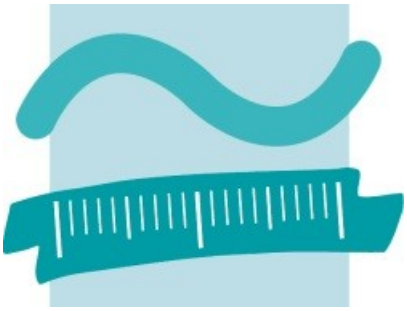
Beuth Hochschule für Technik Berlin
Fachbereich Informatik und Medien
Studiengang Medieninformatik

Serverless Programmierung: State of the Art und Evaluation

Masterarbeit

zur Erlangung des Grades Master of Science



eingereicht von: Svenja Haberditzl
Matrikel-Nr.: 83091
am: 27.02.2017 (Vorversion)
Erstgutachter: Prof. Dr. Stefan Edlich
Zweitgutachter: Prof. Christoph Knabe

Inhaltsverzeichnis

1. Einleitung

1.1 Problemstellung

Das wissenschaftliche Problem dieser Ausarbeitung bezieht sich auf die Frage warum das Programmieren mit Servern Nachteile hat und welche diese im Einzelnen sind.

Seit Jahren ist es im Bereich der Softwareentwicklung üblich, dass Programme mit Hilfe von Servern realisiert werden. Eine relativ neue Möglichkeit ist jedoch dabei sich in diesem Bereich zu etablieren. Sie wird unter dem Namen “Serverless Programming” gelistet und ermöglicht so eine Server arme Umsetzung von Software.
Warum aber ist diese Variante derzeit nur bei einigen Projekten zu beobachten, wenn sie doch Vorteile gegenüber den bisherigen Methoden bietet. Im Bereich der Softwareentwicklung und damit der Programmierung herrscht ein gewisser Stillstand vor sobald sich ein Standard durchgesetzt hat. Dieser wird auf Grund der Vielzahl an Literatur und Lösungsmöglichkeiten bei auftretenden Problemen standardmäßig eingesetzt ohne über Alternativen nachzudenken. Deshalb setzen sich Neuentwicklungen nur recht langsam und mit viel Überzeugungskraft durch.
Um die Möglichkeiten, die die Serverless Programmierung bietet, verdeutlichen zu können, sollen die Nachteile der Anwendungserstellung mit Servern betrachtet werden. Eine Umstellung weg von Servern kann nur erfolgen, wenn das Verständnis dafür hergestellt wird und ein Umdenken im Kopf stattfindet.

Im Zuge dieser Ausarbeitung soll auf die bisherigen Methoden der Softwareentwicklung ebenso eingegangen werden wie auf das Serverless Programmierung. So wird ein Überblick über beide Seiten geschaffen, der die Stärken und Schwächen aufzeigt und die Funktionsweisen verdeutlicht.

1.2 Zielsetzung

Das Ziel der vorliegenden Ausarbeitung ist es aufzuzeigen, wie das bisherige System der Anwendungserstellung aufgebaut ist und welche Nachteile sich daraus ergeben. Weiterhin soll verdeutlicht werden wieso die Zukunft in der Serverless Programmierung liegt und diese immer mehr an Bedeutung gewinnt.
Durch die Auflistung der bisher verwendeten Verfahren und Möglichkeiten in der Softwareprogrammierung soll gezeigt werden, wo im Einzelnen die Unterschiede zum Serverless Programming liegen.

1.3 Methodik

Um auf die genannte Problemstellung eingehen zu können, soll der Bereich des Serverless Programmierung beleuchtet und erläutert werden. Damit die daraus resultierenden Erkenntnisse auch praktisch umgesetzt und betrachtet werden können, um ein besseres Verständnis zu erzeugen, wird ein Prototyp erstellt. Damit die ausgangs aufgestellte These, dass Server-basierte Programmierung Nachteile hat, belegt werden kann werden zusätzlich Performance-Messungen an der im Zuge dieser Ausarbeitung erstellten Anwendung vorgenommen und deren Ergebnisse dahingehend untersucht und ausgewertet. Damit ein schlüssiges Ergebnis zu der Problemstellung aufgestellt werden kann werden Vergleiche zu den bisherigen Möglichkeiten der Softwareentwicklung gemacht. Diese Fülle an Informationen über bisherige und neue Möglichkeiten veranschaulicht durch den Prototyp sollte zur Beantwortung des Problems ausreichend sein.

1.4 Aufbau der Ausarbeitung

Zunächst werden daher die bisher üblichen Methoden aufgelistet und die am weitesten verbreiteten hinsichtlich den Funktionsmöglichkeiten und dem Aufbau näher beleuchtet. Im Anschluss erfolgt eine Erläuterung zum Serverless Programmierung ebenfalls mit Erwähnung der Funktionsweisen der Möglichkeiten und dem Aufbau dieser. Da AWS Lambda in dieser Ausarbeitung zum Zwecke der Erstellung des Prototypen herangezogen wird, soll hierzu eine detailliertere Betrachtung folgen. Nachdem der theoretische Teil beendet ist, wird der Prototyp betrachtet. Hier folgt zunächst eine Erklärung wie dieser aussehen soll und was er für eine Funktionalität erfüllt. Danach wird die Erstellung des Prototypen beleuchtet und es werden wichtige Codepassagen erklärt, um die Funktionalität besser verdeutlichen zu können. Damit die Ausarbeitung die Problemstellung zusätzlich untermauert, werden Performance-Messungen anhand des Prototypen vorgenommen. Diese beziehen sich auf den Durchsatz und die Latenz und sollen den Unterschied zu den bisherigen Methoden sichtbar machen und verdeutlichen. Abschließend erfolgt eine Zusammenfassung als Ergebnis für den Prototypen. Dabei wird auf die Umsetzung ebenso eingegangen wie auf die Messergebnisse und deren Bedeutung. Um die These, dass Serverless Programmierung Vorteile bietet zu verdeutlichen, werden nachfolgend die Vor- und Nachteile dieser Methode aufgelistet und näher erläutert. Diese resultieren aus der zum Anfang gemachten Erklärung zum Serverless Programmierung und dessen Funktionsweise sowie ebenfalls aus den Erkenntnissen der Erarbeitung des Prototypen. Am Ende folgt ein Fazit zu der gesamten Ausarbeitung, in der auf die eingangs gestellte Problemstellung und die Beantwortung dieser eingegangen wird. Weiterhin wird ein Ausblick darauf gegeben, wie in Zukunft mit Serverless Programmierung gearbeitet und wie diese Möglichkeit etabliert werden kann.

2. Server-basierte Programmierung

In der Softwareentwicklung existieren unterschiedliche Möglichkeiten und Arten eine Anwendung zu erstellen. Jeder dieser Bereiche vereint in sich weitere Möglichkeiten, die hier jedoch nicht in Gänze erläutert werden sollen. Vielmehr soll hier der Fokus auf eine Server-basierte Herangehensweise einer Softwareentwicklung liegen. Die vorliegende Arbeit handelt von Serverless Programmierung, deshalb sollen hier nur Möglichkeiten aufgezeigt werden, die nach bisherigen Mitteln mit Servern arbeiten, um im Anschluss zeigen zu können, wo genau im Einzelnen die Vor- und Nachteile der jeweiligen Durchführungsweise liegen.

2.1 Anforderungen an Software

Um eine gute Software zu erhalten, existieren einige Punkte, die bei der Entwicklung von beachtet werden sollte. Eine Missachtung eines oder mehrere dieser Punkte kann gravierende Auswirkungen auf ein Projekt haben. Schon ein kleiner Fehler kann dazu führen, dass die komplette bisherige Entwicklung gelöscht werden und das Team von vorne anfangen muss. Daher sollte von Anfang an sehr genau auf jedes Detail geachtet werden.

Im Folgenden sollen nun die zu beachtenden Punkte in Form einer Checkliste aufgezählt werden.

- → mit allen Beteiligten des Projekts ausreichend kommunizieren um Klarheit zu schaffen
- → Anforderungen früh und möglichst vollständig definieren
- → Nichtfunktionale Anforderungen frühzeitig beachten
- → richtige Architekturart wählen

→ Code gut strukturieren
→ Code leicht änderbar gestalten
→ Qualität nicht aus den Augen verlieren
→ Software möglichst gut wartbar gestalten
→ Alle Komponenten des Projekts bedenken und einbauen
→ geeignete Namensgebung für Komponenten beachten
→ Komponenten nicht über ihre Zuständigkeit hinaus entwickeln
→ Zugriffsregeln für Komponenten einhalten
→ richtige Anordnung der Komponenten beachten
→ Codeelemente klar voneinander trennen und gruppieren
→ Veränderungen im Code zeitnah testen
→ geeignete Patterns verwenden
→ Spaghetticode durch Kapselung vermeiden
→ Horizontale und vertikale Ebenen beachten
→ Code ausreichend und verständlich dokumentieren
→ Prototypen realisieren

Zu beachten ist hierbei auch, dass die einzelnen Punkte der Checkliste nicht einer wahllosen Reihenfolge unterworfen sind, sondern diese bewusst so festgelegt wurde, um möglichst von Anfang an auf wichtige Elemente hinzuweisen. So kann die Liste Schritt für Schritt abgearbeitet werden. Es empfiehlt sich dann die jeweils beachteten oder zur Kenntnis genommenen Punkte zu markieren, damit auch weitere beteiligte Personen am Projekt über den aktuellen Bearbeitungsstand Bescheid wissen und eventuell bei Missachtung eines Eintrags einspringen können.

Weiterhin ist es ratsam den Quellcode mit Hilfe von Bewertungswerkzeugen auf mögliche Fehler oder Unstimmigkeiten hin testen zu lassen. Beispiele für Software dieser Art stellen SonarQube, PMD, FindBugs oder CheckStyle dar. Mit SonarQube lassen sich detailliertere Informationen zu der getesteten Anwendung erzielen, weshalb die

Verwendung dieses Programms anzuraten ist. Die Qualität einer Anwendung kann so bemessen werden verbessert werden, wenn dem Entwickler aufgezeigt wird, wo eventuell noch Schwachstellen vorhanden sind. Dies ist insbesondere im weiteren Verlauf wichtig, damit die Software später im Betrieb keine Probleme verursacht, die verhindert hätten werden können durch eine eingehende Beschäftigung mit der Qualität des Codes. Vor allem die Reaktionszeiten der Anwendung sind für eine spätere Verwendung im Realbetrieb wichtig, um abschätzen zu können, ob auch eine hohe Zugriffszahl von Anwendern für das Programm tragbar ist. Zwar können bei einem Test kaum Szenarien einer tatsächlichen Nutzung der Anwendung simuliert werden, jedoch bieten sie die Möglichkeit einen Eindruck zu erhalten wie sich dieser Bereich später wahrscheinlich verhalten wird.

Ein wichtiger Teil der Softwareentwicklung stellt das Testen eben dieser dar. Viele Programmierer unterschätzen diesen Punkt und gefährden so das erfolgreiche Abschließen eines Projekts. Wird eine Anwendung nicht zeitnah und möglichst von Anfang an getestet, so kann es passieren, dass ein Code schon früh von einem Fehler befallen ist und dieser später nicht mehr ohne weiteres beseitigt werden kann. Nach jeder größeren Änderung bzw. nach dem Einbau wichtiger Funktionen sollte der jeweilige Code umgehend auf seine Funktionalität hin überprüft werden. Weiterhin sollte ebenfalls nicht darauf verzichtet werden die unterschiedlichen Einzelteile der Anwendung zusammenzusetzen, um die Verbindung zwischen diesen zu testen. Insbesondere wenn es sich bei der Anwendung um ein Projekt eines Unternehmens handelt, können durch das nicht Beachten regelmäßiger Tests hohe Kosten und Verzögerungen bei der Endabgabe entstehen, was wiederum dem Image schadet. Deshalb sollten die Mitarbeiter dazu angehalten werden in bestimmten Abständen Tests durchzuführen und diese zu dokumentieren. Die Ergebnisse dieser Tests sollten an alle Beteiligten, die in das jeweilige Projekt involviert sind, weitergeleitet werden, damit

diese einen Überblick über die Funktionalität der jeweiligen Aufgabe des anderen haben.

2.2 Architekturarten

Nachdem auf den vorherigen Seiten die Anforderungen an Software im allgemeinen erläutert wurde, soll im Anschluss nun ein Blick auf die verschiedenen Möglichkeiten der Softwareentwicklung hinsichtlich der Verwendung von Servern betrachtet werden. Dazu wurden die Bereiche Server/Client, Web-Architektur und SOA ausgewählt, die nun kurz erläutert werden.

