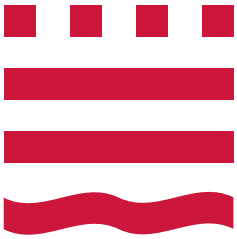


Technische Hochschule Brandenburg
University of Applied Sciences

BACHELORARBEIT

Serverless / Serverlose Architekturen für Konventionelle Webanwendungen

Vorgelegt von: Dragoljub Milasinovic
Matrikelnummer: 20140076

am: XX. Monat XXXX

zum
Erlangen des akademischen Grades

BACHELOR OF SCIENCE
(B.Sc.)

Erstbetreuer: Prof. Dr.-Ing. Schafföner
Zweitbetreuer: Jonas Brüstel, M.Sc.

Inhaltsverzeichnis

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich die von mir eingereichte Masterarbeit selbstständig verfasst, ausschließlich die angegebenen Hilfsmittel benutzt und sowohl wörtliche, als auch sinngemäße entlehnte Stellen als solche kenntlich gemacht habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Brandenburg an der Havel, 21. September 2017

Dragoljub Milasinovic

1 Abstrakt

Pomodoro @Deutch @English

2 Einleitung

Idee-Ausführung-Markt

Die Faktoren am Anfang einer technologischen Umsetzung einer Idee sind:

- Prof of Concept
- Time-To-Market
- Cost of Human Resources:: Skill-shortage
- Technical technological details
- Profitability

2.1 Motivation

Auf dem Weg zur technologischen Umsetzung einer neuen Idee liegen unbekannte Schwierigkeiten bei der Entscheidungen über deren Umsetzung hinsichtlich auf den Architekturentwurf, die IT Infrastruktur, die Drittanbieter von Software, der Auswahl der Infrastruktur usw. Schwierigkeiten die von spezialisierten Kompetenzen, Fertigkeiten und „Know-How“ bedürfen. Gehören jedoch nicht immer zum Problem des Domäns der Anwendung.

Für dieses Problem wurde Function as a Service, kürz FaaS4.4, als Lösung unter der Rubrik „ Serverless7.2“ von den Hauptanbieter von „Cloud“ Technologien vorgestellt.

Der Begriff Serverless weist darauf hin, dass die Verwaltung der darunter liegende Serverinfrastruktur von Cloudanbieter übernommen wird. Jedoch begränzt FaaS dessen Bedeutung durch die Definition des Programmiermodells, nämlich eine Funktion, oder auch „ Nano-Microservice“.

Im Rahmen des Cloud-Computing handelt es sich in dieser Arbeit um eine Untersuchung der Serverless Architekturen am Beispiel einer Konventionellen Webanwendung. Dabei wird besonders geachtet ob und wie solche Technologien die Umsetzung erleichtern. Die Entwurfsmuster und die Kernfunktionalität werden mit ausschließlich Serverless Technologien am Beispiel von Kompetenz Matrix (KOMA) mit AWS umgesetzt.

2.2 Ziel

Das Ziel ist ein Minimal Viable Product (Mivip), kürz Mivip, in Form von einer Single Page Application (SPA), kürz SPA11.5, mit ausschließlich Serverless Technologien vor zur Verfügung zu stellen.

Nach der Umsetzung werden die Erfahrungen und Ergebnisse ausgewertet, um dem Leser bei dem Entscheidungsprozess bei der Umsetzung einer Webanwendung besser zu Informieren.

Die Webanwendung soll möglichst für zukünftige Änderungen flexibel sein.

2.3 Aufbau der Arbeit

Zuerst wird den Leser in die Klassischen Service Modellen4 auffrischt. Zunächst werden die technische Anforderungen und die dazugehörige Beispiele des Serverless Ansatzes erläutert. Das nächste Kapitel überblickt die aktuelle Serverless Angebote der größten Cloud Anbietern. Anschließend trifft der Leser den Kern der Arbeit bei der Analyse und Darstellung von Serverless Architekturen8 fokussiert auf Amazon Web Services (AWS), kurz AWS.

Serverless7.2 Technologien eingeführt, das Programmiermodell vorgestellt und die Entscheidungsprinzipien7.2 erläutert. Als Zweites wird KOMA 9.2 als vorläufiges Beispielanwendung und deren Anforderungen vorgestellt. Nach dem Überblick, lässt sich die Umsetzung besser „Bottom-Up“verstehen. Am Ende wird es darüber diskutiert, welche TradeOffs entstehen und die Zukünftperspektiven von Serveless Technologien/ Architekturen.

3 Grundlagen

3.1 warum Cloud

Ziel: Anwendung Wichtig ist als Software Architekt, Entickle oder ProjMan die Specific properties von Cloud Angebot zu verstehen und entsprechend die Anwendung zusammen bauen.

Ziel: Software as a Service Cloud Service Model

Ziel: IaaS, PaaS

Cloud Properties : on-demand self-service, broad network access, measured service (pay-per-use), resource pooling and rapid elasticity

Cloud App Props : Isolated state, Distribution, Elasticity, Automated management, Loose coupling

3.2 SOA

Service Orientierte Architektur SOA unterlegt die Annahme dass, ein System aus mehreren kleinen, austauschbaren und entkoppelten Diensten bestehen kann. Microservices und Serverless versuchen die Komplexität der SOA anzusprechen. Beide Ansätze zwingen auf Separation of Concerns, häufige Deployments und Heterogeneous DSL Domain Specific Language selection. Auf einer Seite Microservices können ihren Zustand und Daten halten und werden mithilfe von „Frameworks“implementiert. Auf der anderen Seite Serverless sind Zustandlos. ConnectWise[1], Netflix[2] und UNLESS[3] sind beispiele von Unternehmen die von Serverless Architekturen profitieren.

[1]ConnectWise: `https://aws.amazon.com/solutions/case-studies/connectwise/`
[2]Netflix: `https://aws.amazon.com/solutions/case-studies/netflix-and-aws-lambda/`
[3]UNLESS: `https://unless.com/`

cloud is about services and service composition@CiteEssentialsofCloudComputing

4 klassische Service-Modellen

Iaas und PaaS

4.1 IaaS

IaaS can be described as a service that provides virtual abstractions for hardware, servers, and networking components. The service provider owns all the equipment and is responsible for its housing, running, and maintenance. In this case, AWS provides APIs, SDKs, and a UI for creating and modifying virtual machines, their network components, routers, gateways, subnets, load balancers, and much more. Where a user with a physical data center would incur charges for the hardware, shelving, and access, this is removed by IaaS with a payment model that is per-hour (or per-use) type. [You15]

4.2 PaaS

While AWS itself is an IaaS provider, it contains a product named ElasticBeanstalk, which falls under the PaaS category for Cloud models. PaaS is described as the delivery of a computing platform, typically an operating system, programming language execution environment, database, or web server. With ElasticBeanStalk, a user can easily turn a code into a running environment without having to worry about any of the pieces underneath such as setting up and maintaining the database, web server, or code runtime versions. It also allows it to be scaled without having to do anything other than define scale policies through the configuration. [You15]

