicht gegen LDAP,
- sondern gegen einen Identity Provider z.B. Twitter, Paypal.
- Es wird ein Access-Token ausgestellt.
- Mit einem Access-Token werden die Berechtigungsinformationen an weitere im Workflow genutzte Backend-Services weitergereicht, wobei die ursprünglichen Authentifizierungsangaben, Nutzername und Password, nicht weitergegeben werden.

Der große Vorteil besteht darin, dass Sie das Rechte- und Rollenkonzept bei Archiv-Mitarbeitern mit der hauseigenen Nutzerverwaltung umsetzen können, Sie aber keine separate Verwaltung für die Archiv-Benutzer aufbauen müssen.

Folie 26: Animation

Folie 27:

Die Benutzeroberfläche (engl. User Interface) ist die Komponente, die das Gesamtsystem für den Nutzer greifbar erscheinen lässt, und unterliegt eher technologischen und ergonomischen Änderungen, im Gegensatz zu allen anderen bisher dargestellten Komponenten.

Die Benutzeroberfläche soll alle vom Gesamtsystem bereitgestellten Funktionalitäten als eine homogene Einheit erscheinen lassen.

Die Oberfläche integriert die von den Modulen angebotenen Benutzeroberflächen oder visualisiert die über REST-Webservices angebotenen Funktionen auf eigene Art. *(Verweis auf REST-Swagger vs. kompletter Oberfläche)*

Wenn nach dem vorgestellten Architektur-Ansatz entwickelt wird, dann können fachliche Anforderungen schnell umgesetzt werden, und die Integration neuer Funktionalitäten ist kurzfristig möglich.

Folie 28: ausgeblendet

Jetzt folgt die spannende Frage: Wer baut das oder gibt es dafür fertige Produkte? Ja, es gibt fertige Produkte.

Doch vorher sollen einige der für das Framework geltenden Grundsätze, Prinzipien und Anforderungen genannt werden:

- Die Eigenentwicklung soll auf ein Mindestmaß beschränkt werden.
- Die Prozessgestaltung soll auch bei der technischen Umsetzung flexibel bleiben.
- Das Rechte- und Rollenkonzept soll mit der hauseigenen Nutzer-verwaltung umsetzbar sein.
- Alle Schutzbedarfskategorien sollen unterstützt werden.
- Das Gesamtsystem soll horizontal und vertikal skalierbar sein, damit es von kleinen und großen Archiven genutzt werden kann.

Folie 29:

Auf Standards haben wir bereits gesetzt und die Eigenentwicklung wollen wir auf ein Mindestmaß beschränken!

Mit welchen Software-Produkten kann der Architekturansatz umgesetzt werden? Welche Open-Source- oder Community-Produkte gibt es?

Folie 29a:

Das Workflow-Modul kann mit dem Open-Source-Produkt „Camunda" von der gleichnamigen deutschen Firma umgesetzt werden.

Folie 29b:

Bei der IAM-Komponente heißt mein Favorit Keycloak. Ein Open Source Produkt, dass die Standards OAuth2 und OpenID Connect umsetzt und die Integration Ihrer eigenen Nutzerverwaltung, basierend auf LDAP oder dem Active Directory, ermöglicht. Ein weiterer Mehrwert ergibt sich bei Keycloak daraus, dass Identity Provider eingebunden werden können. Sie müssen sich daher nicht um die Verwaltung der „DIP-Consumer" kümmern, denn diese würden sich z.B. mit ihrem Social Media Account authentifizieren können.

Im SIP- und AIP-Bereich werden Sie bei der Authentifizierung und Autorisierung die Ihnen vertraute Benutzerverwaltung integrieren und im DIP-Bereich auf Social Login Accounts setzen.

Sie können mit Keycloak aber auch eine autarke Nutzerverwaltung aufbauen.

Keycloak wird von Red Hat betreut und die Projektbezeichnung lautet „Red Hat Single Sign On".

Das IAM-Modul kann mit dem von Red Hed betreuten Produkt „Keycloak" umgesetzt werden. Die Projektbezeichnung für Keycloak lautet bei Red Hat „Red Hat Single Sign On".