2.2.1 Server/Client

Diese Variante der Programmierung besteht im wesentlichen daraus, dass die Komponenten Servern und Clients bestehen. Dabei können jeweils mehrere Komponenten in einem System verbaut sein. Die Funktionalität kann durch das gegenseitige Bearbeiten von Anfragen und Antworten beschrieben werden. Wird in einer Anwendung durch einen Benutzer eine Aktion ausgelöst, so sendet der Client dabei eine Anfrage an den Server, dieser bearbeitet den Auftrag und sendet das Ergebnis zurück an den Client, der dann dem Anwender entweder die gewünschte Antwort anzeigt oder die Aktion, die aktiviert wurde, auslöst.
Die Anzahl der Server ist nicht auf einen limitiert, so dass die Performance der Anwendung nicht beschränkt ist. Ein Server kann seinerseits eine gestellte Anfrage eines Clients an einen anderen Server weiterleiten. Der weiterleitende Server fungiert dann selber als Client.

Für die Komponente Client können verschiedene Merkmale aufgezeigt werden:

→ Ein Server wird kurzzeitig zum Client, wenn dieser einen Dienst von einem anderen Server anfordert. Darüber hinaus kann das Programm andere Aufgaben lokal ausführen.
→ Der Client leitet den Kontakt mit einem Server aktiv ein.
→ Der Client kann im Laufe einer Sitzung auf mehrere Server zugreifen, kontaktiert aber aktiv nur jeweils einen Server.

Auch für Server können bestimmte Merkmale festgehalten werden:

→ Der Server ist ein Programm, das einen bestimmten Dienst bereitstellt. Er kann in der Regel gleichzeitig mehrere Clients bedienen.
→ Häufig werden Server automatisch beim Hochfahren des Rechners gestartet und beim Herunterfahren beendet.
→ Der Server wartet passiv auf die Verbindungsaufnahme durch einen Client.
→ Da Server eigenständige Programme sind, können mehrere Server unabhängig voneinander auf einem einzigen Rechner als Trägersystem laufen.

Zwischen Server und Client kann die Kommunikation auf zwei verschiedene Arten geschehen. Im Fall der synchronen Kommunikation wartet der Client auf eine Antwort vom Server und erst wenn er diese erhalten hat wendet er sich anderen Aufgaben zu. In der Zwischenzeit macht der Client zwangsweise eine Pause. Bei der asynchronen Kommunikation hingegen wartet der Client nicht auf die Antwort des Servers, sondern befasst sich umgehend mit anderen Aufgaben. Bei einer Rückmeldung des Servers werden dann Routinen für bestimmte Ereignisse beim Client ausgelöst. Auch wenn die asynchrone Variante durch den Wegfall der Wartezeit für den Client zunächst von Vorteil erscheint, so darf nicht vergessen werden, dass die asynchronen Abläufe mehr Aufmerksamkeit bedürfen und es dort ein erhöhtes Fehlerrisiko durch die Anzahl der

Prozesse, die zeitgleich ausgeführt werden, gibt.

Tier-Architektur

Ein Server/Client System kann aus mehreren Ebenen bestehen. Ein klassisches System besteht aus 2 Ebenen und wird 2-Tier-Architektur genannt. Es besteht lediglich aus Server und Client. Die 3-Tier-Archiktektur besteht hingegen aus einer weiteren Komponente, die zum Beispiel aus einem Datenbankserver besteht.
Bei mehreren Ebenen wird auch von Multi-Tier-Architekturen gesprochen.

Es existieren verschiedene Client-Arten. Diese sollen nachfolgend kurz aufgezeigt und erläutert werden.

Bezeichnung	**Beschreibung**
Fat Client	Der Aufbau weist eine recht gleichmäßige Verteilung der Rechenleistung auf Server und Client. Dadurch wird der Server entlastet und kann sich auf die Bearbeitung wichtiger Aufgaben konzentrieren.(KLE11, S.209)
Rich Client	X x
Thin Client	Bei diesem Aufbau ist der Server die Hauptkomponente. Die Clients stehen nur für die Ein- und Ausgabe zur Verfügung.So werden diese nicht durch aufwändige Berechnungen belastet. Allerdings ergibt sich daraus zeitgleich auch das Risiko einer Überbelastung des Servers.(KLE11, S.209)
Ultra-Thin Client	X x

Die folgende Grafik zeigt den Aufbau eines Thin und eines Fat Client-Server Ansatz:

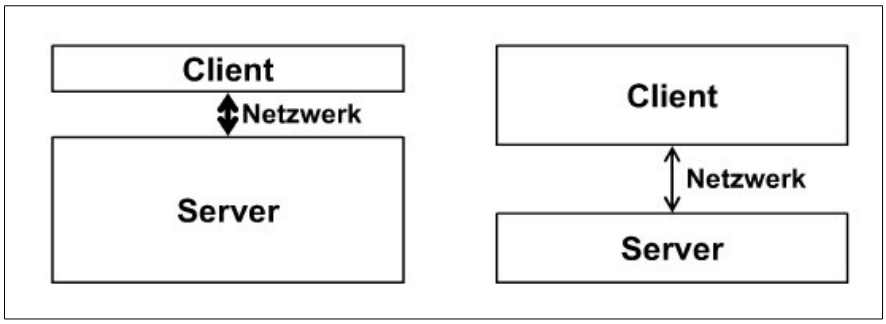


Abb. 1: Aufbau eines Thin und Fat Client-Server Aufbaus (KLE11, S. 209)

Neben den unterschiedlichen Arten gibt es auch verschiedene Zustände, die ein Client in einem System während der Ausführung annehmen kann. Dabei sind folgende Zustände möglich:

→
→
→
→

Es sind weiterhin verschiedene Aufbauten von Servern zu unterscheiden. Bei einem parallelen Server werden die eintreffenden Aufträge nicht ihrer Reihenfolge nach bearbeitet. Stattdessen wird für jeden Vorgang ein neuer Prozess gestartet.

Innerhalb eines Client/Server Systems werden die Arbeitsabläufe in unterschiedliche Prozesse aufgeteilt, die dann von der jeweiligen Komponente verarbeitet werden. Dabei ist jedoch eine Unterart von Prozessen zu benennen, die als Threads bezeichnet werden. Um diese zu erhalten wird ein Prozess in kleine Einzelteile zerlegt. Dies bietet den Vorteil, dass die kleinen Threads schneller verarbeitet werden können.

Serverarten:

→ Shared Server: Ein Prozess wird für mehrere Services zur Verfügung gestellt.

→ Unshared Server: Jeder Service liegt in einem anderen Server.

→ Per Request Server: Bei jeder Anfrage wird ein neuer Server gestartet.
→ Persistent Server: Er muss nicht aktiviert werden und Anfragen werden sofort an den Server weitergeleitet.

Die Trennung von Server und Client kann Probleme erzeugen. Diese sind:

→ Remote-Kommunikation zwischen Client und Server bedeutet oft höherer Implementierungsaufwand
→ Netzwerke zwischen Clients und Servern können mit zeitverzögerter Übertragung zu kämpfen haben (Latenz)
→ Auf Client und Server können unterschiedliche Betriebssysteme, Laufzeitumgebungen oder Implementierungstechnologien eingesetzt werden
→ Fachliche Logik auf Client oder Server implementieren
→ Synchronisation von Aktionen mehrerer paralleler Benutzer

Vorteile

Mit einer Server/Client Anwendung kann eine Benutzersitzung verfolgt werden. Dies resultiert aus der Tatsache, dass pro Client eine Verbindung verwaltet wird. So können auftretende Probleme schneller identifiziert und beseitigt werden.

Da ebenfalls die Anzahl der Benutzer, die maximal Zugriff auf das System haben, bereits von Anfang an bekannt ist, kann ein Entwurf besser erstellt werden. Es ist somit leichter zu planen, was für die jeweilige Software eingeplant werden muss und verhindert so mögliche Fehler in der Vorbereitung.

Ebenfalls ist eine Reduzierung der Systemkomplexität ein Effekt der Server/Client Methode. Die Anwendung ist deshalb übersichtlicher und leichter verständlich für Entwickler.

Weiterhin ist diese Art der Umsetzung für große Netze gut einsetzbar. Dies bietet vor allem für Unternehmen Vorteile, die so auch die Möglichkeit

haben diese Architektur anzuwenden.

Die Tatsache, dass Server/Client Anwendungen ein sehr geringes Risiko eines Ausfalls aufweisen.

Eine Anwendung dieser Art ist flexibel im Einsatz und bietet verschiedene Anpassungsmöglichkeiten an die jeweils benötigte Situation.

Das System bietet die Möglichkeit jederzeit weitere Clients einzubauen. Dazu muss das bisherige System nicht zerstört oder abgeschaltet werden. Im Gegensatz zu anderen Varianten werden hier Ausfallzeiten vermieden. Dabei gibt es keine festgesetzte Anzahl an Clients, die eingebaut werden können, was ein hohes Maß an Flexibilität bedeutet.

Nachteile

Server/Client Systeme sind nur schlecht skalierbar. Dieser Nachteil ist für eine Anwendung nicht zu unterschätzen. Deshalb sollte genau darauf geachtet werden, welchen Stellenwert die Skalierung in der jeweiligen Anwendung hat.

Bei einer reinen Server/Client Software ist es nicht möglich das Internet zu nutzen. Dies bedeutet, dass nur Programme, die lokal auf einem Rechner laufen auf diese Möglichkeit zurückgreifen sollten.

Wird die Plattform, auf der die Anwendung läuft, im Laufe der Zeit gewechselt, dann ist dies nicht ohne weiteres möglich. Dazu muss eine komplette Neuprogrammierung vorgenommen werden, welche viel Zeit in Anspruch nimmt. Daher sollte bereits vorher darüber nachgedacht werden, ob die Software dauerhaft auf der gewählten Plattform verfügbar sein soll.
Auch ist die Verteilung der Software Clientseitig aufwändiger, was eine Verzögerung der Entwicklungsphase gegenüber anderen Möglichkeiten bedeutet.

Da für eine Umsetzung zusätzliche Hardware benötigt, muss mit höheren Kosten gerechnet werden im Vergleich zu anderen Möglichkeiten was die Umsetzung angeht.

Da der Server sich mitunter als ein schwaches Element dieser Umsetzungsart zeigt, ist sie nicht für jede Art von Unternehmen geeignet. Wird dort mit sensiblen Daten gearbeitet, so muss das Sicherheitsrisiko einkalkuliert werden.

Werden weitere Server in das System eingebaut, so ist nicht vorherzusagen wie sich dies auf das bisherige System auswirkt. Dadurch entsteht ein unkalkulierbares Risiko, welches nicht zu unterschätzen ist.

2.2.2 Web-Architektur

Bei den Web-Architekturen werden Web-Techniken eingesetzt bei der Verteilung der unterschiedlichen Schichten auf die Clients und die Server. Dabei befinden sich jeweils Teile des GUI's auf verschiedenen Clients, welche im Browser ausgeführt werden. Im Gegensatz zu der Client/Server Architektur werden hier die Anfragen nicht vom Client an den Server, sondern an den Web-Server geschickt. Dieser beantwortet die Anfragen, wie sonst der Server. Weiterhin ist bei Anwendungen im Web möglich eine unbegrenzte Anzahl an Anwendern zu verwalten. Dies ist bei einer Software auf einem Rechner nicht ohne weiteres möglich, da sich die Kapazitäten anders verteilen. Ebenfalls gibt es im Web keine begrenzte Anzahl an potentiellen Nutzern, während auf einem Rechner nur die Nutzer vorhanden sind, die Zugang zum Rechner und der Software haben. Ein weiterer Unterschied zum Client/Server System ist der Zugriff auf die Laufzeitumgebung, die bei der Web-Architektur für die Entwickler nicht möglich ist.

Vorteile

Ein großer Vorteil ist, dass bei der Web-Architektur der Einsatz im Internet und Intranet ohne Architekturwechsel möglich ist.

Im Gegensatz zum Server/Client Verfahren ist hier eine gute Skalierbarkeit als Vorteil anzusehen. Für Anwendungen, die darauf angewiesen sind ist dieses Verfahren positiv zu bewerten.

