

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	Quellen	1
2	Einleitung	1
2.1	Motivation	1
2.2	Ziel	2
2.3	Aufbau der Arbeit	2
3	Grundlagen	3
3.1	Serverless	3
3.2	KOMA	3
4	Umsetzung	5
4.1	Komponenten	5
4.2	Anforderungen Analyse	5
4.3	Serverless Kombiniert	5
4.4	Datenhaltung Analyse und Auswahl	5
4.4.1	Ontologie	6
4.5	Abfragen	7
4.6	Einloggen	7
4.7	Patterns	8
5	Ergebnis und Auswertung	9
6	Zusammenfassung und Ausblick	11

1 Quellen

Extending Moodle Functionalities with Ontology-based Competency Management
Ontology Supported Competency System
OpsWorks AWS :: Deployment Strategy
Cloud Design patterns :: Profi Patterns Man Trade Offs Arch :: Auswertung + Design
guide <- self-adaptive arch?? AWS Sol. Arch :: Best Practices AWS vs Patterns general
Amazon Web Services in Action :: Best Practices Arch Impl Cloud Design-Patterns for
AWS :: Patterns list Serverless Arch AWS :: Main :: lambda: compute as a back end

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

2.1 Motivation

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

2.2 Ziel

@acronym MVP Minimal Viable Product : [Rad16] Als Ergebnis wird ein MVP Minimal Viable Product in Form einer @acronym SPA Single Page Application zur Verfügung gestellt.

Start::Chancen: Arch + Domänen Flexibilität - Schnelle Arch Änderungen

Umsetzung der Kernfunktionalität einer Beispielanwendung mit ausschließlich „Serverlosen“ Architekturen. wenn Zeit: Identifizieren von unverzichtbaren generischen Funktionen für Serverless Anwendungen.

2.3 Aufbau der Arbeit

Zuerst wird der Leser in die Serverless 3.1 Technologien eingeführt, das Programmiermodell vorgestellt und die Entscheidungsprinzipien erläutert. Eine Zahl von möglichen Kombinationen von aktuellen Technologien mit Serverless werden beispielhaft dargestellt. Hinzu wird deren Analyse und Umsetzung durchgeführt und erläutert.

3 Grundlagen

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

3.1 Serverless

Prinzipien von Serverless Architekturen: [Sba17] Rechen-Dienst nach Anfrage dass in isoliert, unabhängig und granular ausgeführt wird. . . .

Programmiermodell Dienste werden nach Anfrage ausgeführt. Die Kosten werden nach Ausführung abgerechnet.

One size fits NOT all

Vergleich mit Microservices.

Stand der Technik: Vorgehensweise bei Traditionelle Webanwendungen: Software Architektur: "What's important". Frühe, un-/schwer- veränderbare Entscheidungen. Web Services are processes that expose their interfaces to the Web so that users can invoke them. Facilitate service discovery and meaning encoded in schemas

Design: Lambda Orchestrator -> Pool of Lambdas to use

3.2 KOMA

Beispiel Anwendung

Die heutige Rahmenlehrpläne sind nicht mehr fachlich sondern kompetenz orientiert. @cite Deren ziel ist Kompetenzprofile für Lerner zu gestallten. 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 Schuler erworbene und zu erwerbenden Kompetenzen und deren Niveau nachvollziehen. <- ? Der Kompetenzstand einer Person ist mit dem EQF-Rahmen vergleichbar, und daher Internationell anerkenntbar. Wenn diese Anwendung in Bildungsintitutionen eingesetzt wird, dienen die Rahmenlehrpläne als Leitphad für die belegung der einzelnen Kompetenzen und KOMA für die Organi-sation 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 gestaltungsmittel 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 Modellierungszwecke.

Beispiel: Der Lehrplan fordert die Kompetenzen K in K1, K2, K3 .. Kn für den Bachelordiplom. Nach deren Manuellen Eingabe werden automatisch ihren Requirements/Abhängigkeiten Baum erzeugt. Dadurch können bereiche des Baums als LV betrachtet werden und Kompetenzen aufeinander bauend für eine Klassenstufe/Se-mestergang 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.

4 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 zu implementieren.

4.1 Komponenten

@Diagramm

4.2 Anforderungen Analyse

Mit dem EQF vergleichbare Kompetenzdefinitionen.

Von Browser abrufbar.

Private Datenspeicherung. Daher Login.

Zukünftige Erweiterungen berücksichtigen.

4.3 Serverless Kombiniert

Spring Boot + Lambda Wildfly Swarm REST + Lambda Django + Lambda

4.4 Datenhaltung Analyse und Auswahl

Vergleich:: DB Schema : Ontology Welt-Annahme Existiert nur Abbild : mindestens Abbild Individual=Instanz muss unique : kann ≥ 1 Info Ableitung = x : ja
Orientation Data : Bedeutung

Es wurden folgende Faktoren für die Entwicklung einer Ontologie erkannt: Zirkuläre Abhängigkeiten sind zugelassen. Äquivalenzklassen 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.