4.3 SaaS

AWS also provides a marketplace where a user can purchase official and third-party operating system images that provide configurable services such as databases, web applications, and more. This type of service falls under the SaaS model. The best interpretation for the SaaS model is on-demand software, meaning that the user need only configure the software to use and interact with it. The draw to SaaS is that there is no need to learn how to configure and deploy the software to get it working in a larger stack and generally the charges are per usage-hour. The AWS suite is both impressive and unique in that it doesn't fall under any one of the Cloud service models as described previously. Until AWS made its name, the need to virtualize an entire environment or stack was usually not an easy task and consisted of a collection of different providers, each solving a specific part of the deployment puzzle. The cost of using many different providers to create a virtual stack might not be cheaper than the initial hardware cost for moving equipment into a data center. Besides the cost of the providers themselves, having multiple providers also created the problem of scaling in one area and notifying another of the changes. While making applications more resilient and scalable, this Frankenstein method usually did not simplify the problem as a whole. [You15]

4.4 FaaS

Some people prefer to use the acronym FaaS (function as a service) to describe technologies like Lambda. In fact, they prefer not to use the term serverless at all. They feel that it's not accurate enough and that it needs to be constantly explained. In this book, we've been using the term serverless not as a synonym for Lambda but as a descriptor for an approach that encourages you to use a compute service, use third-party services and APIs, and employ powerful patterns and architectures (such as having thick front ends that talk directly to services using delegation tokens). Therefore, we say that serverless is an umbrella term that encompasses FaaS and that FaaS is just one aspect (albeit a very important one) of what serverless technologies and architectures have to offer [Sba17]

5 nichtfunktionalen/technischen Anforderungen zur Entwicklung des Serverless-Ansatzes

Gegenüberstellung Skalliebarkeit ok Vendor-lockin Customisation Decentralisation

5.1 Ja/Neins

Die Unterschiede zwischen den [Kin16]

Wann Nicht :

Betriebsystemabhängigkeiten erkennen: Traditionellen Computing wie EC2, wo die Entwickler root zugrifsrechte auf alle Ressources hat, in Lambda ist diser Opaque.

OS Attribute Konfigurationen: Priorisieren CPU, GPU, Networking, or Disk Speicher und Geschwindigkeit. In Lambda skalieren proportional.

Sicherheit: Lambda nicht sichtbar host-based intrusion detection systems cannot be installed, system-level access logs

Dauerhafte Prozesse. Lambda kann bis 300s. Kosten pro Monat 1Gb Lambda = 37 EC2 = 9Dollar. Wenn warten auf Requests, oder auf Callbacks geht nicht.

Wann Ja EventDriven Tasks: Lambda solves the polling problem by creating an on-demand response to particular events.

Cron. Events: EC2 als behälter von Scripts, kostet auch ohne sie auszuführen. Fehlerhafte Scripts nicht einfach zu erkennen. Skallierung auch von Fehler. Permissions zu

offen. Mit Lambda: the permissions can be much more narrowly applied, failures are much more easily noticed, deployments can be easily triggered, logs are aggregated in one place, and the underlying server management is handled by AWS.

Heavy Processing: Um autoskalling zu vermeiden wegen z.B.Bildverarbeitung. So entkoppelt sich der Empfänger und der Verarbeiter und können mehr Requests angenommen werden.

Serverless API Gateway vermeidet api servers

Selten verwendete Services. z.B <= 5 porZent average CPU t2.micro < > 3x10hoch6 = 9dollar

6 wie es sich in die Informatik einbettet

HTTP, Funktionale Programmierung, Fassaden, ... ?? Funktionale prog. abstrah. auf Infrastrukt.

6.1 Use Cases

next [Sba17] Application Backend: z.b Internet of Things IoT: push to S3, push queue to SQS and invoke Lambda.

Data Procesisng and manipulation: pipeline of :collation and aggregation of data; image resizing; and format conversion.

Real time processing and analitics: Ingestion of Data -> Kinesis Streams; if Batch size -> Process, Save, Discard -> Lambda

Legacy Api Proxy: Extra RESTful GateWay with lambda on top legacy api. Easier Usage.

Scheduled Services

Bots and Skills

next [Kin16] Shuttdown untagged EC2 instances.

Code Deploy

Process inbount mail: attachment to S3 + link to it spam filter

Detect expiring certificates

7 kurzer Überblick über Serverless-Angebote/Architekturen

7.1 Architekturen

Vergleich und Ergänzung von serverbasierte Archs. mit Serverless Archs.

Die Architekturentwurfsmuster helfen uns zu kommunizieren welches Zweck unsere Software erreichen möchte und bieten generische Lösungen für wiederkehrende Probleme bei der Softwareentwicklung.

list of Separation of concerns Reusable layers Maintenance Koma standalone snapshot Components: Controller: router, handle in-req build out-res Service: @Transaction Repository: 1-1 mapping to db

Wenn eine erfolgreiche Codeänderung von andere Änderung abhängt, soll die Architektur überprüft werden.

Für Konventionelle Webanwendung wird hier damit gehalten, als ein System das über Presentation, Data, and Application/Logik Tiers/Stufe verfügt. Jede Stufe kann mehreren Logik-Layers/Lagen enthalten, die für unterschiedlichen Funktionalitäten des Domäns verantwortlich sind. Logging wäre ein beispiel für Cross-Cutting Concern der Layers hinaus ausspannt. Die Komplexität wächst mit der Beschichtung zusammen. Tier = Major Component System eg. Presentation Data Layer = Logical Responsability or Funtionality Domain

Serverless versucht das System in Funktionen runter zu brechen und auf ihnen das sichere Zugang dem Front-End zu erlauben.

7.2 Serverless

Serverless ist ein @Glossar Web Dienst/Service von Cloud-Anbieter, wird auch als @Glossar FaaS bezeichnet. Welcher grundlegende Serverinfrastruktur vom Cloud-Anbieter wie Amazon Web Services verwaltet wird. Komplexe Probleme wie horizontale und vertikale Skalierbarkeit, Fehlertoleranz, Flexibilität werden von Kunden und Benutzer nur noch nach bedarf Konfiguriert.

Prinzipien von Serverless Architeturen: [Sba17] Rechen-Dienst dass, nach Anfrage isoliert, unabhängig und granular ausgeführt wird. SRP Single Responsability Prinzip: Funktionen werden dadurch mehr Testable/Prüfbar. Deren Vernetzung erlaubt komplexe Systeme zu entwerfen, die dank der Stateless Natur der Funktionen schnell zu skalieren sind. Diese Vernetzung kann durch einen Push-Based Event driven Pipe-line. Um die Komplexität des Systems zu reduzieren und die längerfristige Wartbarkeit zu verbessern, wird der Controller und/oder Router im Client und Third Party Dienste hinzugefügt.