Die Web-Architektur erlaubt eine hohe Benutzeranzahl

Nachteile

Ein großer Nachteil ist der Engpass, der beim Server zu Stande kommt.
Weiterhin gibt es schlechte Antwortzeiten, was gerade bei Anwendungen im Internet keine gute Basis darstellt. Passiert dies zum Beispiel bei einem Onlineshop, so kann dies eine Kundenabwanderung zur Konkurrenz zur Folge haben, wenn die Antwortzeiten insgesamt zu lange dauern.

Ein Problem, dass häufig im Zusammenhang mit Web-Architekturen auftritt, ist das Sperren von Plug-ins. Einige Browser sind so konfiguriert, dass sie einige Plug-ins sperren und die Anwendung somit nicht aufrufbar und verwendbar ist. Dies ist im Zeitalter der starken Konkurrenz im Web ein nicht zu vernachlässigendes Kriterium, dass beachtet werden sollte.

Ebenfalls ist es nur bedingt möglich eine gute Benutzeroberfläche zu erstellen, da es dort Einschränkungen gibt, die es Entwicklern schwer macht das umzusetzen, was laut Planung angedacht und gewünscht war.

2.2.3 SOA (Serviceorientierte Architekturen)

Bei den serviceorientierten Architekturen handelt es sich um eine Zusammenfassung zusammengehörender Funktionalitäten. Durch die Kombination mehrerer Services können komplexe Anwendungen realisiert und verwaltet werden. Dabei ist es das Ziel dem Nutzer die Möglichkeit zu geben große Funktionssammlungen zusammenzufassen um ad-hoc Anwendungen aus fast vollständig vorhandenen Services zu erstellen

Vorteile

Ein großer Vorteil von SOA ist es, dass verschiedene Anwendungen integriert werden können. So kann in einem Unternehmen leichter eine Verbindung zwischen verschiedenen Programmen hergestellt werden.

Weiterhin können auch externe Services integriert werden. Ein Unternehmen kann von externen Anbietern Software einbinden ohne dabei auf Flexibilität und Zuverlässigkeit verzichten zu müssen. Dies bietet eine schnellere Möglichkeit die benötigten Programme zu beschaffen und zu verwalten.

Ein wichtiger Punkt aus unternehmerischer Sicht ist die Kosteneinsparung, die sich aus der Verwendung von SOA ergibt. Dadurch, dass externe Anwendungen einfach und schnell integriert werden können, muss ein Unternehmen keine eigenen Entwicklungen mehr vornehmen.

Nachteile

Auch wenn SOA prinzipiell für ein Unternehmen von Vorteil ist, so ist der Anfangsaufwand, der sich dort ergibt, als recht hoch zu bewerten. Erst nach einiger Zeit sind die Vorteile spürbar und dies sollte in Betracht gezogen werden bei der Überlegung der Verwendung von SOA.

Ein weiterer Nachteil sind die komplexen Abläufe beim SOA. Diese führen dazu, dass beim Auftreten eines Fehlers dieser nur durch eine recht aufwändige Suche gefunden werden kann. Dies kostet Zeit und kann zu massiveren Ausfällen führen, die über einen längeren Zeitraum vorhanden sind. Für ein Unternehmen ist dies als Katastrophe anzusehen, da so Verluste von Kunden zu befürchten sind. Auch sind Tests der Software ebenfalls aufwändig und somit nur selten durchführbar, was ein Risiko bezüglich der Entstehung von Fehlern aufweist

Da es recht aufwändig ist die Services zu entkoppeln, ist SOA ebenfalls nicht für jede Art von Unternehmen anzuraten.

2.3 Schichtenmodell

Das Schichtenmodell stellt ein häufig eingesetztes Architekturmuster dar. Dabei werden die Daten, die vom Client angefragt werden, durch unterschiedliche Schichten geleitet, bis sie an ihr Ziel gelangen. Jede Schicht stellt der nächsten Schicht unter sich bestimmte Dienste zur Verfügung, damit diese die Verarbeitung der Daten erfolgreich weiterführen kann. Die Struktur der einzelnen Schichten ist nach außen hin nicht sichtbar und somit geschützt vor fremden Blicken. Das Schichtenmodell ist besonders gut geeignet, um große Systeme übersichtlich gestalten zu können. Dadurch, dass die einzelnen Aufgaben auf die verschiedenen Schichten verteilt werden, ist die Belastung der einzelnen Schichten nicht so hoch und Fehler sind nur auf den jeweiligen Bereich beschränkt. Die Komponenten können beliebig untereinander aufeinander zugreifen und bieten so hohe Flexibilität.

Die möglichen Komponenten eines Schichtenmodells bestehen aus den folgenden Punkten:

- → Benutzeroberfläche
- → GUI
- → Anwendung / Verarbeitung
- → Datenmanangement
- → persistente Datenspeicherung / Daten

Vorteile

Der größte Vorteil, der sich aus dem Schichtenmodell ergibt, ist eine gute Strukturierung. Diese ist für jede Anwendung wichtig, um einen guten Überblick über die Komponenten einer Software behalten zu können und in einem Fehlerfall schnell reagieren zu können. Weiterhin können sich so bei einem Mitarbeiterwechsel auch diese schnell in das Programm einarbeiten.

Diese gute Strukturierung wird durch die virtuellen Maschinen erreicht.

Weiterhin sind die einzelnen Schichten voneinander unabhängig. Somit muss eine Anwendung nicht auf andere Schichten warten, um an das eigentliche Ziel gelangen zu können.
Durch die nicht nicht vorhandenen Einschränkungen des Entwicklers existiert beim Schichtenmodell eine hohe Flexibilität was die Umsetzung betrifft.

Ein weiterer Vorteil bietet die Austauschbarkeit von Schichten, wenn diese die gleichen Dienste aufweisen. Ist eine Schicht durch einen Fehler nicht mehr verwendbar, so kann sie schnell und einfach durch eine andere ersetzt werden, ohne dass ein großer Aufwand betrieben werden muss. So ist gewährleistet, dass die Anwendung möglichst keine Ausfallzeiten aufweist.

Nachteile

Der größte Nachteil ist, dass alle Daten verschiedene Schichten durchlaufen müssen. Dies hat zur Folge, dass die Daten recht lange unterwegs sind, bevor sie an ihr Ziel kommen.

Auch wenn es positiv zu bewerten ist, dass die Schichten voneinander unabhängig sind, so geht damit einher ein gewisses Chaos innerhalb der jeweiligen Schichten.

2.4 Middleware

Kommt noch

2.5 Zusammenfassung des Kapitels

In diesem Kapitel wurden die bisherigen Methoden, die zum Aufbau von Anwendungen und Programmen seit vielen Jahren verwendet werden,

aufgezeigt und kurz erläutert. Dabei fällt auf, dass die jeweiligen Möglichkeiten sich durch die Verwendung mehrerer Komponenten auszeichnen, die jeweils durchlaufen werden müssen, um ein Ergebnis auf eine Anfrage zu erhalten. Ein Ausfall einer Komponente kann unter Umständen zu einem Totalausfall des gesamten Systems führen und erzeugt somit Kosten und führt zu einem Vertrauensverlust bei Kunden. Durch die Vor- und Nachteile der jeweiligen Methoden wurde gezeigt, wie sich die bisherigen Möglichkeiten der Softwareentwicklung in der Praxis bewähren. Dies ist für die Problemstellung der Ausarbeitung als wichtig anzusehen, da so die Unterschiede zum Bereich der Serverless Programmierung deutlich werden.

3. Serverless Programmierung

Die vorliegende Ausarbeitung befasst sich vorranging mit der Thematik der Serverless Programmierung, welche im Zuge dieses Kapitels aufgegriffen und erläutert werden soll. Dabei soll zunächst auf die Definition eingegangen werden, um den Begriff der Serverless Programmierung zu erläutern und einen Einstieg in den Bereich zu geben. Anschließend wird das Augenmerk auf die Funktionsweise gelegt, bei der der Aufbau einer Anwendung mit Serverless Programmierung und dessen Möglichkeiten näher beleuchtet werden sollen. Da der Bereich in der Softwareentwicklung als relativ neu angesehen werden kann ist ein Einblick in die derzeitige Etablierung am Markt als nächsten Unterpunkt interessant, um sich einen Überblick verschaffen zu können. Abschließend sollen die Einsatzbereiche für Serverless Programmierung aufgezeigt werden. Dabei wird auf bereits aktuell verwendete Möglichkeiten ebenso eingegangen werden wir auf in Zulunft denkbare Verwendungen.

3.1 Definition

Eine eindeutige Definition was Serverless genau bedeutet ist derzeit nicht existent. Vielmehr ist es ein Begriff für eine Ansammlung von Gegebenheiten und Elementen. Allgemein wird aber unter Serverless Programmierung eine Möglichkeit der Softwareentwicklung verstanden, bei der es nicht mehr von Nöten ist einen Server anzumieten. Stattdessen werden Kapazitäten bei einem Drittanbieter eingekauft, der sehr große Server besitzt und diesen Platz an Kunden vermietet. Was auf den ersten Blick vielleicht nicht spektakulär aussieht, bietet den Entwicklern von Software einige Erleichterungen. Dennoch sollte auch Serverless Programmierung genauso wie die bisher gängigen Möglichkeiten nicht wahllos verwendet werden. Was auf der einen Seite Vorteile für bestimmte Projekte aufweist, ist für andere Umsetzungen nicht empfehlenswert. Deshalb ist auch hier eine vorherige Auseinandersetzung mit den genauen Anforderungen an die zu erstellende Software zwingend notwendig. Weiterhin sollte bedacht werden, dass sich die Plattformen der Drittanbieter voneinander abgrenzen und auch diese Entscheidung am Anfang sehr genau getroffen werden sollte, da ein späterer Wechsel der Plattform große Probleme nach sich zieht.

Die Art der Berechnung der Kosten für die Serverkapazität verändert sich hierbei grundlegend. Jeder Kunde zahlt dabei einen komplett anderen Preis, als die anderen Kunden, da jeder nur das zahlt, was er auch tatsächlich an Serverleistung verwendet.
Die meisten der webbasierten Anwendungen haben ein back end und arbeiten mit Servern. Dies bedeutet diverse front-ends, damit der Anwender diese verwenden kann. Somit wird vorausgesetzt, dass eine Menge Leistung vorhanden sein muss um die hohe Anzahl der unterschiedlichen Prozesse bewerkstelligen zu können. Dies ist bei Serverless Programmierung nicht

relevant zu beachten, da so viel Leistung wie nötig erhalten werden kann.
Die Daten werden durch mehrere Schichten des Systems transportiert, bevor sie in einer Datenbank landen oder von dort geladen werden. Dass bedeutet es müssen diverse Antworten generiert werden, welche wiederum zum Client geschickt werden. Für all diese Prozesse werden Server benötigt.

Die Entwickler verlieren beim Serverless Programmierung die Verantwortung für den oder die Server. Da ein Drittanbieter die Serverkapazität zur Verfügung stellt, wartet dieser auch seine Server. Demnach fällt dieser Aufgabenbereich hier weg und der Server ist immer auf dem aktuellsten Stand. Weiterhin haben die aufgezeigten Möglichkeiten aus dem vorigen Kapitel gezeigt, dass es Probleme bei der Skalierbarkeit der Software geben kann. Dies ist beim Serverless Programmierung nicht mehr vorhanden, da die Skalierung automatisch geschieht. Wird mehr Serverleistung benötigt, so gibt der Drittanbieter automatisch mehr frei.
Geeignet ist Serverless Programmierung besonders für Nanoservices.

Das Ziel von Serverless Programmierung lässt sich abschließend wie folgt zusammenfassen: Die Entwickler sollen sich möglichst ausschließlich auf die Entwicklung des Codes und somit der Anwendung widmen und sich nicht mit anderen Bereichen auseinandersetzen, die von ihrem eigentlichen Ziel ablenken und das Risiko bieten eine schlechte Qualität oder Fehler zu erhalten.

3.2 Funktionsweise

Die Funktionsweise beim Serverless Programmierung besteht aus der Definition von Events, welche die unterschiedlichen Funktionen auslösen. Insgesamt kann festgehalten werden, dass es weniger Komponenten als bei den bisherigen Möglichkeiten gibt und sich daraus eine Vereinfachung der Organisation ergibt.

Weiterhin wird die Möglichkeit geboten Probleme, die bei verschiedenen Anwendungen auftreten, zeitgleich zu lösen durch die Funktion des Adressierens an verschiedene Stellen.

