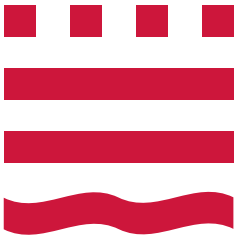

Technische Hochschule Brandenburg
University of Applied Sciences

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

1 Einleitung

Idee-Ausführung-Markt

> *" An idea is not a mockup*
> *A mockup is not a prototype*
> *A prototype is not a program*
> *A program is not a product*
> *A product is not a business*
> *And a business is not profits."*
>
> Balaji S. Srinivasan

Die vorantreibende Aspekte solcher Zustandsmaschine sind die Ausführung/Umsetzung der Idee bis zum Produkt und derer Beziehung zum Markt. Deren Details sind jedoch unbekannt und variabel.

Die Faktoren am Anfang einer technologischen Umsetzung einer Idee sind:

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

Time-To-Market Kost of Human Resources:: Skill-shortage Prof of Concept Profitability

1.1 Motivation

Auf dem Weg zur technologischen Umsetzung einer neuen Idee liegen unbekannte Schwierigkeiten bei der Entscheidungen über die Architektur der Anwendung/Pro-

jekt/Umsetzung?, der 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.

Um sich von diesem spezialisierten wissen aufzulösen wurde ...

Für dieses Problem wurde „FaaS“als Lösung unter der Rubrik „Serverless“von den Hauptanbieter von „Cloud“Technologien vorgestellt.

Im Rahmen des Cloud-Computing handelt es 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 KOMA mit AWS umgesetzt.

1.2 Ziel

@acronym MVP Minimal Viable Product : [Rad16] Das Ziel ist ein MVP Minimal Viable Product in form eine @acronym SPA Single Page Application mit ausschließlich „Serverlosen“Architekturen 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.

1.3 Aufbau der Arbeit

Zuerst wird den Leser in die Serverless 2.2 Technologien eingeführt, das Programmiermodell vorgestellt und die Entscheidungsprinzipien erläutert. Als Zweites wird KOMA 2.3 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.

2 Grundlagen

" There are only two hard things in computer science: cache invalidation and naming things."

Phil Karlton

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.

2.1 Architekturen

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

Separation of concerns Reusable layers Maintenance @Schafföner JEE1 Koma standalone snapshot Components: Controller: router, handle in-req build out-res Service: @Transaction Repository: 1-1 mapping to db

[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/`

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.

2.2 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.

2.2.1 Use Cases

2.2.2 Architecturen

2.2.3 Muster

Befehlmuster wird bei Fusekiserver3.1 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.

2.3 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.

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]. Nach Weinert (2001, S. 27f.) versteht man **Kompetenzen** 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 Bereitschaf-ten und Fähigkeiten, um die Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können“ [Kli03]

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.

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.

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.

3 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

3.1 Komponenten

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

@Diagramm

3.2 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

Tabelle 3.1: Vergleich relationalem mit ontologischem Schema

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

3.3 Datenhaltung Analyse und Auswahl

Die Polymorphe Gestaltung von Kompetenzmodellen und deren zukünftige Weiterentwicklung stellt die Benutzung des traditionellen RDBMS 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 Ontologie weiter erläutert wird.

3.3.1 Ontologie

Das „Semantic Web“ist eine Erweiterung des herkömmlichen Web, in der Informationen mit eindeutigen Bedeutungen versehen werden@Cite. Diese Bedeutungen wirden für Maschinen durch Ontologien dargestellt. Die Ontologien werden in der OWL2 Spezifikation von W3C@Ref beschrieben. Eine Ontologie ist eine Darstellung von Wissen. Präziser: eine formale Spezifikation über eine Konzeptualisierung [SBF98].

Während der Umsetzung wurde Protege[1] benutzt. Der Entwurf der Ontologie wurde nach Ontology-Engineering-101 durchgefuhrt:

[1] `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."

Die auf die Entwicklung vorbereitende Aufgaben folgen.

Zunächst es wird die Umgebung oder Software Platform der Anwendung festgelegt.