In einem beispielhaften konventionellen AdServer, nach einem Click auf eine Werbung wird eine Nachricht über ein Kanal an einen Clickprozessor, derer Laufzeit eine Anwendung ist, geschickt. Im Serverless wird dieser Clickprozessor als Funktion pro Nachricht instantiiert derer Laufzeitumgebung und Messagebroker der Cloudanbieter liefert.@Cite Flower's Blog

7.3 Patterns Allgemein vs Serverless

next [Sba17] Compute as a back end for web and mobile

compute as a glue pipelines built to carry out workflows <-> Async Messag Primer [HSB+14]

Pipes and Filters [HSB+14] <-> lambda event driven Health endpoint monitoring <-> EC2 certificate Leader election -> TRansaction mngmt S3 Priority Queue Queue Based Load Leveling -> Not appliable for Lambda Computing Model Retry -> Runtime Reconfiguration <-> Api Gateway Zero Downtime on Redploy Scheduler Agent supervisor -> Not needed if Design to Failure. Sharding -> Divide DataStore to

Scale better -> Dynamo db + S3 Static Content Hosting -> Solved by S3 Throttling -> Scallability of Serverless Solved vs Transaction Vallet key <-> Auht0

Befehlmuster wird bei Fusekiserver11.1 als erteiler der Httpanfrage indem die SparQL Abfrage weiterleiten kann. Daher wird ein Pfad haus/hunde und haus/katze zur gleichen Funktion führen. Dieser kann aber offline gehen, also mehreren priorisierten MessagePattern als Queue vor eine oder mehreren Lambdas zu setzen absichert die Stabilität des Systems und entkoppelt Komponenten @RoundRobin?? BSRN. Die Verkettung von Funktionen mittels „Pipes“erlaubt die mehrfache Filterung von Daten.

7.4 Guidance

Caching -> Lambda Reads Same source and Process Compute partitioning -> Lambda per use. solved Data Consistency -> Embrace Eventual Consistency in Distributed DBs Data partitioning <-> Sharding Instrumentation and Telemetry Guidance -> Errors Handling Service Metering -> understand future use of services

8 Fokus auf AWS, Vorstellung der AWS-Serverless-Angebote

Per Type: The Company API Gateway : AWS Simple Notification Service (SNS) : AWS vs Cloud Pub/Sub : Google Simple Storage Service (S3) : AWS vs Cloud Storage : Google Simple Queue Service (SQS) : AWS Simple Email Service (SES) : AWS Relational Database Service (RDS) : AWS DynamoDb : AWS CloudSearch : AWS CloudFront (CDN) : AWS DNS management (Route 53) : AWS Caching (ElastiCache) : AWS

Elastic Transcoder : AWS Kinesis Streams : AWS

Cognito : AWS AuthO : AuthO Firebase : Google Stack Driver Logging : Google Cloud Machine Learning Engine : Google Cloud DataFlow : Google -> Stream batch pipelines Big Query : Google

9 Beispiel(!)-Fall KOMA

9.1 Anforderungen Analyse

Zu den Anforderungen

- Mit dem EQF vergleichbare Kompetenzeindefinitionen
- Von Browser abrufbar
- Private Datenspeicherung u.d Login
- Zukünftige Erweiterungen berücksichtigen
- Ertrag von großen Nutzlastschwankungen

9.2 KOMA

Beispiel Anwendung

Die heutige Rahmenlehrpläne sind nicht mehr fachlich sondern nach Kompetenzentwicklung orientiert, um Kompetenzprofile für Lerner zu gestallten. @cite aber wenn? Das Modell von KOMA basiert auf dem Grundmodell von @ref „European Qualifications Framework Semantics“und dem deutschen Qualifikationsrahmen. <– really?

„KOMA“ist ein Akronym für Kompetenz-Matrix. Die Umsetzung der Anwendung soll die von einem Individuum oder Schüler erworbene und zu erwerbenden Kompetenzen und deren Niveau nachvollziehen. Der Kompetenzstand einer Person ist mit dem EQF-Rahmen @Acro vergleichbar, und daher International anerkennbar.

Wenn diese Anwendung in Bildungsintitutionen eingesetzt wird, dienen die Rahmenlehrpläne als Leitpfad für die Belegung der einzelnen Kompetenzen und KOMA für die

Organisation der einzelnen Fachrichtungen. Der Kern solcher Org. ist die Zuweisung von Aktivitäten auf vordefinierten Kompetenzen. Aktivitäten lassen sich einzeln oder in einer Sequenz anordnen. Sequenzen werden in LVen zusammengestellt. So können Aktivitäten, Sequenzen und Kompetenzen als gestalltungsmittel für LV benutzt. Das Modell verfügt von eine Figur um die erledigte Aktivitäten auszuwerten, nämlich Evaluation.

Wegen der Kompetenzorientierung, wird Kompetenz als Unit-Of-Work betrachtet für Modelierungszwecke.

Hochschule Beispiel Beispiel: Der Lehrplan fordert die Kompetenzen K in K1, K2, K3 .. Kn für den Bachelordiplom. Nach deren Manuellen Eingabe werden automatisch ihren Requirenments/Anforderungen/Abhängigkeiten Baum erzeugt. Dadurch können bereiche des Baums als LV betrachtet werden und Kompetenzen aufeinander bauend für eine Klassenstufe/Semestergang sequenziert werden. Die Auswertung der Ergebnisse der Aktivitäten und die Aktualisierung des Kompetenzstands des Schulers folgen.

Schule Beispiel Allgemeines Beispiel:

Beispielsweise drückt sich die Kompetenz beim Erwerb einer Fremdsprache – wenn mankommunikative Handlungsfähigkeit als Bildungsziel vorgibt – darin aus, wie gut man kommu-nikative Situationen bewältigt, wie gut man Texteunterschiedlicher Art verstehen und selbst adressatengerecht Texte verfassen kann, aberunter anderem auch in der Fähigkeit, gram-matische Strukturen korrekt aufzubauen und bei Bedarf zu korrigieren, oder in der Fähigkeitund Bereitschaft, sich offen und akzeptierend mit anderen Kulturenauseinander zu setzen. Standards für das Fremdsprachenlernen müssen diese Teilkompetenzen darstellen und je-weils verschiedene Niveaustufenunterscheiden (vgl. Anlage a). Hierbei spielen nicht nurkognitive Wissensinhalte eineRolle;diese sind vielmehr– wie Weinert im obigen Zitat her-vorhebt und das zuletzt genannte Beispiel der sog. Interkulturellen Kompetenz besonders deutlich macht – mit Einstellungen, Werten und Motiven verknüpft. Kompetenzen spiegeln diegrundlegenden Handlungsanforderungen, denenSchülerinnen und Schüler in der Domäneausgesetzt sind. Durch vielfältige, flexible und variable Nutzung und zunehmende Vernetzungvon konkreten, bereichsbezogenen Kompetenzen können sich auch „Schlüsselkompetenzen“entwickeln, aber der Erwerb von Kompetenzen muss – wie Weinert (2001) hervorhebt – beim

systematischen Aufbau von „intelligentem Wissen“ in einer Domäne beginnen. - Jede Kompetenzstufe ist durch kognitive Prozesse und Handlungen von bestimmter Qualität spezifiziert, die Schüler auf dieser Stufe bewältigen können, nicht aber Schüler auf niedrigeren Stufen. Zum Bildungsstandard gehört, dass für einzelne Jahrgänge festgelegt wird, welche Stufen die Schülerinnen und Schüler erreichen sollen.

Use Case Ein Nutzungs-Fall aka. Use-Case: -Als Professor, will ich eine Auflistung der erworbene Fertigkeiten einer Klassenstufe abrufen können.

-Als Professor, will ich das Kompetenzniveau einer Kompetenz eines Studenten und deren Fertigkeiten abrufen können.

-Als Student, will ich mein Kompetenzprofil für meinen Lebenslauf benutzen.