3.3 Etablierung am Markt

Obwohl das Prinzip des Serverless Programmierung kein gerade erst neu erfundene Variante der Softwareentwicklung darstellt, ist es doch bisher nahezu unbekannt und wird nur von verhältnismäßig wenigen Unternehmen verwendet. Wer versucht zu dem Thema Informationen zu erhalten, der wird auf die Problematik stoßen, dass es nur recht wenig Literatur dazu gibt. Bücher zum Beispiel sind im Prinzip nicht existent, werden allerdings, das sei hier angemerkt, bereits verfasst und bearbeitet und sollen im Jahr 2017 Einzug in den Büchermarkt erhalten. Dem Interessierten und Neugierigem bleiben zahlreiche Artikel, die sich alle mit der Thematik des Serverless befassen und einen Einblick in das gibt, was derzeit im Kreise der Entwickler auf dem Vormarsch ist. Menschen, die nicht der englischen Sprache mächtig sind, dürften es jedoch auch dort schwer haben sich Informationen über die Thematik zu besorgen, da nahezu fast alle Texte nur in Englisch zu finden sind. Auch dies ist der bisher eher nur leichten Verbreitung geschuldet und es wird noch ein Weile dauern, bis die Welle der Begeisterung die Skeptiker der alten Schule erreicht hat.

Weiterhin ist es ein Problem in der Welt der Softwareentwicklung neue Systeme und Möglichkeiten integrieren zu wollen. Denn die Entwickler halten sich zumeist an altbekanntes und zeigen nur wage Bereitschaft dazu sich neuen Möglichkeiten zu öffnen. Wenn ein Entwicklerteam die Auswahl hat Möglichkeit A zu wählen, zu der es eine Unmenge an Literatur und Foren gibt, die bei der Erarbeitung durch zahlreiche Beispiele oder der Beseitigung von Problemen mit hilfreichen Informationen und Tipps helfen,

dann ist es wenig verwunderlich, dass Möglichkeit B, die dies nicht aufwarten kann, dabei verliert.

Trotz aller Probleme, die bei der Etablierung am Markt auftreten, existieren bereits einige Anbieter, die die Möglichkeit einer Serverless Programmierung bieten. Diese werden zu meist in bereits existente Plattformen integriert und bieten so die Möglichkeit auf die anderen zur Verfügung stehenden Dienste zugreifen zu können. Somit kann ein Programmierer bzw. ein Unternehmen eine komplette Anwendung durch die Verwendung einer dieser Plattformen realisieren. Einige der Angebote befinden sich jedoch noch in der Testphase und eignen sich daher nur bedingt für eine kommerzielle Umsetzung, da damit zu rechnen ist, dass sich an der Umsetzung der Möglichkeit der Serverless Programmierung im Zuge der Testphase noch einige Veränderungen ergeben werden. Somit kann ein kompletter Neuanfang von Nöten sein oder bei zu großen Änderungen auch ein Umzug auf eine andere Plattform. Dennoch stellen die derzeit vorhandenen Plattformen eine Möglichkeit dar sich mit der Möglichkeit der Serverless Programmierung auseinanderzusetzen und sich damit beschäftigen zu können, um einen Einblick in die Funktionsweise zu erhalten. Durch diesen Vorgang kann es später leichter sein sich intensiver mit der Erstellung einer Anwendung durch eine Serverless Programmierung zu beschäftigen, da ein gewisses Grundverständnis bereits vorhanden ist.

3.4 Einsatzorte

Um zu zeigen, wie vielfältig Serverless Programmierung eingesetzt werden kann, sollen in diesem Kapitel die Einsatzorte dieser Methode näher betrachtet werden.

Zuerst lässt sich allgemein festhalten, dass sich die Serverless

Programmierung für jegliche Größe von Projekten eignet. Es ist egal, ob es sich nur um eine kleine Anwendung eines Anfängers der Programmierung handelt oder um ein großes Unternehmen, welches sehr viele Zugriffe auf die Anwendung vorweisen kann. Durch die schnellere Verarbeitung der Anfragen kommt es auch dort zu keinen Ausfallzeiten.

Für den Aufbau von Websiten eignet sich ein Serverless Aufbau, da dort viele Anwender zeitgleich auf die Anwendung zugreifen. Gerade für einen Online-Shop, der verschiedene Aktionen auf seiner Seite in Form von einer Suche, den Artikelseiten, den Kundenkonten und dem Bestellvorgang realisieren muss, ist eine geringe Latenz der Anwendung sehr wichtig.

Der Markt für mobile Anwendungen in Form von Apps erfährt seit einigen Jahren einen erheblichen Wachstum, was sich durch die stetige Verbreitung von Smartphones und Tablets begründet. Deshalb zeigen immer mehr Unternehmen Interesse an diesem Markt. Dabei müssen die Apps den Anforderungen der Anwender gerecht werden, die insbesondere Wert auf eine schnelle Reaktionszeit innerhalb dieser legen. Deshalb eignet sich die Serverless Programmierung gut für solch eine Umsetzung.

Eine Real-Time Analyse, die darauf angewiesen ist, dass die jeweilige Anfrage möglichst schnell bearbeitet wird und das Ergebnis zeitnah vorliegt, ist ebenfalls ein Einsatzbereich, der sich für Serverless gut eignet, denn auch hier spielt die geringe Latenz eine große Rolle.

Auch wenn dies keine klassische Umsetzung einer Anwendung darstellt, ist die Verwendung eines Aufbaus mit serverless Programmierung um ein Backup von Daten, Datenbanken oder einer Anwendung erstellen zu können, eine weitere Möglichkeit des Einsatzes.
Die Anzahl der Fotos, die im Zeitalter des Smartphones mit eben diesem gemacht werden nimmt täglich zu. Jedoch haben diese Bilder den Nachteil, dass sie eine recht große Dateigröße haben und ein Hochladen dieser auf

eine Plattform oder das Kopieren auf ein anderes Gerät mehr Zeit benötigt. Deshalb gehen immer mehr Anwender dazu über die Dateigröße der Bilder zu reduzieren. Damit dieser Vorgang schneller umgesetzt werden kann, ist eine serverless Anwendung von Vorteil.

Für Anwendungen, die in der Cloud entwickelt werden ist die Serverless Programmierung besonders geeignet, da sie bei Anbietern solch einer Plattform auch erstellt werden kann. Ebenfalls kann die so erstellt Anwendung dann auch dort betrieben werden. Dies vereinfacht die Programmierung, da alles in der gleichen Umgebung vorgenommen werden kann.

Eine weitere Art von Anwendung findet im Internet und auch lokal als Software anklang. Sie dient dazu eine Datei in ein anderes Format umzuwandeln. Dabei kann es sich sowohl um Bilder handeln, die nicht dem gewünschten Format entsprechen oder um Textdateien, die nicht Programm übergreifend geöffnet werden können. Auch hier lohnt sich der Einsatz von serverless Programmierung, da die Konvertierung so schneller umgesetzt werden kann. Gerade für die Online-Variante ist dies ein wichtiger Faktor.

Die Erstellung von Microservices erfordert eine einfache und leicht in fremde einzubauende Umsetzung dieser, damit sie leicht zu einer gesamten Anwendung zusammengefügt werden können. Die Event-basierten Funktionen einer Serverless Programmierung bieten sich dafür gut an.
Der Bereich des Internet of Things hat in letzter Zeit immer mehr an Bedeutung gewonnen. Um dort möglichst kleine Anwendungen konzipieren zu können, die nicht über viel Datengröße verfügen. Durch serverless Programming wird dies ermöglicht.

Für Firmen, die gerade neu gegründet wurden und nun eine Erstellung einer Anwendung oder eines Online-Shops planen ist es reizvoll sich mit der Thematik der Serverless Programmierung auseinanderzusetzen, da sie nicht

in alten Standards denken. Es bietet sich daher an sich mit den aktuellen Möglichkeiten der Softwareentwicklung zu beschäftigen, um mit der Umsetzung des geplanten Projekts auf dem neusten Stand der Dinge zu sein und sich auch von anderen abheben zu können. Wenn so zudem die Performance der Anwendung verbessert werden kann, bietet es weitere Wettbewerbsvorteile, die andere nicht vorweisen können. Deshalb sind Start-Ups eine interessante Möglichkeit für einen Einsatz der Serverless Programmierung, da die Gründer und Mitarbeiter dort meistens auch mit einer anderen Art und Weise des Denkens an Umsetzungen herangehen, als ein Programmierer, der seit 20 Jahren auf die gleiche Art und Weise Anwendungen umsetzt.

Mitunter gibt es Anwendungen, die sehr viele Anfragen auf das System produzieren, ansonsten aber wenig andere Funktionen haben als dies. Durch die große Anzahl an Events, die auf diese Art und Weise ausgelöst werden, wird die Anwendung extrem belastet was die Performance angeht. Da sich die Serverless Programmierung sich unter anderem genau dieser Problematik widmet, ist sie für eine Anwendung, die sehr viele Events verarbeiten muss als sinnvoll anzusehen. Die Anwendung hat so die Möglichkeit die Verarbeitung schneller abwickeln zu können und der Anwender selber kann sich einer schnellen Bearbeitung seines Anliegens erfreuen.

Das Einsatzfeld der Erstellung von Statistiken ist ebenfalls gut geeignet für die Serverless Programmierung. Bei einer solchen Erhebung werden teilweise unzählige Zahlen ins System eingetragen, die dann verarbeitet werden müssen und dies kann mitunter eine Weile dauern. Ein Aufbau ohne Server umgeht das Durchlaufen diverser Schichten und spart deshalb Zeit, so dass die Ergebnisse eher vorliegen.

Ein weiterer Bereich, der darauf angewiesen ist, dass die Verarbeitung

schnell funktioniert, ist die Datenverarbeitung. Gerade wenn mit großen Datenbanken gearbeitet wird innerhalb einer Anwendung sind dort viele Zeilen zu durchsuchen, wenn eine Aktion bzw. eine Suche in der Datenbank vorgenommen wird. Im kommerziellen Bereich ist es unbedingt nötig, dass der Kunde seine Daten schnell vorliegen hat. Hat die Konkurrenz eine schnellere Lösung anzubieten, kann dies einen finanziellen Schaden verursachen.

Anwendungen bestehen immer aus einer Reihe von Zeichen, die kodiert werden müssen. Handelt es sich jedoch dabei um sehr viele Zeichen, so kann die Kodierung aufwendig werden. Auch hier kann die Performance dieser Anwendungen durch eine serverless Umsetzung gesteigert werden.

Verallgemeinert lässt sich festhalten, dass jede Anwendung, bei der eine große Menge an Daten verwaltet und verarbeitet werden muss, sich für den Einsatz der Serverless Programmierung eignet. Sie ermöglicht es schneller die gewünschten Ergebnisse zu erzielen und durch den einfacheren Aufbau wird die Performance deutlich erhöht. Im Bereich der Online-Anwendungen gibt es immer mehr Anbieter und jeder davon muss eine Möglichkeit finden sich von der Konkurrenz abzuheben. Das Arbeiten ohne Server stellt daher eine neue Möglichkeit dar, die bisher wenig verbreitet ist und somit einen Vorteil gegenüber anderen Anbietern aufzeigt.

→ contemporary applications that rely on real-time streaming
→ sensor data aggregation
→ create online bots (IoT)

3.5 Zusammenfassung des Kapitels

Auf den Seiten des Kapitels 3 wurde das Themenfeld der Serverless Programmierung näher betrachtet. Der Verzicht auf einen Server, der angemietet werden muss, wirkt sich vor allem in der effektiveren Kostenberechnung nach tatsächlich genutzen Ressourcen aus. Die automatische Skalierbarkeit, die je nach Bedarf und Zgruffszahlen auf die Anwendung die Leistung der Anwendung selbstständig regelt, ist ein Gewinn für Unternehmen, die schwankende Anwenderzahlen haben und so Kosten sparen können. Die Erläuterung hat weiterhin einen kleinen Einblick gegeben welche Vorteile sich aus der Serverless Programmierung ergeben. Dies ist für das Kapitel der Vor- und Nachteile interessant, um die dortigen Argumente besser nachvollziehen zu können. Nachdem nun die Randbedingungen dargestellt wurden, sollen im folgenden Kapitel Möglichkeiten aufgezeigt werden mit welchen Mitteln diese umgesetzt werden können.