Folie 29c:

Bei der integrierenden Benutzeroberfläche heißt mein Favorit „Application Express" von Oracle, kurz APEX genannt. Sie können mit geringem Aufwand Web-Anwendungen und eine integrierende Benutzeroberfläche entwickeln.

Die Benutzeroberfläche ist, wie bereits mehr erwähnt, das Anhängeschild der Software und die Komponente, über welche das Gesamtsystem für den Archivar greifbar wird.

Unabhängig von APEX sollte Sie so umgesetzt werden, dass sie durch die konsequenten REST-Schnittstellenorientierung jederzeit ausgetauscht und Produkte nach eigener Präferenz ausgewählt werden können.

Folie 30:

Bisher wurde nicht eine Zeile programmiert, sondern nur konfiguriert.

Schauen wir uns beispielsweise das Validieren einer XDOMEA-503-Nachricht im SIP-Modul an.

Diese Funktionalität könnte mit BaseX umgesetzt werden. BaseX ist ein natives und kompaktes XML-Datenbankmanagementsystem, das als deutsches Community-Projekt auf GitHub entwickelt wird.*[1]*

Es wird vorwiegend zur Speicherung, Anfrage und Visualisierung großer XML-Dokumente und -Kollektionen eingesetzt.[2] BaseX ist plattformunabhängig und wird unter einer freizügigen Open-Source-Lizenz (BSD) angeboten.

Mit BaseX können alle drei Schichten realisiert werden.

Sie sehen, der gesamte Programmieraufwand besteht aus 25 Zeilen gut lesbaren XQuery-Anweisungen.

Die Konfiguration der BPMN-Service-Task (http-connector) ist mit vier Parametern sehr überschaubar.

Folie 31: *ausgeblendet* - *Text ändern.* Hier habe ich die Folie aus einem anderen Vortrag eingefügt. Dabei sollte gezeigt werden, dass die SIP-Komponente mit der XML-Datenbank BaseX als Archival Storage für SIPs. BaseX stellt daneben REST-Web-Services bereit und innerhalb derer können XML-Abfragen konfiguriert werden.

Die Kapselung der BaseX-Web-Services erfolgt über einen eigenen http-Server.

Folie 32: *ausgeblendet* - *Text ändern.* AIP-Komponente: Das DIMAG-Kernmodul könnte um weitere Web-Services wie JHove, DROID, VeraPDF und ClamAV etc ergänzt werden und alle Services inkl. des DIMAG-Kernmoduls könnten über einen Apache-http-Web-Server gekapselt werden.

Folie 33: Animation

Folie 34:

Salopp kann formuliert werden, dass die Architektur aus über Workflows orchestrierten Services besteht.

Folie 35:

Was ändert sich dadurch?

Die Arbeitsweise ändert sich, weil modellorientiert analysiert, konzipiert und umgesetzt bzw. entwickelt wird.

Folie 36:

Es beginnt mit

- einem fachlichen BPMN-Modell,
- welches um technische Aspekte ergänzt und
- in einem iterativen Prozess fachlich und technisch verfeinert wird.

Werden steuernde Elemente benötigt, dann kann auf die BNO der BPMN-Engine oder APEX zurückgegriffen werden.

Anschließend wird der Workflow

- getestet und
- veröffentlicht und
- kann mit anderen Archiven ausgetauscht werden.

In dem konkreten XDOMEA-Anwendungsfall wurde wie folgt vorgegangen:

Der gesamte 503-Verarbeitungsprozess wurde durchmodelliert.

Anschließend wurde es um einige BNO-Elemente erweitert und einer BPM-Engine zur Abarbeitung übergeben.

Insgesamt würden vier Wochen für eine auslieferbare Umsetzung genügen.

Folie 37:

Alle Schutzbedarfskategorien können umgesetzt werden, weil sich die Daten im Backend-Bereich, also auf den Servern, befinden und nicht unnötig bewegt werden.

Folie 38:

Was wird weitergegeben?

Zwischen Entwicklungspartnern werden nur ausgetauscht:

- das BPMN-Modell (inkl. der evtl. vorhandenen Forms),
- die REST-Webservice-Definitionen und
- das APEX-SQL-Skript.