4.4.1 Ontologie

Das „Semantic Web“ ist eine Erweiterung des herkömmlichen Web, in der Informationen mit eindeutigen Bedeutungen versehen werden @Cite. Die Bedeutung wird für Maschinen durch Ontologien ausgedrückt. Die Ontologien werden in OWL 2 beschrieben. Eine Ontologie ist eine Darstellung von Wissen. Präziser: eine formale Spezifikation über eine Konzeptualisierung.

Der Entwurf der Ontologie wurde nach Ontology-Engineering-101 durchgeführt:

Software platform ? Die Clients : REST: sparql Endpoints

Die Daten werden in Produktion @Anglizismus in einer „Triple-Store“ mit grundsätzlich 4 Alternativen: Monolithische Triple Speicherung, Property Werte, Vertikal Partitionierte Tables und Hexastore @Cite hpi web-sem 2.9. 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 .

Der Zugriff auf die Ontologie wird direkt mithilfe von „SPARQL“ realisiert. Deren Quelldateien werden in einer Lambda @Acro ausgeführt.

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

2-Consider reuse: 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 @Cite onto-moodle einer Ontologie finden. Bei der Analyse lässt der Entwurf und dessen Dokumentation freie Interpretation über Begriffe und deren Zweck bzw. die „isComposedOf“, „subsumes“. Ein Standard zur graphischen Darstellung ist zur Zeit?? noch nicht anerkannt. @Cite research-gate graphol

Andererseits 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 Enumeration der auf unseren Anwendungsfall angepasste und ergänzende Interpretation der dargestellten Terminologie des Entwurfs und der RCD.

3-Enumerate Terms: Terminologie Competence CompetenceProfile: Skill: Knowledge: Other: ProficiencyLevel:

Lehrveranstaltung: KlassenStufe: LearnActivity:

Actor Learner

4-Define Classes and Hierarchies: Learner a owl:Class Actor

5-Define Properties/Attributes of Classes:

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 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? Eine Kompetenz ist...

4.5 Abfragen

Um den

<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



4.6 Einloggen

Einloggen: OAuth Google gibt token, der wird in Lambda überprüft, Session in oauth.com verwaltet Query: SparQL ?x, ?y, ?z WHERE ... Datenspeicherung Architektur: DynamoDB: speichert :individual als Schlüssel und seine relative URL S3: speichert die .owl Dateien.

Lambda Funktion: Maps zwischen S3 und DynamoDB.

4.7 Patterns

Valet Key [HSB⁺14]

Static Content Hosting ok

Sharding ok

Compute Resource Consolidation

Command and Query Responsibility Segregation CQRS <- readS3UpdateDynamo.js

5 Ergebnis und Auswertung

Vorteile

Automatische Skalierung und Fehlertoleranz Automatisches Kapazitätsmanagement
Flexible Ressourcenverwaltung Schnelle Bereitstellung der Ressourcen Exakte nutzungs-
abhängige Abrechnung der Ressourcen Konzentration auf den Kern des Source-Codes
Nachteile:

Kontrollverlust Erhöhtes Lock-in Risiko

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

6 Zusammenfassung und Ausblick

Beispielhaftes Bild Abbildung 6.1

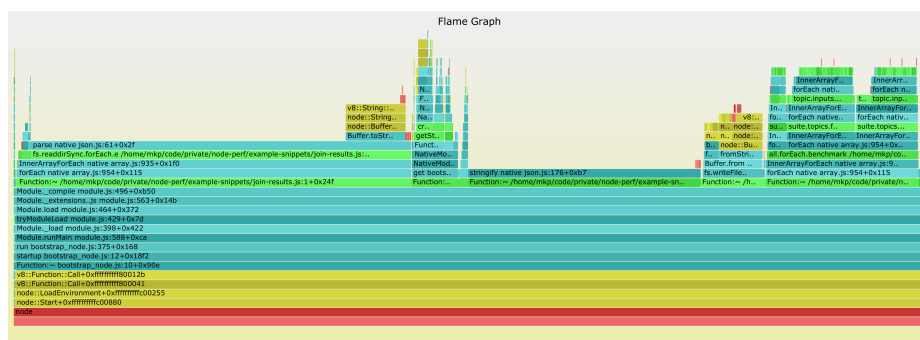


Abbildung 6.1: Beispiel Flame-Graph eines Node.js Skripts

Beispielhaftes Code-Snippet siehe Listing 6.1.

Listing 6.1: 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

6.1	Aufnahme der „real“-Zeit	11
-----	------------------------------------	----

Abbildungsverzeichnis

6.1	Beispiel Flame-Graph eines Node.js Skripts	11
-----	--	----

Abkürzungen

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.
- [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.
- [Rad16] B. Rady. *Serverless Single Page Apps: Fast, Scalable, and Available*. Pragmatic programmers. Pragmatic Bookshelf, 2016.
- [Sba17] P. Sbarski. *Serverless Architectures on AWS: With Examples Using AWS Lambda*. Manning Publications Company, 2017.

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