4. Vorstellung der Möglichkeiten Serverless Programmierung

Vorige Kapitel hat den Sinn der Serverless Programmierung und dessen Funktionsweise sowie der bisherigen Etablierung aufgezeigt, doch wurde bisher die tatsächliche Umsetzung nicht beachtet. Deshalb sollen in diesem Kapitel nun Möglichkeiten in Form von vier Plattformen aufgezeigt werden, die eine Umsetzung für Serverless Programmierung anbieten. Hier soll ein Überblick über die derzeit verfügbaren Funktionen und den Aufbau der Plattformen gegeben werden.

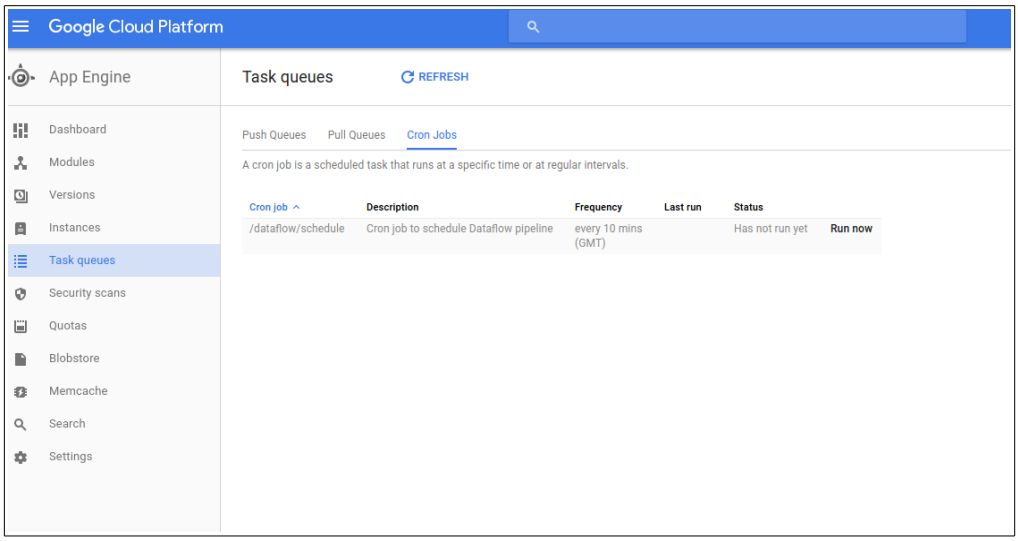
4.1 Google Cloud Functions

Google hat seine eigene Möglichkeit für Serverless Programmierung erstellt, die im Jahre 20xy vorgestellt wurde und sich derzeit in der Alpha-Phase befindet. Sie ist Teil der Google Cloud Platform und wird in Form eines Services dort zur Verfügung gestellt. Um Zugang zu der Plattform erhalten zu können, müssen die Anwender dafür eine Gebühr bezahlen. Anfangs hat jeder die Möglichkeit sich 2 Monate lang durch eine kostenlose Testzeit einen Überblick über die Cloud zu verschaffen. Für die Alpha-Version des Serverless Bereichs muss eine gesonderte Anmeldung erfolgen, die nur mit einem bereits vorhandenen Account auf der Plattform möglich ist.
Die Funktionen werden dabei derzeit über eine Konsole eingegeben, da derzeit noch keine ausgereifte Version auf der Plattform selber vorhanden ist, die es ermöglicht über eine Maske Funktionen zu erstellen.
Durch die derzeitige Alpha-Phase findet die Plattform im Gegensatz zu anderen Konkurrenzprodukten nur wenig Aufmerksamkeit. Auch Tutorials beziehen sich eher auf andere Anbieter, weshalb Google Cloud Functions es schwer haben dürfte sich in dem Bereich der Serverless Programmierung zu etablieren bzw. sich durchzusetzen.
Die Nutzer müssen nicht zuvor ein passendes System einrichten und keine Hardware oder virtuelle Maschine reservieren. Deshalb ist es auch gut für Einsteiger geeignet, die im Bereich der Programmierung kaum Erfahrung haben und sie mit Entwicklungsumgebungen nicht gut auskennen. So können sie schnell einen Einblick erhalten und Erfolge erreichen.

Die Funktionen werden in Java geschrieben und in Node.js ausgeführt. Jede der Funktionen lässt sich einzeln umsetzen und wird auch einzeln abgerechnet. Der Dienst der Plattform ist als Open-Source Software verfügbar. Zusätzlich sind Schnittstellen zu IBMs Computersystem Watson

vorhanden, wodurch Entwickler Hilfe bei der Auswahl der für sie geeignetsten Programmierschnittstellen erhalten können. Der Code der Plattform ist quellofen verfügbar und einsehbar auf GitHub. Entwickler können so einen Blick hinter die Kulissen werfen und sich ansehen wie die Plattform aufgebaut ist.

Die folgende Grafik zeigt den Aufbau der Google Cloud Plattform. Hier hat der Entwickler die Möglichkeit seine Dokumente zu verwalten und auszubauen.



→ Container Engine, um einfach und schnell Container zu erstellen

→ Google Cloud Functions is a lightweight, event-based, asynchronous compute solution that allows you to create small, single-purpose functions that respond to cloud events without the need to manage a server or a runtime environment

4.2 Windows Azure Functions

Windows Azure Functions ist die Möglichkeit des Serverless Programmierung, die von Windows angeboten wird. Die Plattform wurde im März 2016 vorgestellt. Die Tatsache, dass die Plattform noch sehr jung

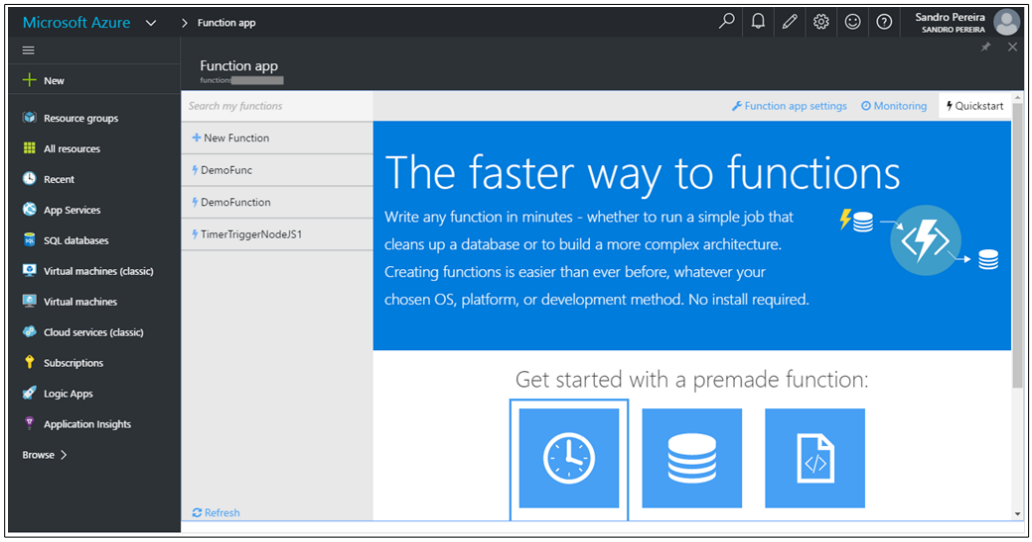
ist, ist auch bei der Literatursuche zu sehen, da sie sich erst langsam in die Riege der Serverless Anbieter integriert und bei den Entwicklern verbreitet. Der Nachteil hier ist, dass andere Plattformen, die bereits früher auf den Markt gebracht wurden bereits von Programmierern verwendet wurden und sich diese nicht in jede Plattform einarbeiten. So haben es neue Möglichkeiten des Serverless Programmierung schwer die Entwickler davon zu überzeugen sich ihrer Plattform anzunehmen.

Windows Azure Functions arbeitet mit Funktionen, die in Java oder C# geschrieben werden. Die Plattform bietet die Möglichkeit an sich Beispiele und Beispielfunktionen anzeigen zu lassen. So können Einsteiger die Funktionsweise besser nachvollziehen und erlernen. Weiterhin können sie diese verändern und dadurch den Aufbau der Funktionen besser verstehen. Die Plattform unterstützt C#, F#, Python und PHP und bietet somit eine gute Vielfalt für Entwickler. Ein Vorteil der Plattform ist, dass Dateien auf viele Wege bereitgestellt werden können z.B. durch Git, Dropbox oder OneDrive. Dies ermöglicht eine unabhängige Verwaltung der Dokumente und ist auch bei einer ortsunabhängigen Bearbeitung eines Projekts hilfreich für die Organisation der Mitarbeiter.

Eine Funktion kann mit wenigen Klicks über eine Maske erstellt werden. Jedoch werden weitere Einstellungen erst nach Erstellung der jeweiligen Funktion vorgenommen. Dazu erscheinen dann im Menü weitere Unterpunkte, die ausgewählt werden können. Allgemein können die einzelnen Services und Daten über ein Menü an der linken Seite der Plattform aufgerufen werden, welches seinerseits weitere verschachtelte Unterpunkte beinhaltet. So verliert ein Anfänger recht schnell den Überblick und wird eher verwirrt, als dass er sich dort zurecht findet, wenn er nicht gezielt weiß wonach er suchen muss.

Das folgende Bild zeigt den Aufbau der Benutzeroberfläche von

Windows Azure Functions. Hier hat der Entwickler die Möglichkeit diverse Einstellungen vorzunehmen und den Code zu erstellen.



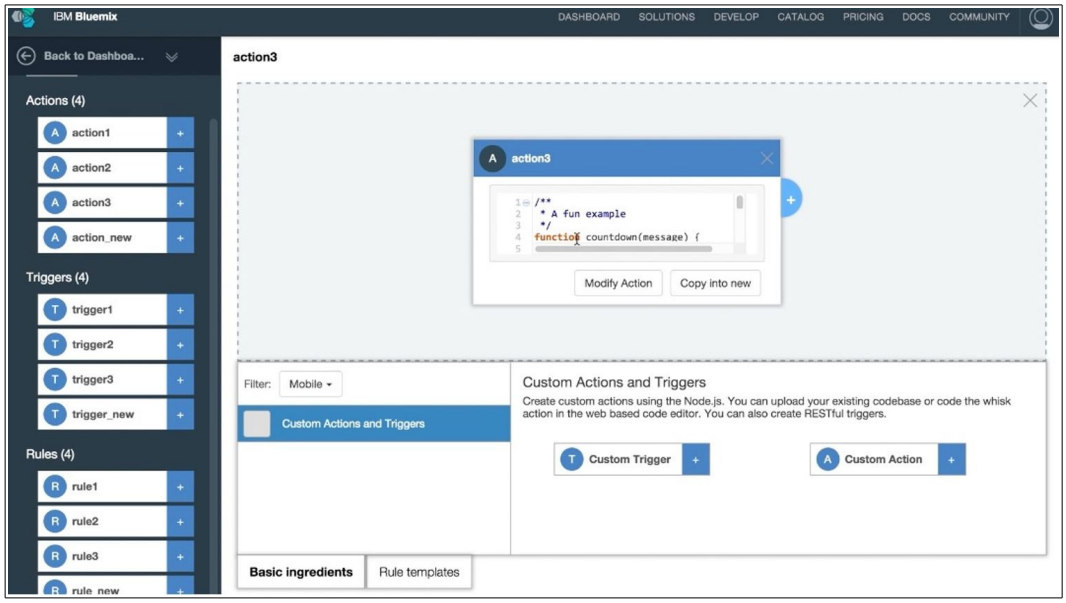
4.3 IBM Open Whisk

Auch IBM hat sich in die Reihe der Plattformen integriert, die Serverless Programmierung ermöglichen. Ebenso wie die vorherigen beiden Plattformen ist diese nicht so weit verbreitet wie AWS Lambda, findet jedoch auch immer mehr Verwender. Durch den übersichtlichen Aufbau der Verwaltung können auch Einsteiger leicht die Möglichkeiten überblicken und finden sich dort gut zurecht. IBM Open Whisk ist gut für kleine Teams geeignet und bietet eine hohe Flexibilität. Weiterhin verfügt die Plattform über eine integrierte Containerunterstützung und erleichtert so den Entwicklern ihre Arbeit. Durch die Möglichkeit einer ortsunabhängigen Bearbeitung eines Projekts mehrerer Mitarbeiter zur gleichen Zeit ist ein flexible Verteilung von Aufgaben an unterschiedlichen Standorten eines Unternehmens möglich, ohne dass diese alle am gleichen Ort sein müssen. Dies bietet die Möglichkeit die Erfahrung aus unterschiedlichen Orten miteinander zu vereinen, um das bestmögliche Produkt zu erhalten.