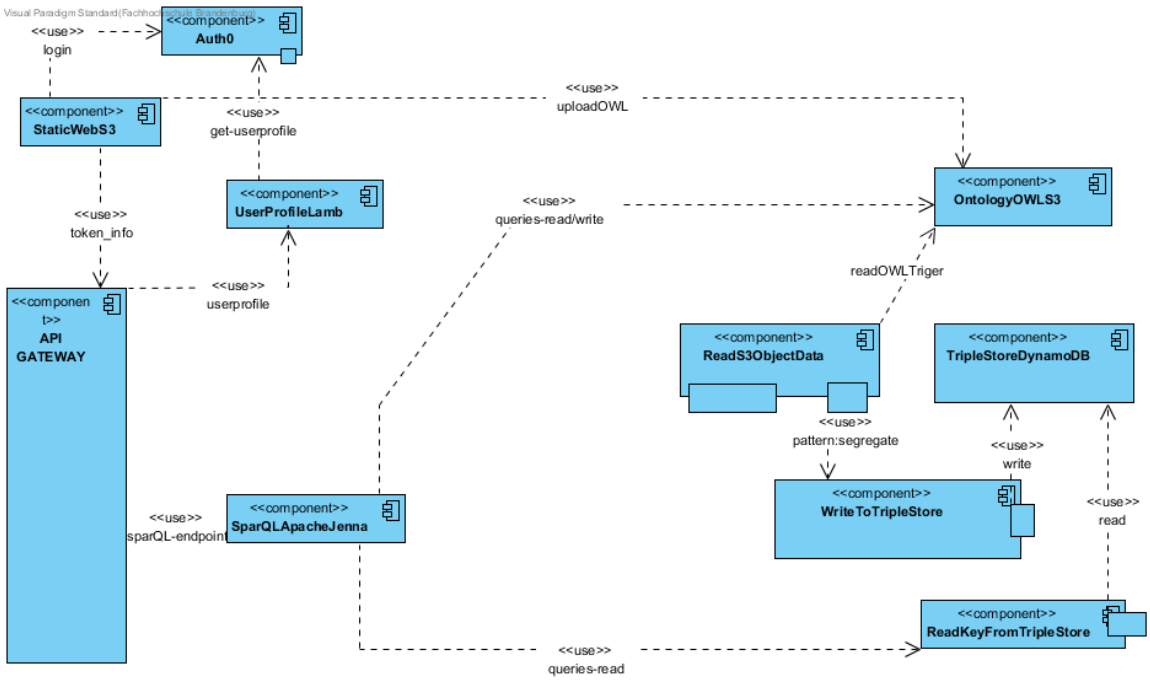
10 EntwurfII

11 Umsetzung

Die grundlegende Vorgehensweise bei der Umsetzung dieses Projekts wird Analyse, Entwurf, Implementierung und Test sein. Der Forschender Charakter dieses Projekts lässt sich nicht Testgetrieben implementieren. Das

11.1 Komponenten Übersicht

Um dem Leser einen Anhaltspunkt zu verleihen, werden hier die Softwarekomponenten beschrieben.



legacy api proxy [Sba17] lambda<convert/invoke> fuseki-server

nice: graphQL= json matcher over multiple DBs

Tabelle 11.1: Vergleich relationalem mit ontologischem 11.2.2 Schema

Eigenschaft	Relational	Ontologisch
Darstellung Welt-Annahme	Existiert nur	Existiert mindestens
Individual	muss Unique	kann >= 1
Info	Ableitung = x	ja
Oritentation	Data	Bedeutung

11.2 Datenhaltung Analyse und Auswahl

Die Polymorphe Gestaltung von Kompetenzmodellen und deren zukünftige Weiterentwicklung stellt die Benutzung des traditionellen RDBMS für KOMA in Frage. Im Folgendem werden die Eigenschaften von relationalen mit ontologischen Schemas verglichen.

Es wurden folgende Faktoren für die Entwicklung einer Ontologie erkannt:

- Zirkuläre Abhängigkeiten sind zugelassen
- Equivalenzklassen zwischen KOMA und andere Ontologie soll die Anerkennung von erworbenen Kompetenzen ermöglichen
- Die Begriffe der Kompetenzen können sich ändern oder die Ontologie soll erweitert werden
- Die Verbindung zu externen Ontologien kann neues Wissen ableiten

6.2.2 [Oro12]Benefits on Interoperability and Linked-Data by Using Ontology Engineering

Es handelt sich daher um eine „Linked Data Driven Web Application“.Dieser Begriff gehört zum „Semantic Web“,der in der Sektion der Ontologie11.2.2 weiter erläutert wird.

11.2.1 Semantic Web

Das „Semantic Web“ist eine Erweiterung des herkömmlichen Web, in der Informationen mit eindeutigen Bedeutungen versehen werden [GOS09]. set of standards and best practices for sharing data and the semantics of that data over the Web for use by applications [Bob13] . Diese Bedeutungen werden für Maschinen durch Ontologien

dargestellt. Die Ontologien werden in der OWL2 Spezifikation von W3C[1]beschrieben. Die hälfte der Linked Datenbasis sind veroffentlicht.

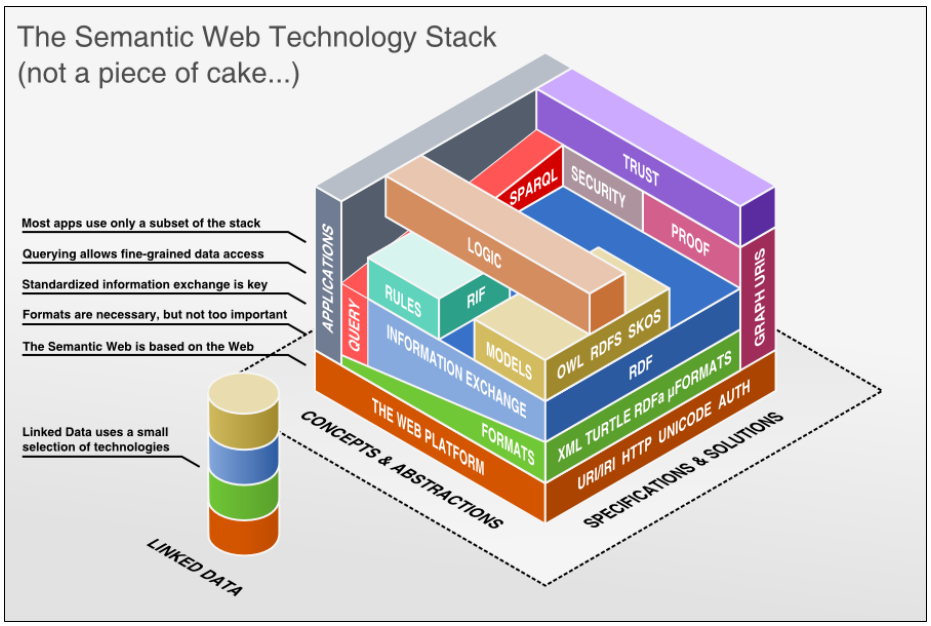


Abbildung 11.1: Überblick von benutzten Technologien

11.2.2 Ontologie

Konzeptuelle Modellierung

Eine Ontologie ist eine formale Spezifikation über eine Konzeptualisierung [SBF98]. Die Denotation jeweiliger dargestellten Signifikanten lässt sich durch seinen weltweit eindeutigen Präfix identifizieren. Deren Beziehungen können auch zu externen Ontologie-signifikanten verweisen und dadurch ein Consensus über Begrifflichkeiten.