Die für KOMA unterstutzende Infrastruktur wird ein Rechner mit Browser wie Firefox oder Chrome und die AWS Dienste wie Lambda, API Gateway und S3. D Die Comsumer/Verbraucher der Anwendung wird ein SparQL-Endpoint [Bob13]. Die Clients : REST: sparql Endpoints

Die Daten können in Produktion @Anglizismus in einer „Triple-Store“mit grundsätzlich 4 alternativen umgesetzt: Monolitische Triple Speicherung, Property Werte, Vertikal Partitionierte Tables und Hexastore @Cite hpi web-sem 2.9. Für die Entwicklung und als Ansatz 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 @Acro .

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

ontol-eng 101 1-determine-scope: Die Verwaltung von Fortschritten der Studierenden im Kompetenzrahmen der Lehrpläne.

Obwohl die LOD @Acro NNN Datebasis erkannt hat und Watson @Cite Begriffe wie Kompetenz in zahlreiche Ontologien gefunden hat, konnte ich einen aktuell öffentlichen graphischen Entwurf [RMG14] @Cite onto-moodle einer Ontologie finden. Bei der Analyse lässt der Entwurf und dessen Dokumentation freie Interpretation über Begriffe und deren Zweck bzw. bie „isComposedOf“, „subsumes“. Ein Standard zur graphischen Darstellung ist zur Zeit?? noch nicht anerkannt.@Cite research-gate graphol

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

Daher folgt eine beispielhafte Enumeration der auf unseren Anwendungsfall angepasste und ergänzende Interpretation der dargestellten Terminologie des Entwurfs und der RCD. Kompetenzanforderung: Aspekte, Abstufung, Entwicklungsverlauf von Kompetenz Kompetenz: verfügbare oder erlernbare Nur Beispielhaft

Tabelle 3.2: 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

4-Define Classes and Hierarchies: Um Wissen abzuleiten und Inkonsistenzen zu identifizieren werden die Klassen und Properties durch Restrictions versehen oder definiert. Folgende Restrictions wurden angewendet.

Tabelle 3.3: Klassendefinition

Klasse	Definition
LV	only Sequenz

Alle Klassen in einer Ontologie leiten sich von „owl:Thing“. ?? Skill skos:Concept @Ref @Build : ableitungs Baum von Protege

5-Define Properties/Attributes of Classes:

Tabelle 3.4: Properties

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

ontol- engineering for methodology types : top - domain - app semantic gap:: how to find out whether 2 ontologies mean the same thing ontologies enable interoperability of metadata: design for develop mapping for comparison merging for efficient combination of ontologies learning for learn new ontos from given sets

desing: activities: management - develop - support management scheduling - control - quality assurance development pre - develop - post suport knowledge acquisition - eval - integr - merge - align->map - doc - config man

design:app Konkrete Fragestellungen für die pädagogische Diagnostik und Intervention.

welche stand von kompetenzen hat eine Klassenstufe? kompetenz-rückstände/auffälligkeit von eine Klassenstufe? gegeben sein ein stand, welche leistung kann ich von Klassenstufe erwarten? wurde skill-x zu Klassenstuf-y vergeben?

3.4 Repository

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. @Bild Protege SPARQL query

So dass auch die Benutzer von KOMA solche Abfragen stellen können wird ein „Sparql-endpoint“mit Hilfe von Apache Jena Fuseki@Cite, eine Open Source Software, zur

Verfügung gestellt. Fuseki ist ein Server zur Verwaltung von Sparqlabfragen und deren Transaktionen. @Cite CRUD und ACID RDB in sparql.

Fuseki wird Embedded benutzt. Fuseki entspricht der Repositoryschicht der Anwendung und wird nach Anfrage von der Ontologie in S3 mittels Sparql JSON Objekte zurückliefern. Eine Lambdafunktion arbeitet als Schnittstelle zwischen Fuseki, den Client und die darunterliegende Infrastruktur.

Um sich von die durch Lambda entstandene Abhängigkeit möglichst entkoppelt [Flo99] zu halten, wird das Modul von Fuseki nur als externe Abhängigkeit des Lambdaprojektes zugewisen. @Code : Gradle build [Mus14]

Listing 3.1: Abhängigkeiten für Lambda in Java

```
1
2 shadowJar {
3   mergeServiceFiles()
4 }
```

RESTful API
method url req.body res. header res.body GET https://<host>

GET https://<host>/sparql raw-sparql-query 2xx/5xx tab x? y? z?