IBM Open Whisk bietet die Möglichkeit Microservices mit Node.js und Swift zu erstellen. Außerdem können weitere Sprachen integriert werden. Die Entwicklungsoberfläche bietet die Möglichkeit verschiedene Services, die bereits integriert sind, einzubinden und anzuwenden. Die schnelle Verkettung von Microservices garantiert ein schnelles Arbeiten seitens der Entwickler. Weiterhin ist eine bedarfsgesteuerte Ausführung des Codes in einer hoch skalierbaren Server losen Umgebung gewährleistet.

Zunächst wird jedem Anwender ein Testzeitraum von 30 Tagen gewährt, in dem er Zeit hat sich mit der Plattform auseinanderzusetzen und festzustellen, ob er diese auch weiterhin nutzen möchte. Ähnlich wie AWS bietet IBM Bluemix dabei die Möglichkeit von freien Kontingenten bei einigen der angebotenen Services, die der Anwender nutzen kann. So können Privatpersonen mit nur kleinen Projekten oder Anfänger kostengünstig arbeiten. Zudem kann durch einen integrierten Kostenrechner ermittelt werden bei welchen Services welche Kosten anfallen und wie sich diese optimieren lassen.

Die folgende Grafik zeigt den Aufbau der Oberfläche von Bluemix.



4.4 Amazon Webservices

Für die Umsetzung des Prototyps wird die Plattform AWS von Amazon verwendet, weshalb an dieser Stelle ein genauerer Blick auf eben diese Plattform erfolgen soll. So wird ein Überblick über den Aufbau und die Möglicheiten, die dort angeboten werden, geschaffen. Weiterhin soll verdeutlicht werden, wieso AWS derzeit die am weitesten verbreitete Plattform seiner Art ist und wie sich diese Tatsache auswirkt.

4.4.1 Allgemeines

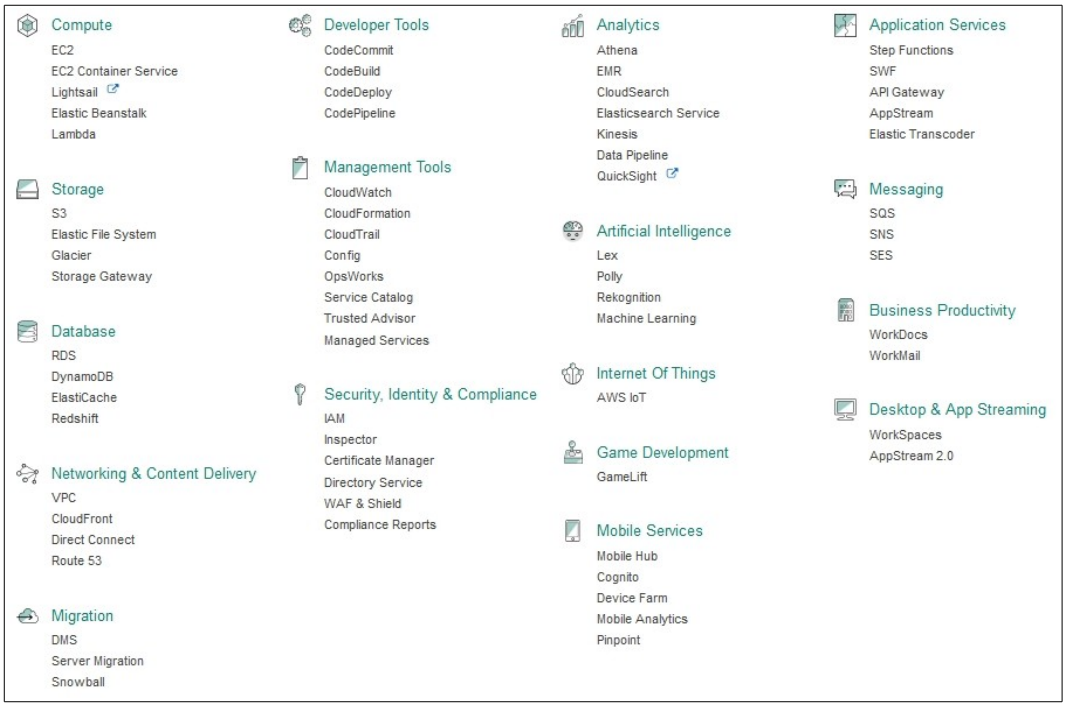
Die Plattform kann von jeglicher Art von Anwendern genutzt werden. Ein Privatnutzer, der nur kleine Anwendungen erstellen oder testen möchte ist ebenso willkommen wie das große Unternehmen, welches einiges an Serverleistung und Dienste nutzen möchte. Gerade Firmen gehen immer mehr dazu über sich an Drittanbieter solcher Plattformen zu wenden und dort ihren Anwendungen zu erstellen und zu verwalten. Dort werden viele verschiedene Dienste und Services geboten, die alle vereint sind auf einer Plattform und es so ermöglicht einen besseren Überblick über alles zu haben und einfacher einzelne Elemente miteinander verbinden zu können.
AWS selber bietet dazu unterschiedliche Möglichkeiten der Preisgestaltung. So sind Kunden, die lediglich wenig Ressourcen nutzen meistens davon befreit von monatlichen Kosten. Insbesondere wenn es sich dabei lediglich um unregelmäßige Zugriffe handelt werden monatliche freie Kontingente angeboten, die sich auf Zugriffe auf eine erstellte Anwendung oder auf genutzten Speicherplatz beziehen. Bei Unternehmen werden entweder Paketpreise angeboten oder nach exakt verbrauchten Daten abgerechnet. Dabei ist jedoch zu beachten, dass jeder Dienst eine andere Preisgestaltung hat und auch nicht überall ein freies Kontigent zur Verfügung steht. Deshalb sollte vorher genau recherchiert werden welche Dienste sinnvoll für das

eigene Projekt sind, damit es am Ende des Monats keine bösen Überraschungen gibt. Manche der angebotenen Datenbanken zum Beispiel laufen unendlich weiter und generieren somit immer Kosten, solange sie laufen. Wer sich nur zur Einarbeitung damit auseinandersetzen möchte, sollte auf solche Details achten und regelmäßig die aktuelle Rechnung ansehen, um einen Überblick zu haben und zu behalten.

Der Anwender hat die Möglichkeit auf folgende Bereiche in AWS zuzugreifen:

→ Compute
→ Developer Tools
→ Analytics
→ Application Services
→ Storage
→ Management Tools
→ Artificial Intelligence
→ Messaging
→ Database
→ Security, Identity and Compliance
→ Internet of Things
→ Business Productivity
→ Desktop und App Streaming
→ Networking und Content Delivery
→ Migration
→ Game Development
→ Mobile Services

Die folgende Grafik zeigt eine Übersicht über die jeweils zur Verfügung gestellten Services und Dienste von AWS.



Hier wird deutlich wie groß die Vielfalt ist, die AWS bietet und wieso so eine Plattform wie diese attraktiv für Programmierer und Unternehmen ist. Durch die Möglichkeit verschiedene Datenbanken zu nutzen und mit anderen Services nutzen zu können ebenso wie diverse Sicherheitseinstellungen und Spieleprogrammierung können hochwertige und aufwendige Anwendungen konzipiert werden.

Die Dienste können dabei mitunter auf verschiedene Weisen verwendet werden. An sich hat jeder Service, der aus der Übersicht ausgewählt wird, eine Maske, mit deren Hilfe eine Bedienung dessen vorgenommen werden kann. Dies ist für Anfänger gut geeignet, da sie die Felder, die ausgefüllt werden müssen sowie die Einstellungen die ausgewählt werden müssen, bereits vorgefertigt sind und nur ausgewählt werden müssen. So können sich Anwender besser an den Möglichkeiten orientieren. Jedoch gibt es auch die Möglichkeit über die AWS Command Line zu arbeiten. Diese kann für verschiedene Betriebssysteme lokal installiert werden und über die Eingabeaufforderung des Systems bedient werden. Durch die Eingabe der

Zugangsdaten zum AWS Account kann Verbindung mit diesem aufgenommen werden. Anschließend können die verschiedenen Services von AWS angesteuert und verwendet werden.

Um mit AWS arbeiten zu können muss der Interessent erst einen Account dort erstellen. Für Kunden vom Onlinehändler Amazon ist diese neue Registrierung nicht nötig, da sie sich mit diesen schon vorhandenen Anmeldedaten auch bei AWS einloggen können.

Wird ein Service das erste Mal verwendet, dann erscheint bei vielen die Auswahlmöglichkeit „getting started“. Unter dieser Option verbirgt sich eine Einführung in die Verwendung und Erstellung des jeweiligen Services. So können Anfänger recht schnell ein Verständnis für den Aufbau dieser erlernen und zeitnah erste Erfolge erzielen. Gerade bei einer so komplexen Plattform, wie AWS sie darstellt ist dies als sehr hilfreich und sinnvoll anzusehen.

Werden für eine Anwendung, die entwickelt werden soll zusätzliche Bibliotheken benötigt, so können diese einfach nachinstalliert und eingebunden werden. Dazu muss für Windows xy und für Linux xy installiert werden. Diese können dann in der Eingabeaufforderung des jeweiligen Systems verwendet werden. Die so installierten Dateien können dann in den Zip-Ordner mit den Fuktionen und Codedateien eingefügt und hochgeladen werden. Der Aufruf der Bibliotheken erfolgt dann in der Code-Datei, in der sie verwendet werden soll.

Die Sprache auf der Plattform AWS ist in Englisch gehalten und kann nicht verändert werden. Damit bietet sie eine einheitliche international gültige Sprache, die in sämtlichen Ländern der Welt verwendet werden kann. So können auch Programmierer verschiedener Länder oder Unternehmen mit Standorten auf der ganzen Welt miteinander arbeiten und die Daten gemeinsam nutzen.

- memory tu use lambda
- timeout lambda
- roles s. 80

4.4.2 Regionen

Allgemein kann in AWS die Region ausgewählt werden, in der sich der Anwender befindet. Dies dient dazu, damit möglichst die am nächsten liegende Region vom Programmierer aus gewählt werden kann, um so die Latenz zu reduzieren und schnellere Zugriffszeiten erhalten zu können. Dies ist gerade für größere Anwendungen, die eine Vielzahl an Zugriffen haben wichtig zu berücksichtigen, da das System sonst zu langsam arbeitet und somit mögliche Kunden abschreckt. Zur Verfügung stehen dabei folgende Hauptregionen: US-west, US-east, Canada, EU, Asia Pacific und South America. Jede dieser übergeordneten Regionen hat Unterregionen, die sich auf bestimmte Städte beziehen. Insgesamt gibt es 14 verschiedene dieser Unterteilungen.
Dabei ist unbedingt zu beachten, dass nicht jeder Service, den AWS zur Verfügung stellt, in jeder der Regionen verfügbar ist. Deshalb sollte sich ein Programmierer oder Unternehmen von vornherein darüber bewusst sein was es genau an Ressourcen und Möglichkeiten für das anvisierte Projekt benötigt. Wird dabei etwas übersehen, so kann es vorkommen, dass ein Teil nicht umgesetzt werden kann weil der benötigte Service nicht in der am nächsten gelegenen Region verfügbar ist. Natürlich kann in dem Fall einfach eine Region ausgewählt werden, die den gewünschten Service anbietet. Dies ist jedoch nicht ohne Probleme durchführbar. Wurde bereits Services in einer anderen Region verwendet und zum Beispiel in Lambda Funktionen erstellt, so können diese nicht in einer anderen Region als der in

der sie erstellt wurden verwendet werden. Kommt es also zu einem gezwungenen Wechsel wegen nicht Verfügbarkeit eines Services, so müssen die bereits erstellten Inhalte in der neu gewählten Region ebenfalls erstellt werden. Dies bedeutet einen erheblichen Mehraufwand für den Programmierer und sollte unbedingt vermieden werden. Da AWS an sich über eine sehr gute Dokumentation über ihre Plattform und deren Services verfügt, sollten diese Informationen im Vorfeld auch genutzt und hinreichend studiert werden.
Aus einem Wechsel der Region ergibt sich zudem die Problematik einer geteilten Rechnungsstellung, da jede Region ihre eigene Rechnung erzeugt. Dies birgt die Gefahr, dass eine Rechnung übersehen wird und der AWS Account als Folge gesperrt werden kann.