Während der Umsetzung wurde Protege[2] benutzt. Der Entwurf der Ontologie wurde nach Ontology-Engineering-101 durchgeführt:

Die auf die Entwicklung vorbereitende Aufgaben folgen.

Software Plattform Zunächst es wird die Plattform der zu entwickelnde Software der Anwendung festgelegt. Die Daten können in Produktion mittels einer „Triple-Store“mit grundsätzlich 4 alternativen gespeichert:

[1]WWW Consortium `https://www.w3.org/standards/semanticweb/`

[2]`http://protege.stanford.edu`: "This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health."

- Monolitische Triple Speicherung
- Property Werte
- Vertikal Partitionierte Tables
- Hexastore

Die für KOMA unterstützende Infrastruktur wird ein Rechner mit Browser wie Firefox oder Chrome und die AWS Dienste wie Lambda, API Gateway und S3. Das Kommunikationsprotokoll auf der Anwendungsebene zwischen Endpunkten ist HTTP v1.1. Eine Ontologie kann mittels Sparql abgefragt werden. Sparql ist eine Abrfragesprache für RDF, Spreadsheets, XML und JSON formate [Bob13]. Wir beschreiben die Ontologie mit dem Datenmodel RDF und der TURTLE syntax.

Für die Entwicklung und als MVP eignet sich S3 @Acro für die Speicherung von großen Datenmengen. Der nächste Schritt wird eine Indexierung der URIs jeweiliger Subject, Predicate und Object in DynamoDB.

Die Ausführung von Abfragen auf die Ontologie wird direkt mithilfe von „Jena SPARQL ARQ“Framework realisiert. Deren Quelldateien werden in einer Lambda @Acro Funktion ausgeführt.

Der Umfang der Ontologie ist die Konkrete Fragestellungen für die pädagogische Diagnostik und Intervention von Fortschritten der Studierenden im Kompetenzrahmen der Lehrpläne.

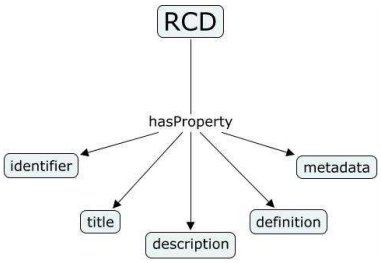
- Welche stand von Kompetenzen hat eine Klassenstufe?
- Welche Kompetenzrückstände oder Auffälligkeit sind von eine Klassenstufe erkennbar?
- gegeben sei ein Kompetenzstand, welche Leistung kann ich von eine Klassenstufe erwarten?

Um die Neuerfindung des Rades zu vermeiden, die Recherche ergab einen aktuell öffentlichen graphischen ontologischen Entwurf [RMG14] siehe11.2 der in Moodle mit eine Relationalen Datenbasis und PHP umgesetzt. Nach dessen Ontologien wurde mittels Watson[3] und LOD[4] nichts öffentlich gefunden.

[3]Watson `http://watson.kmi.open.ac.uk/WatsonWUI/`
[4]LOD Cloud `http://lod-cloud.net/`

Bei der Analyse lässt der Entwurf und dessen Dokumentation freie Interpretation über Begriffe und deren Zweck, beispiele davon sind „isComposedOf“, „subsumes“. Ein Standard zur graphischen Darstellung ist zur Zeit?? noch nicht anerkannt. Obwohl Graphische Benutzeroberfläche @Cite research-gate graphol

Anderseits wurde ein „EQF Framework[5]“für Ontologien beschrieben, aber nicht öffentlich umgesetzt. Es bietet dabei eine europäisch anerkannte Definition von Kompetenz, nämlich RCD [DCAB17]



Daher folgt eine beispielhafte Auflistung der auf unseren Anwendungsfall angepasste und ergänzende Interpretation der dargestellten Terminologie des Entwurfs und der RCD.

Auflistung der Terminologie 11.2 Die zwei Leitmotive sind auf eine Seite Kompetenzanforderungen: sie legen fest, über welche Kompetenzen ein Schüler, eine Schülerin verfügen muss, wenn wichtige Ziele der Schule als erreicht gelten sollen. Systematisch geordnet werden diese Anforderungengen in Kompetenzmodellen, die Aspekte, Abstufungen und Entwicklungsverläufe von Kompetenzen darstellen [Kli03]. Und auf der Anderen nach Kompetenz als die bei Individuen verfügbaren oder durch sie erlernbaren kognitiven Fähigkeiten und Fertigkeiten, um bestimmte Probleme zu lösen, sowie die damit verbundenen motivationalen, volitionalen und sozialen Bereitschaften und Fähigkeiten, um die Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können [Wei02].

[5]EQF Beschreibung `https://ec.europa.eu/ploteus/content/descriptors-page`

Tabelle 11.2: Terminologie

Darstellung	Begriff	Bedeutung
Competence	Kompetenz	Kontext adäquate Anwendung von Fertigkeiten
CompetenceProfile	Kompetenzprofil	Sammlung von Kompetenzen
Competence	Kompetenz	Kontext adäquate Anwendung von Fertigkeiten
Skill	Fertigkeit	Das systematisch Tun-Können einer Aufgabe
Knowledge	Wissen	Verstehen von Informationen
Other	Andere	Nicht kategorisiert aber zu beachten
ProficiencyLevel	Kompetenzniveau	Bewertung einer Kompetenz
PerformanceIndicator	Kompetenzmaß	Kriteria zur Auswertung
Course|LV	Lehrveranstaltung	Sammlung von Sequenzen
Sequenz	Sequenz	Sammlung von Aktivitäten
Activity	Aktivität	Lernaktivität
Action	Aktion	Lerntat
Actor	Täter	Aktiver Agent aka Lerner
Learner	Lerner	Kompetenz erwerbender Agent
CompetencyRecord	Kompetenzaufnahme	–
EvidenceRecord	Kompetenzbeweis	–
EvidenceSource	Kompetenzherkunft	–
todo	todo	todo
todo	todo	todo

Klassenhierarchie Um Wissen abzuleiten und Inkonsistenzen zu identifizieren werden die Klassen und Properties durch Restrictions versehen oder definiert. Folgende Restrictions wurden angewendet.

Tabelle 11.3: Klassendefinition

Klasse	Definition
LV	*onlySequenz*

@Build : ableitungs Baum von Protege an stelle von Tabelle

Tabelle 11.4: Properties

isComposedOf	$\forall Class \equiv onlyClass$
subsumes	$\exists Class \equiv someClass$

Properties und Attributen von Klassen

11.2.3 RDF

Die bisher erreichte Analyse des Domänsproblems soll nun anhand Protégé in einen RDF format beschrieben werden. Der Menschen lesbarsten RDF Format ist TURTLE. Seine Syntax besteht aus Triples mit „.“beendete Zeilen. Ein Triple stellt ein Fakt dar, un besteht aus einem Subjekt, einem Prätikat und einem Objekt. Diese können sich in TURTLE verschachteln wie das folgende Listing11.1 zeigt.