Das Framework wird **nicht** verändert.

Das BPMN-Modell inkl. Forms wird per curl deployed.

Die REST-Webservice-Konfigurationen werden beispielsweise im BaseX bereitgestellt.

Wenn eine zusätzliche BNO mit APEX entwickelt wurde, dann wird diese per SQLPlus importiert.

Folie 39:

Welche grundsätzlichen Anforderungen werden erfüllt?

- Es wird eine flexible Prozessgestaltung von der Analyse bis zur Implementierung erreicht.
- Das Rechte- und Rollenkonzept kann mit der eigenen Nutzer-verwaltung umgesetzt werden.
- Alle Schutzbedarfskategorien können unterstützt werden.
- Die Anwendung kann benutzerfreundlich gestaltet werden.

Folie 40:

Welche nicht-funktionalen Vorteile ergeben sich?

- Sie können sich auf die fachliche BPMN-Modellierung konzentrieren.
- Alles wird durch die Verwendung von BPMN und REST ausreichend dokumentiert.
- Es werden Standards verwendet und damit professionelle Produkte einsetzbar, was schlussendlich Eigenentwicklung vermeidet.
- Durch den Architektur-Ansatz kann ein einheitliches Framework eingesetzt, aber individuell genutzt werden.
- Die Finanzierungsprobleme verringern sich – sprich, Aufträge können viel konkreter ausformuliert und vergeben werden.
- Durch ein integriertes und erweiterbares DokuWiki wird das Gesamtsystem selbstdokumentierend.
- Das bisher vorhandene Framework (Stand September 2020) kann für den Testbetrieb innerhalb von vier Stunden out-of-the-box installiert werden.

Folie 41:

Skalierung mit Docker und Kubernetes

Die vertikale und horizontale Skalierung vereinfacht sich durch den Einsatz von Container-Virtualisierung. Bei großen Archiven bietet sich eine Container-Orchestrierung mit Kubernetes an.

Fragen Sie bei Ihrem IT-Dienstleister nach.

Folie 42:

Ich habe Ihnen gezeigt, wie das OAIS-Referenz-Modell mit BPMN und REST umgesetzt werden kann.

Die Kernaussage meines letzten AUdS-Vortrages möchte ich an dieser Stelle leicht modifiziert wiederholen:

Die Archival Storages werden mit REST gekapselt, die Prozessmodellierung und Prozessverarbeitung erfolgt mit BPMN, alles wird über eine integrierende Benutzeroberfläche dem Nutzer zur Verfügung gestellt und die vorhandene Rechteverwaltung kann mit dem Identity and Access Management genutzt werden.

Folie 43:

Abschließen möchte ich meinen Vortrag mit zwei Empfehlungen.

Nutze Technologien mit offenen Standards zum Bau moderner Web-Anwendungen.

Use open-standards technologies to build modern web apps.

Baue für die Änderung und nicht für die Ewigkeit.

Build to Change Instead of Building to Last.

Folie 44: ausgeblendet

Mit meinem Vortrag wollte ich den „leichtfüßigen" Ansatz für eine software-technische Umsetzung des OAIS-Referenzmodells vermitteln.

Folie 45:

Schreiben Sie mir, bei Fragen zum Vortrag und zum Framework und wenn Sie in einem Online-Seminar die praktische Benutzung ausführlich vorgestellt haben möchten.

Herzlichen Dank für Ihre Aufmerksamkeit.

oais-obermeit@gmx.de

frank.obermeit@la.sachsen-anhalt.de

Die eben erwähnten exemplarischen Umsetzungen von OAIS-Referenzmodellfunktionen habe ich natürlich ausführlich im DIMAG-DAN-Archivverbund vorgestellt. Es war offensichtlich, dass fachliche Anforderungen innerhalb kürzester Zeit umsetzbar sind.

Folie 46 - 50: weitere zusätzlich Folien

Anmerkung *Das Framework steht seit August 2019 bereit und wird bis zum 30. September 2020 mit der hier vorgestellten IAM-Komponente vervollständigt und abgeschlossen.*