In AWS können Rollen erstellt und verwaltet werden, Diese können bei unterschiedlichen Diensten dann aus einer Liste aller Rollen ausgewählt und verwendet werden. Die einzelnen Rollen tragen Zugriffsrechte in sich, die in den Anwendungen bestimmte Teile zugänglich machen oder eben den Zugriff verwehren. Insbesondere für die Verwendung einer Anwendung mit Anmeldefunktion verschiedener User ist wichtig. Dabei existieren bereits viele Rollen für unterschiedliche Services und Gegebenheiten, die verwendet oder geändert werden und somit an die jeweiligen Anforderungen angepasst werden können.

Um eine Anwendung für den Browser zu erstellen, können HTML Dateien verwendet werden, die dann mit einer ausführenden Node.js Datei und den gewünschten AWS Services verbunden werden können. So können verschiedene Elemente, die nicht in AWS integriert sind, zum Aufbau der gewünschten Anwendung in unterschiedlichen Bereichen erstellt werden ohne dabei auf Flexibilität in den Gestaltungsmöglichkeiten verzichten zu müssen.

4.5 AWS Lambda

AWS Lambda gehört zum Anbieter Amazon und wurde 2014 vorgestellt. Eine nähere Suche im Internet zum Thema Serverless Programmierung zeigt, dass AWS Lambda häufig im Zuge von Fachartikeln verwendet wird. Ebenfalls existieren viele Tutorials dazu, so dass hier eine große Verbreitung von Lambda zu erkennen ist. Es ist daher gut möglich als Einsteiger in die Thematik von AWS Lambda einzusteigen. Zusätzlich ist eine ausgiebige Dokumentation vorhanden, welche die Möglichkeiten der Verwendung aufzeigt und näher bringt. Da AWS Lambda durch seine Einführung 2014 eines des ersten Plattformen war, die Serverless Programmierung ermöglicht hat, ist dementsprechend auch verhältnismäßig viel Literatur in Form von Anleitungen und Foren vorhanden. So wird es Einsteigern erleichtert sich damit zurecht zu finden.

AWS Lambda unterstützt standardmäßig Node.js und Python und bietet den Verwendern der Entwicklungsumgebung Eclipse die Möglichkeit zusätzliche Plug-ins einbinden zu können, welche den Umgang mit den Funktionen erleichtert. Aber auch andere Umgebungen können verwendet werden, ganz nach Belieben des Entwicklers. Weiterhin können auch andere Programmiersprachen wie Java verwendet werden, diese sind jedoch nicht direkt in AWS verwendbar, sondern müssen erst extern eingebunden werden. Dennoch werden so viele Anwendungsmöglichkeiten und vor allem Flexibilität für Programmierer und Unternehmen angeboten.

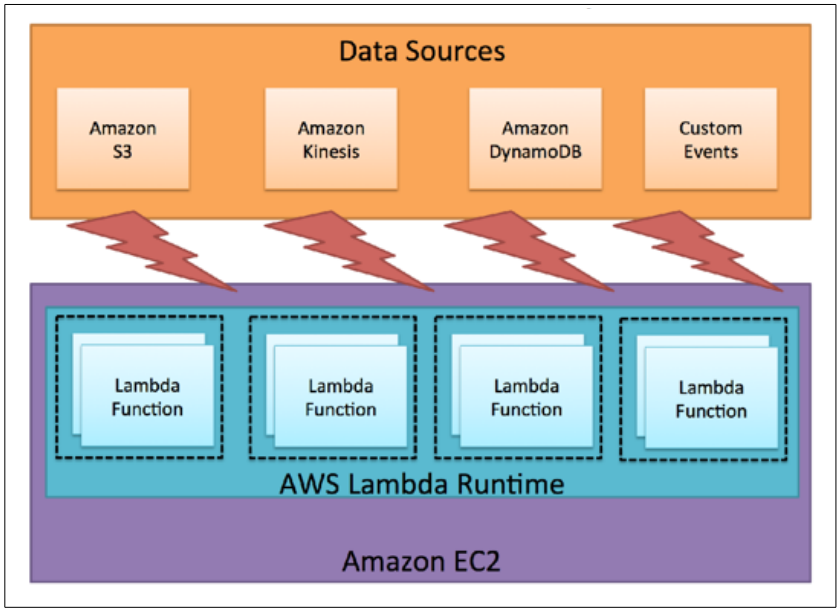
Eine neue Funktion kann recht einfach und übersichtlich erstellt werden. Dazu wählt der Anwender aus der Übersicht der Services unter dem Punkt „Compute“ den Eintrag Lambda aus. Danach öffnet sich eine Oberfläche, auf der alle bisher erstellten Funktionen in der gewählten Region aufgelistet sind. Dort kann der Programmierer die vorhanden Funktionen aufrufen und verändern oder er erstellt eine neue. Als nächstes

erscheint eine Auswahl an sogenannten Blueprints. Diese stellen Beispiele von verschiedenen Einsatzmöglichkeiten dar, die verwendet werden können. Die erste Auswahl in der Übersicht zeigt einen blanko-Blueprint, bei dem der Anwender alle Einstellungen selber vornehmen muss. Nachdem eine Auswahl getroffen wurde, erscheinen die Einstellungsmöglichkeiten. Dabei wird zunächst der Name der Funktion nebst einer Beschreibung dieser festgelegt. Anschließend folgt ein Codefeld, in welches der Code für die Lambda-Funktion geschrieben werden kann. Darüber befindet sich eine Drop-Down Liste, aus der die Art, wie der Code integriert werden soll, ausgewählt werden kann. Standardmäßig ist die Eingabe des Codes direkt in der Maske voreingestellt. Die anderen beiden Optionen ermöglichen es entweder den Code per Zip-Datei oder per Integration aus S3 zur Verfügung zu stellen. Weiterhin muss ausgewählt welche Art von Code verwendet werden soll. Dabei steht Node.js, Java, C#, Python und Edge Node.js zur Verfügung. Andere Sprachen können über Umwege verwendet werden wenn es gewünscht wird. Auch hier gibt es eine Voreinstellung, die in diesem Fall Node.js ist. Nachdem der Code erstellt oder integriert wurde, muss der gewünschte Handler ausgewählt werden ebenso wie die Rolle, die für diese Funktion verwendet werden soll. Dabei kann jede zuvor erstellte Rolle verwendet werden oder aber eine neue erstellt werden. Bei der Auswahl der Angaben zum Speicherplatz
Nachdem die weiteren Einstellungen vorgenommen wurden, wird dem Anwender eine Zusammenfassung gegeben. Anschließend kann die Funktion erstellt und verwendet werden. Die Funktion kann nun getestet oder verändert werden. Dabei können Änderungen an dem Code sowie an den Einstellungen vorgenommen werden.

Weiterhin bietet AWS Lambda eine automatisierte Verwaltung der Dokumente. So können Entwickler übersichtlicher arbeiten und haben ihre Dokumente jederzeit griffbereit. Weiterhin sorgt Amazon für eine

regelmäßig Aktualisierung der Server, so dass eventuell auftretende Komplikationen minimiert werden. Durch den Service, den AWS bietet, können sich Kunden bei Problemen umgehend mit den Mitarbeitern in Verbindung setzen. So können Ausfälle oder andere Fehler schnell und effektiv gelöst werden. Mit AWS Lambda können neue Backend-Services für Anwendungen erstellt werden, die On-Demand ausgelöst werden und dazu die Lambda API oder mit Amazon API Gateway entwickelte benutzerdefinierte API-Endpunkte verwenden. Außerdem kann mit AWS Lambda eine benutzerdefinierte Logik zu AWS-Ressourcen wie Amazon S3-Buckets und Amazon Dynamo DB-Tabellen hinzugefügt werden. Durch die nutzungsabhängige Zahlung zahlt der Kunde nur das, was er auch wirklich an Serverleistung verbraucht hat ohne fixe Kosten eingehen zu müssen.

Die folgende Grafik zeigt mögliche Kombinationen der AWS Komponenten für Serverless Programmierung.



Wenn es um das Thema Literatur geht, so ist es AWS Lambda welches dort am meisten Beachtung findet. Die wenigen Bücher, die zum Thema Serverless Programmierung bisher existieren oder in naher Zukunft veröffentlicht werden, beinhalten Beispiele und Erläuterungen, welche sich

auf AWS Lambda beziehen. Auch in Artikeln im Internet oder Tutorials wird häufig mit AWS Lambda gearbeitet. Dies ist mitunter der Tatsache geschuldet, dass andere vergleichbare Plattformen, die sich mit der Thematik des Serverless Programmierung beschäftigen erst später auf dem Markt erschienen sind und sich deshalb diverse Autoren und Menschen vom Fach mit AWS Lambda auseinandergesetzt haben. Da jede Plattform einen eigenen Aufbau hat und mit unterschiedlichen Services und Diensten arbeitet, ist es nicht möglich diese miteinander zu kombinieren. Wer sich mit einer Plattform beschäftigt hat, der wird sich aller Wahrscheinlichkeit nach nicht auch noch mit einer anderen auseinandersetzen.

5. Erläuterung des Prototypen

Im Zuge dieser Ausarbeitung soll ein Prototyp erstellt werden, der die auf den vorherigen Seiten vorgestellten Möglichkeiten der Serverless Programmierung aufgreift und verdeutlicht. Durch die praktische Vorführung einer gezielten Anwendung lassen sich die Vorteile zeigen und es findet eine Verknüpfung mit dem theoretischen Teil des Dokuments statt. Zunächst sollen daher die Anforderungen und Ziele, die an den Prototyps gestellt werden, erläutert werden. Anschließend soll der Aufbau der Anwendung betrachtet und die einzelnen Elemente, die dabei verwendet werden sollen, erläutert werden.

5.1 Anforderungen und Ziele

Die Anforderungen an diesen Prototypen beziehen sich maßgeblich darauf verschiedene Elemente, die AWS bietet miteinander zu vereinen und so die Flexibilität und die Komplexität der Plattform in Verbindung mit Serverless Programmierung zu zeigen.

Weiterhin soll der Prototyp so aufgebaut sein, dass es die Möglichkeit gibt für die nachfolgende Performance-Messung Daten sammeln und vergleichen zu können für einzelne Aktionen der Anwendung und der Anwendung an sich. Dies setzt voraus, dass auch wissenschaftliche Mittel genutzt werden, die eine Messung ermöglichen.

Geplant ist dabei eine Anbindung an eine Datenbank zu ermöglichen, in der Daten gespeichert werden, die Anwender dann abrufen können durch Aktionen in der Anwendung. Weiterhin soll auch eine Speicherung von Daten von Anwendern erfolgen können.
Die Anwendung an sich soll mit Hilfe von HTML erstellt werden.

5.2 Aufbau

Der konkrete Aufbau der Anwendung soll dabei die folgenden Teile und Elemente beinhalten:

- → HTML Gerüst
  - → Begrüßungsseite
  - → Unterseite Hochladen von Dateien
  - → Einloggen für Nutzer
  - → Registrierung als Nutzer
  - → Verwaltung der Dateien als angemeldeter Nutzer
  - → Hochladen mehrerer Dateien zeitgleich
- → Datenbank
  - → speichern der hochgeladenen Dateien
  - → speichern der Beschreibung der Dateien
- → AWS
  - → Lambda Funktionen
    - → createrUser

→ login

→ Roles

→ Policies

→ S3 Bucket

Die Anwendung soll demnach eine HTML Homepage als Anwendungsoberfläche haben, auf der die einzelnen Aktionen ausgeführt und angewählt werden können. Zunächst sehen alle Anwender eine Begrüßungsseite, die kurz die Anwendung erläutert. Dann gibt es einen Button, über den der Anwender zu einer neuen Unterseite gelangt und sich als Nutzer auf der Seite registrieren kann. Dazu muss er seinen Namen und seine E-Mail Adresse eingeben. Damit der User verifiziert werden kann, erhält er eine Bestätigungsmail zu seiner Registrierung, in der er den mitgesendeten Link anklicken muss, damit seine Anmeldung abgeschlossen werden kann. Erst wenn dieser Vorgang vollständig abgeschlossen wurde ist er auch mit seinen Anmeldedaten in der Datenbank als User hinterlegt. Er kann sich nun auf der Startseite durch den Button „Login“ einloggen, in dem er seine Daten für E-Mail Adresse und Passwort eingibt, die er bei der Registrierung verwendet hat. Wurden diese mit der Datenbank abgeglichen und als richtig bestätigt, so kann er auf den internen Bereich zugreifen.
Auch Anwender, die sich nicht registrieren möchten, können mit der Anwendung arbeiten. Sie können über einen Button die Datei von ihrem Rechner auswählen, die sie gerne hochladen möchten. Durch einen Klick auf den Hochladen-Button wird der Vorgang dann gestartet. Nachdem die Datei erfolgreich hochgeladen wurde, kann sie durch einen Klick auf den Dateinamen in einem neuen Fenster aufgerufen werden und erhält eine eigene Web-Adresse, welche den Namen der Datei enthält.
Handelt es sich um einen registrierten Nutzer, so können Beschreibungen zu der Datei hinzugefügt werden. Weiterhin kann er im internen Bereich alle bisher hochgeladenen Dateien einsehen. Ein Anwender ohne Registrierung

kann nur die jeweils gerade hochgeladene Datei einsehen. Zudem kann er auch nur Bilder hochladen.