GET https://<host>/page/individual 2xx/5xx Individuals page db artig [H+17] GET https://<host>/ontology/entitytype 2xx/5xx page db artig GET https://<host>/ontology/propertytyp 2xx/5xx page db artig

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

http://www.example.com/id/alice Identifier for Alice, the person http://www.example.com/people/alice Alice's homepage http://www.example.com/data/alice RDF document with description of Alice

3.5 DevOps

gradle clean build shadowJar

3.6 UI-IT

Die Benutzeroberfläche soll im Browser realisiert werden. Initializr bietet die Erstellung einer konfigurierten Projektstruktur an. Es wird NodeJS als Laufzeitumgebung, NPM als Packetmanager, Bootstrap als Stylescheet und jQuery als Javascript-Bibliothek benutzt.

Grund weshalb Statisch und S3::

Die Webseite wird Statisch mittels S3 geliefert. Da der Zugriff auf die Datenspeicherung gesichert werden soll, wird die Login-Funktionalität hinzugefügt.@Ref Einloggen @Code webify bucket

3.7 Einloggen

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 Query: SparQL ?x, ?y, ?z WHERE ... Datenspecherung Architektur: DynamoDB: speichert :individual als Schlussel und seine relative URL Ś3: speichert die .owl Dateien.

Lambda Funktion: Maps zwischen S3 und DynamoDB.

3.8 Patterns

Valet Key [HSB+14]

Static Content Hosting ok

Sharding ok

Compute Resource Consolidation

Command and Query Responsability Segregation CQRS <- readS3UpdateDynamo.js

4 Ergebnis und Auswertung

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

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

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

5 Zusammenfassung und Ausblick

Docke container + Load Balancer als möglichkeit.

Beispielhaftes Bild Abbildung 5.1

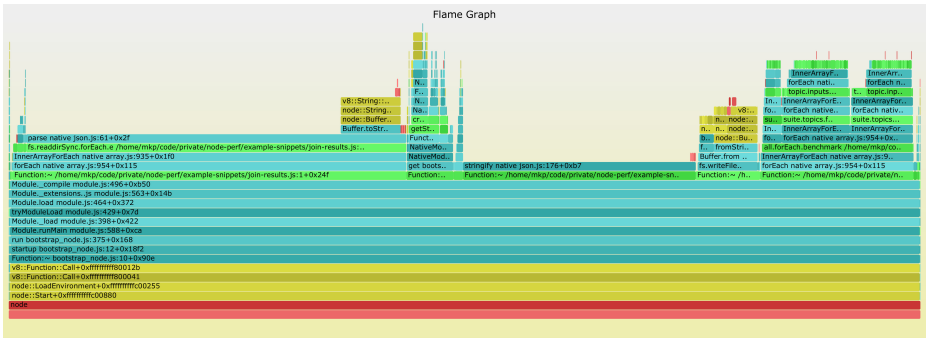

Abbildung 5.1: Beispiel Flame-Graph eines Node.js Skripts

Beispielhaftes Code-Snippet siehe Listing 5.1.

Listing 5.1: Aufnahme der „real“-Zeit

```
1 START=$(date +%s.%N)
2 node ${JS_FILE}
3 END=$(date +%s.%N)
4 DIFF=$(echo "$END - $START" | bc)
```

Hier kommt eine Bibliography-Referenz: [BME+07]

Listings

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

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.

Nach Weinert (2001, S. 27f.) versteht man **Kompetenzen** 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 Bereitschaf-ten und Fähigkeiten, um die Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können“

Beispielsweise drückt sich die Kompetenz beim Erwerb einer Fremdsprache – wenn man kommunikative 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, denen Schülerinnen 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 durchkognitive Prozesse und Handlungen von bestimmterQualität spezifiziert, die Schüler auf dieser Stufe bewältigenkönnen, nicht aber Schüler auf niedrigeren Stufen. Zum Bildungsstandard gehört, dass für einzelne Jahrgänge festgelegtwird, welcheStufen die Schülerinnen und Schüler erreichen sollen.

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

[BME+07] G. Booch, R. Maksimchuk, M. Engle, J. Conallen, K. Houston, and B.Y.P. D. *Object-Oriented Analysis and Design with Applications*. Pearson Education, 2007.

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

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

[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.

[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.

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, XX. Monat 2017

Vorname Nachname