Listing 11.1: Darstellung von Triples in TURTLE

```
:EvidenceRecord rdf:type owl:Class .

:actionPerformed rdf:type owl:ObjectProperty ;
rdfs:domain :EvidenceSource ;
rdfs:range :Action .
```

11.2.4 Sparql

Um aus Ontologien Informationen zu entnehmen, wird die Abfragesprache „SPARQL“@Acro verwendet. Diese ist ähnlich zu SQL. Mit dem Programm „Protétegé“können SPARQL Abfragen lokal ausgeführt werden. Sparql ist auch ein Protokol. Als solches, kann das entsprechende sparql endpoint HTTP Responses des „Content-Type:

application/sparql-results+xml; charset=UTF-8“wie es bei der dbpedia[6] sparql endpoint, der mit Virtuosso server in HTML formatierte Ergebnisslisten zurückliefert.

```
1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX dbo: <http://dbpedia.org/ontology/>
3
4 select ?author where {
5
6 ?author rdf:type dbo:Writer .
7 ?author dbo:notableWork ?work .
8
9 } LIMIT 20
```

Die Terminologie von KOMA wird von außen als Vokabular gesehen. Dessen DOM elementen können annotiert werden, so dass maschine unterschiedliche Vokabuläre verstehen können, siehe HTML5 Microdata[7]

```
1 <table class="sparql" border="1">
2 <tr>
3 <th>author</th>
4 </tr>
5 <tr>
6 <td><a href="http://dbpedia.org/resource/Abbie_Hoffman">http://dbpedia.org/resource/
      Abbie_Hoffman</a></td>
7 </tr>
8 ...
```

Die einfachste Abfrage in Sparql wählt alle Triples vom Datenmodell.

```
1 SELECT * WHERE { ?s ?p ?o .}
```

Am Beispiel von KOMA, konkatenieren die Ergebnisse mit Zwei Abfragetriples dank eine Hilfsvariable. Die folgende Abfrage ließe sich „Wähle alle Properties des Graphes und wähle alle dessen Subjekten mit Alice als Objekt“

```
1 PREFIX owl: <http://www.w3.org/2002/07/owl#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4 PREFIX koma: <https://s3-us-west-2.amazonaws.com/ontology.thb.de/koma-complex.owl#>
5 SELECT ?x WHERE {
6 ?y rdf:type owl:ObjectProperty .
7 ?x ?y koma:Alice .
8 }
```

[6]Dbpedia dbpedia.org/sparql/
[7]Microdata schema.org

Nach der Modellierung in Protégé wird die OWL Datei in einem S3 Bucket verpackt und dadurch deren Zugriffsrechte konfiguriert, so dass in Produktion nur eine berechtigte Anfrage z.B. von eine Lambda Funktion oder DynamoDB angenommen wird.

```
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": "s3:*",
"Resource": "*"
}]}
```

So dass auch die Benutzer von KOMA solche Abfragen stellen können wird ein „Sparql-endpoint“mit Hilfe von Apache Jena ARQ, ein Sparql-Engine, zur Verfügung gestellt. Dieser Sparql-Endpoint entspricht der Repositoryschicht der Anwendung und wird nach Anfrage von der Ontologie in S3 mittels Sparql JSON Objekte zurückliefern. Eine Lambdafunktion arbeitet als Schnittstelle 11.2 zwischen die ARQ Bibliothek, den Client und die darunterliegende Infrastruktur.

Listing 11.2: Schnittstele Lambda

```
public class Handler implements RequestHandler<RequestClass, String> {

@Override
public String handleRequest(RequestClass input, Context context) {
return new Controller(System.getenv(ENV_BUCKET)
, Regions.US_WEST_2.getName())
.executeQuery(request.getQuery()
, request.getBucketKey());
}

}
```

Um sich von die durch Lambda entstandene Abhängigkeit möglichst entkoppelt [Flo99] zu halten, wird das Modul von Jena-ARQ nur als externe Abhängigkeit des Lambdaprojektes zugewisen. Die AWS Lambda stellt eine Interface zur Verfügung und der Nutzer die zu benutzende Bibliotheken zusammen in eine JAR Datei verpackt. Dem entsprechend wird der Buildscript von Gradle [Mus14] angepasst11.3.

Listing 11.3: Abhängigkeitenverwaltung für Lambda in Java

```
plugins {
id 'com.github.johnrengelman.shadow' version '2.0.1'
```

```
3 id 'java'
4 }
5 shadowJar {
6 mergeServiceFiles()
7 }
8 ...
9 $ gradle clean build shadowJar
```

Die AWS Console lässt die Funktionen hochladen und konfigurieren, ohne nötige Kenntnisse von der AWS-CLI. Zusätzlich muss der RESTful Pfad auf die verantwortliche Funktion in API Gateway zugewiesen werden.

Um den Datenmodell möglichst simpel programmatisch abzufragen wird zunächst dessen Schnittstelle11.4 definiert.

11.3 Abtrennung des Monoliths

Im Folgendem es wird beispielhaft eine abtrennung einer JEE Anwendung.

JEE.war „And“Test vs SRP Identifizieren der Concerns und ihre Trennung:

Einloggen Login beispiel: user + password + userData -> user + passord -> uuID -> userData refactor and share common code

Autorisierung und Authentifizierung. Einleitung

Auth0 bietet Authentifizierung as a Service an. Der Benutzer erhält einen JSON Web Token JWT und schickt ihn Encoded JSON Web Signature JWS oder JSON Web Encription zur Anwendung mit.

Einloggen: 0Auth Google gibt token, der wird in Lambda überprüft, Session in oauth.com verwaltet

Dynamo DB Funtion: Read + Write + updateS3 zu Split S3, Split R/W

Datenspecherung Architektur: DynamoDB: speichert :individual als Schlussel und seine relative URL Ś3: speichert die .owl Dateien.

Lambda Funktion: Maps zwischen S3 und DynamoDB.

11.4 RESTful API

Designing a friendly Hypertext Transfer Protocol (HTTP) API means abstracting the intricate business logic and data your service uses into the four basic CRUD operations (create, read, update, delete).

Die Komplexität des darunterliegenden Datenmodells erlaubt eine RESTful [H+17] Schnittstelle nur einfache abfragen zu formulieren. Daher zusätzlich ein Endpunkt11.5 für Komplexe Sparql Abfragen, die im Body des HTTP Requests in JSON geschickt wird.

Tabelle 11.5: RESTful API

Methode	URL	Rückgabe
GET	/ontology	Information über KOMA
GET	/ontology/{individual}	RDF von Individual
GET	/page	Auflistung von Entitäten
GET	/page/{individual}	Information über diesen Fakt
POST	/sparql	Abfragenergebnis

AWS API Gateway ermöglicht die Definition, Konfiguration und das Importieren von Schnittstellen. Beispielsweise kann die Abfrage GET https://<host>/page/{individual}

Listing 11.4: Mapping Template

```
1 GET https://<host>/page/{individual}
2 ...
3 {
4 "individual" : "$input.params('individual')"
5 }
```

Damit wurde die zu erwartende Eingabe für den Sparql-Endpoint definiert. Der Zugang auf die Schnittstelle wird durch CORS konfiguriert um deren Ausnutzung zu vermeiden.

Dieser Endpunkt unterstützt nicht nur GET-Abrufe, sondern auch POST-Anforderungen mit einer Nutzlast. Unter der verfügbaren SparQL endpoints Implementierungen

11.5 Single Page Application

Da KOMA ohne Vorkenntnisse gebrauchsfertig sein soll, mit dem Fakt dass Milliarden von Desktop Geräte die Web mit einem Browser erkundigen können, lässt sich die Entscheidung über die Art der Benutzeroberfläche leicht Treffen.

Die Web Anwendung ist für alle Rechenaufgaben verantwortlich die im Browser aus dem Sicherheitssichtpunkt kein Gefahr darstellen, um den Backend oder Servers möglichst wenig auszulasten. Deswegen bietet sich eine Single Page Application an. Die SPA besteht aus ein einziges HTML Dokument. Dies vereinfacht man die Konfiguration der Authentifizierung und unterbricht den Fluss der UI-Darstellung zwischen Seiten.

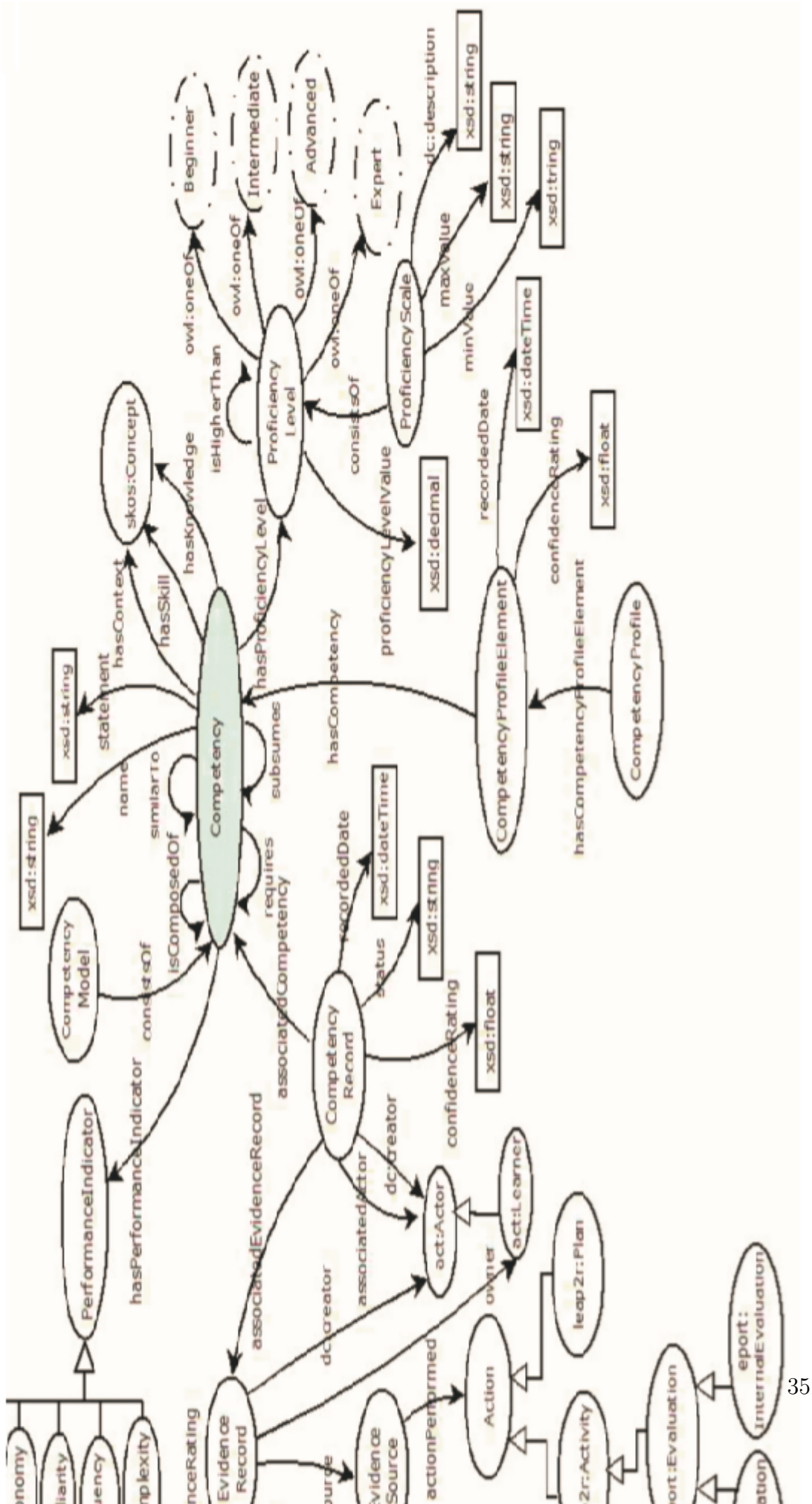
Ein konfiguriertes Anfangsprojekt/Quickstart kann mithilfe von Initializr[8] oder JHipster[9]. Für die lokale Entwicklung der Webseite werden anhand von NodeJS und NPM folgende Bibliotheken als Abhängigkeiten verwaltet: Bootstrap als Stylescheet und jQuery als Javascript-Bibliothek. Die Webseite wird Statisch mittels S3 geliefert. Dies geschieht mit einem Befehl:

Listing 11.5: Webseite veröffentlichen

```
1 $ aws --region us-west-2 s3 website --index-document index.html --error-document error.
      html 's3://koma.thb.de'
```

Da der Zugriff auf die Datenspeicherung gesichert werden soll, wird die Login-Funktionalität hinzugefügt.

[8] Initializr http://www.initializr.com/
[9] JHipster http://www.jhipster.tech/


Competency
CompetencyProfileElement
CompetencyProfile
CompetencyModel
CompetencyRecord
ProficiencyLevel
ProficiencyScale
EvidenceRecord
EvidenceSource
PerformanceIndicator
Action
skos:Concept
xsd:string
xsd:decimal
xsd:float
xsd:dateTime
Beginner
Intermediate
Advanced
Expert
hasCompetency
hasProficiencyLevel
proficiencyLevelValue
hasSkill
hasKnowledge
hasContext
statement
name
similarTo
subsumes
isComposedOf
requires
hasPerformanceIndicator
associatedEvidenceRecord
associatedCompetency
associatedActor
recordedDate
confidenceRating
status
dc:creator
owner
actionPerformed
hasCompetencyProfileElement
isHigherThan
consistsOf
minValue
maxValue
dc:description
owl:oneOf
act:Actor
act:Learner
leap2r:Plan
leap2r:Activity
eport:Evaluation
eport:InternalEvaluation

12 Bewertung

Anwendung Latenz Die entstandene Webanwendung befindet sich in US-WEST-2, Oregon, in den USA. Da keine Cache oder CDN Funktionalität weder Implementiert noch konfiguriert ist, ist die Latenz direkt proportional zur Ausführungsdauer der Lambda Funktion. @Benchmark testing curl @Lambda Monitoring

In Zeiten des Cloud computings

Frameworks und FaaS Frameworks helfen aber sind platform abhängig. Entweder JEE und JVM oder PHP. Es kann auf die Layer of Abstraction in FW verzichtet werden. Die Ersetzbarkeit des FaaS entkoppelt die Anwendung und den Entickler von der darunterliegende Technologie.

DevOps Frameworks Die benötigte Fertigkeiten für die Umsetzung einer Serverless Anwendung werden mithilfe von Deploymentframeworks gemindert. Die Aufnahme von 3.Anbieter ist deswegen notwendig. Es existieren bereits solche Hilfe wie z.B Serverlessframework@Ref

Risiko: Entickler brauchen einen guten Testplan und eine gute DevOps Strategie.<- skills shortage

Transaktionen Transaktionen können nicht parallel ausgeführt werden. Sequenciel aka Messaging Pattern. Zusammenspiel Arch. interfaces prog.modell und FW Arch 1st -> def interfaces and interactions. to program to a inteface

Eventual consistency -> event driven + ontology quality Consistenty -> koma-standalone <- transaction mgm

Vorteile Automatische Skalierung <!– größe und kleine Apps –> und Fehlertoleranz Automatisches Kapazitätsmanagement Flexible Ressourcenverwaltung Schnelle Bereitstellung der Ressourcen Exakte nutzungsabhängige Abrechnung der Ressourcen Konzentration auf den Kern des Source-Codes

Nachteile SLA Service Level Agreement: Latency, Bank:High volume Transactions, Decentralisation of Services = Challenge = Overhead, time, energy <- orchestration of events. Decentralisation vs monotithik != –komplexity Kontrollverlust Erhöhtes Lock-in Risiko

kurzlebige konfigurationen herausfinden ?? tracking? viel Konfiguration, kaum Konveniton -> .json 4 everything local testing braucht event-symulation.json

Zur Entwicklung Die Starke Komponentisierung und Dezentralisierung von Software, die Variabilität von Programmiermodellen, Frameworks, Tools, -Sprachen und dessen Entwicklungsumgebung erhöht die Komplexität des Enwicklungscyklus und hervorhebt die Bedürfnis von Tools zur Automatisierung von Tests, Deployment und Konfiguration. Also ein wohldefiniertes Handlungsplan bei der Softwareentwicklung dass von der nicht zu bearbeitende Details abstrahiert. Die DevOps Kultur spricht solche Probleme an. Neben dem Entwurf der Softwareachitekture muss, um derer Umsetzung Zeitgemäß zu gewährleisten, eine zum Projekt passende DevOps Strategie. Um Vorteil von der neuen Technologien zu nehmen, ist die Recherche nach schon existierenden DevOps Frameworks besonders wichtig. Dessen Integration in der DevOps Strategie diene für eine Agile Entwicklung.

Zum Datenmodel Aus der Anforderungs analyse einer Informations Techonologie Web Anwendung sind die Builder, Texste und dessen Darstellung das ergebniss, dass ohne Daten inhaltlos wäre. Auf eine Seite Das Relationale Datenschema stellt keine Semantik für sich dar, sondern durch von der Software entstandene Verknüpfung zwischen dem Endergebnis und dem Datenschema. Auf der Anderen Seite die RDF Daten einer Ontologie *is* das Modell.

Zum API Gateway Bei Frameworks wie JEE werden Schnittstellen zwischen Layers und Tiers bereitgestellt und diese am Laufzeit entdeckt aka Service Discovery. Im

Fall der API Gateway wird die Kopplung bei derer Konfiguration festgelegt wo derer Rekonfiguration ein neues Deployment ohne Downtime bedeutet. Die Der Quellcode der Dienste bleiben unberührt und kein Load Balancer muss rekonfiguriert werden.

Zum Serverless In dieser Arbeit wurde eine "nach buch"weise die Architektur gestalltet. Die unterschiedliche Interpretationen des Begriffs Serverless kann auch zu Kreativen anzätzen führen@AdamBien JEE Es kann daher auch als Serverless betrachtet wenn neue Quelldataien eine Docker Instance neu Erzeugen oder nur Updaten, dessen LoadBalancing auch als Serverless Quellcode verpakt werden kann.

13 Ausblick

Listings

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

Abkürzungen

da Dragoljub Milasinovic „dddd“

Dragoljub Milasinovic (da) some info

GC Garbage Collection

„Garbage Collection“ bezeichnet die automatische Speicherwaltung zur Minimierung des Speicherbedarfes eines Programmes. Garbage Collection (GC) wird zur Laufzeit durch Identifikation von nicht mehr benötigten Speicherbereichen ausgeführt. Im Vergleich zur manuellen Speicherverwaltung benötigt GC mehr Ressourcen.

Literaturverzeichnis

[Bob13] DuCharme Bob. Learning sparql. sl, 2013.

[DCAB17] Diego Duran, Gabriel Chanchí, Jose Luis Arciniegas, and Sandra Baldassarri. A semantic recommender system for idtv based on educational competencies. In *Applications and Usability of Interactive TV*. Springer, January 2017.

[Flo99] *Refactoring: Improving the Design of Existing Code.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[GOS09] Nicola Guarino, Daniel Oberle, and Steffen Staab. What is an ontology? In *Handbook on ontologies*, pages 1–17. Springer, 2009.

[H+17] II Hunter et al. Advanced microservices: A hands-on approach to microservice infrastructure and tooling. 2017.

[HSB+14] A. Homer, J. Sharp, L. Brader, M. Narumoto, and T. Swanson. *Cloud Design Patterns: Prescriptive Architecture Guidance for Cloud Applications.* Patterns & practices. Microsoft Developer Guidance, 2014.

[Kin16] W. King. *AWS Lambda: The Complete Guide to Serverless Microservices - Learn Everything You Need to Know about AWS Lambda!* AWS Lambda for Beginners, Serverless Microservices Series. CreateSpace Independent Publishing Platform, 2016.

[Kli03] Eckhard Klieme. ua: Zur entwicklung nationaler bildungsstandards–eine expertise. *Berlin 2003*, 2003.

[Mus14] Benjamin Muschko. *Gradle in action.* Manning Publications Co., 2014.

[Oro12] J. Martín Serrano Orozco. Ontologies for cloud service and network management operations. In *Applied Ontology Engineering in Cloud Services, Networks and Management Systems*. Springer, January 2012.

[Rad16] B. Rady. *Serverless Single Page Apps: Fast, Scalable, and Available*. Pragmatic programmers. Pragmatic Bookshelf, 2016.

[RMG14] Kalthoum Rezgui, Hédia Mhiri, and Khaled Ghédira. Extending moodle functionalities with ontology-based competency management. *Procedia Computer Science*, 35:570–579, 2014.

[Sba17] P. Sbarski. *Serverless Architectures on AWS: With Examples Using AWS Lambda*. Manning Publications Company, 2017.

[SBF98] Rudi Studer, V Richard Benjamins, and Dieter Fensel. Knowledge engineering: principles and methods. *Data & knowledge engineering*, 25(1-2):161–197, 1998.

[Wei02] F.E. Weinert. *Leistungsmessungen in Schulen*. Beltz Pädagogik. Beltz, 2002.

[You15] Marcus Young. *Implementing Cloud Design Patterns for AWS*. Packt Publishing Ltd, 2015.