Auf jeder Seite der Anwendung hat ein registrierter und gerade eingeloggter Nutzer die Möglichkeit sich aus seinem Account auszuloggen. Für das Hochladen von Dateien gibt es eine Grenze von xy Stück.

6. Erläuterung Erarbeitung des Prototypen

Kommt noch

7. Performance-Messungen

Ein Ziel der Serverless Programmierung ist es, dass die Anwendungen bessere Reaktionszeiten erzielen können. Durch die geänderte und vereinfachte Struktur, bei der nicht unnötig viele Schichten durchlaufen werden müssen, können Aktionen in einer Anwendung schneller ausgeführt werden. Um zu verdeutlichen wie dies konkret zu bewerten ist, sollen im Zuge dieser Ausarbeitung anhand des vorgestellten Prototyps Performance-Messungen durchgeführt werden. Diese beziehen sich auf die Latenz und den Durchschlag. Dazu werden bei Aufruf der Anwendung und der Durchführung von Aktionen innerhalb dieser die Daten aufgezeichnet und analysiert.

Um möglichst verschiedene Messergebnisse erhalten zu können, die dann miteinander verglichen werden können, sollen Zugriffe auf die Anwendung simuliert werden. Dabei werden verschiedene Abstufungen an fiktiven Usern, die auf die Anwendung zugreifen, festgelegt. So soll die Effizienz von der Serverless Programmierung aufgezeigt werden. Weiterhin sollen verschiedene Aktionen innerhalb der Anwendungen aufgerufen werden und die Performance-Messungen anschließend vergleichen zu können. Durch die große Anzahl an Messergebnissen, die aus den getroffenen Anforderungen für die Messungen entstehen, kann die Effizienz der Anwendung gut beurteilt werden.

Um die fiktiven Anwender zu simulieren wird auf eine Software zurückgegriffen, welche sich auf Performance Test für Anwendungen spezialisiert hat und auch mit der Plattform AWS integriert werden kann. Dabei stehen unterschiedliche Anbieter einer solchen Software zur Verfügung, die jeweils ihre eigene Umgebung mit sich bringen, um die nötigen Einstellungen und Ergebnisse präsentieren zu können.
AWS selber bietet auf seiner Marketplace-Homepage verschiedene

Möglichkeiten an, Messungen von anderen Anbietern auf ihrer Plattform durchführen zu können. Mitunter auch von Anbietern, die eigentlich nur mit Unternehmen einer gewissen Größe zusammenarbeiten und kein Interesse an Privatkunden hat. Zu beobachten ist auch, dass Anbieter von Hardware solche Testmöglichkeiten anbieten, die auf ihre Produkte zugeschnitten sind und somit eine noch bessere Integration ermöglichen.

8. Ergebnis

Kommt noch

9. Vor - und Nachteile Serverless Programmierung

Um auf die eigentliche Problemstellung dieser Ausarbeitung einzugehen, sollen im Folgenden nun die Vor- und Nachteile der Serverless Programmierung näher betrachtet werden. Diese sind nun nachdem die derzeit existenten Methoden sowie die Serverless Programmierung hinsichtlich ihres Aufbaus erläutert wurden sowie nach Umsetzung des Prototyps als Praxisbeispiel in vollem Umfang definierbar. Zunächst wird dabei auf die Vorteile Bezug genommen, um abschließend die Nachteile aufzulisten und zu erläutern.

9.1 Vorteile

Der wichtigste Punkt an dieser Stelle ist, dass es weniger Kosten verursacht. Dadurch, dass ein Unternehmen oder der Entwickler nicht mehr gezwungen ist alle Komponenten, die benötigt werden, selber im Besitz haben zu müssen, muss er auch dementsprechend weniger bezahlen. So ist eine Kostenersparnis möglich, da

Ein weiterer Hauptgrund, der für die Verwendung von Serverless Programmierung spricht ist die Skalierbarkeit. Diese läuft hierbei völlig automatisch ab und der Entwickler muss sich dieses Problems nicht mehr annehmen. Je nachdem wie viele Anwender auf das Programm zugreifen gibt er Leistung frei oder reduziert sie.

Durch die Verwendung von Serverless Programmierung wird weniger Code benötigt. So müssen die Komponenten nicht wie vorher alle miteinander verbunden und verknüpft werden. Außerdem werden zunehmend nur noch Events programmiert, die dann je nach Aktion in der Anwendung ausgelöst werden und schon definierten Code enthalten. Dieser muss vom Programmierer dann nicht mehr erstellt werden. Er selber erstellt

dann nur die benötigten Events. Dies bedeutet einen erhebliche Ersparnis an Code. So entsteht zum einen eine enorme Zeitersparnis, aber auch ein übersichtlicherer Code, in den sich andere Programmierer besser und schneller einarbeiten können. So kann im Notfall auch relativ schnell jemand Fremdes an dem System arbeiten und benötigt keine lange Einarbeitungszeit in den Code. Zudem wird die Anzahl der möglichen Fehler reduziert, da durch den übersichtlicheren Code Fehlerquellen besser und schneller identifiziert und ausgebessert werden können.

Ein weiterer Vorteil bietet die Tatsache, dass es keine fixen Kosten mehr gibt wenn auf eine serverless Methode zurückgegriffen wird. Dies ergibt sich daraus, dass kein Server mehr angemietet werden muss. Diese werden in fixen Größen zum mieten angeboten und nehmen keine Rücksicht darauf ob die Servergröße für das jeweils vorliegende Projekt überhaupt benötigt wird. Der Entwickler bzw. das Unternehmen wird gezwungen mehr zu mieten, als er eigentlich braucht und hat deshalb keinen Einfluss auf die entstehenden Kosten. Bei der Serverless Programmierung wird jedoch kein Server mehr angemietet, sondern Platz auf dem Server eines Drittanbieters verwendet. So kann nur das was auch wirklich an Serverleistung benötigt wird am Ende bezahlt werden. Somit resultieren im Umkehrschluss daraus monatliche variable Kosten, da die Zugriffe auf die Anwendung und die damit ausgelösten Events immer unterschiedlich sind.

Die Erstellung einer Anwendung basierend auf Serverless Programmierung bietet eine hohe Flexibilität. So können Änderungen am System unkompliziert vorgenommen werden ohne dass zeitgleich auch eine Änderung anderer Komponenten wie dem Server erfolgen muss. Ändert sich die Anzahl der Zugriffe in der Anwendung, dann

Ebenfalls ein Vorteil ist zu benennen in der Tatsache, dass weniger Personal benötigt wird. So müssen nicht extra Mitarbeiter eingestellt

werden, welche sich um die Wartung der Server kümmern. Diese müssen auf dem aktuellsten Stand gehalten werden und korrekt eingebunden werden. Da der Drittanbieter jedoch die Wartung seiner Server selber übernimmt wird somit das Personal nicht benötigt für diese Aufgabe. Weiterhin müssen sich die Programmierer auch nicht in die Thematik der Server einarbeiten, um dies in ihrer Planung zu berücksichtigen oder sich selber in diese Einzuarbeiten, um im Notfall reagieren zu können.

Ein Vorteil, der die Qualität der Anwendung beeinflusst, ist in der Arbeitsweise der Programmierer zu finden. Diese waren bisher dazu gezwungen sich auf mehrere Bereiche konzentrieren zu müssen. So gehörte nicht nur die Erstellung des Codes der Anwendung zu ihren Aufgaben, sondern auch die Organisation bezüglich des Managements der Anwendung und mitunter auch der Server. Durch die Verwendung von Serverless Programmierung können sich die Entwickler vollkommen auf den zu erstellenden Code konzentrieren. Dies hat zur Folge, dass die Qualität des Code steigt, da die Konzentration nur auf diesem liegt und die Programmierer nicht durch andere Umstände abgelenkt werden und ihre Arbeit unterbrechen müssen. Weiterhin

Ein weiterer Punkt betrifft die Größe und den Umfang von Projekten, die nun keine Limits kennen. Serverless Programmierung kann von jedem Entwickler verwendet werden. Dabei ist es unerheblich ob es sich um einen Privatanwender handelt, der in seiner Freizeit an einer Anwendung arbeitet oder ob es ein großes Unternehmen ist, welches die Server für seine Zwecke nutzen möchte. Gerade Privatpersonen haben so die Möglichkeit sich auszuprobieren ohne einen Server anmieten zu müssen. Da sie verhältnismäßig wenig Serverauslastung haben werden ist die Methode nur das zu zahlen was auch verwendet wurde für sie ideal um günstig und schnell Anwendungen testen zu können. Zwar sind die Drittanbieter potentiell an Unternehmen interessiert und nicht an kleinen

Privatanwendern, aber auch diese werden dort bedient.

Serverless Programmierung bietet den Vorteil, dass unterschiedliche Programmiersprachen miteinander kombiniert werden können. So ist es möglich, dass Mitarbeiter an einem Projekt arbeiten und dabei ihren Code in verschiedenen Sprachen schreiben. Diese Dateien können dann trotzdem miteinander verbunden und in die Anwendung eingebaut werden, ohne dass Probleme dabei entstehen und Teile nicht funktionieren.

Durch Serverless Programmierung können mehrere Anwendungen zeitgleich angesprochen werden. Dies ist in der Hinsicht praktisch, als dass so auftretende Probleme in mehreren Anwendungen zeitgleich gelöst werden können anstatt jede Anwendung einzeln durchgehen zu müssen, um eine Lösung zu finden. So können Events in verschiedenen Anwendungen verwendet und aufgerufen werden. Dies spart Zeit und macht Änderungen in unterschiedlichen Anwendungen einfacher und minimiert die Möglichkeit von Fehlern.

Die Programmierung in einem Serverless Projekt bezieht sich auf die Definition von Events. Dadurch werden lange komplizierte Funktionen vermieden und die Möglichkeit von Fehler wird minimiert. Die Events lösen dann hinterlegte Funktionen aus und handeln selbstständig. Die Anwendung erkennt dann je nachdem welche Aktion vom Anwender ausgeführt wird welches Event er ausführen muss.

Im Bereich Serverless sind viele Möglichkeiten einer Umsetzung denkbar. Während derzeitig verbreitete Methoden eher nur in eine bestimmte Richtung laufen und daher nicht viele Wahlmöglichkeiten bieten, dann im Serverless Programmierung mit verschiedenen Methode gearbeitet werden. So können die unterschiedlichsten Projekte umgesetzt und gestaltet werden, je nach Belieben des Unternehmens oder des Kunden.

9.2 Nachteile

Einer der größeren Nachteile des Serverless Programmierung ist die Auslagerung bei Drittanbietern. Dies ist zwar aus dem Gesichtspunkt der Nutzung des Servers des Drittanbieters und den damit verbundenen Vorteilen als sinnvoll anzusehen, jedoch bedeutet dies auch eine Kontrolle durch eben diesen Drittanbieter. Denn dieser kann genau einsehen was auf seinem Server passiert und somit auch Details über die einzelnen Projekte und Anwendungen einsehen. Der fade Beigeschmack nicht der Einzige zu sein, der auf die Daten Zugriff hat verleiht eine Unsicherheit und setzt Vertrauen in den Drittanbieter voraus.

Ebenfalls ist es zwar ein Vorteil, dass sich keine zusätzlichen Mitarbeiter um die Server kümmern müssen, jedoch geht damit auch der Verlust über Serveroptimierungen einher. So müssen sich Entwickler darauf verlassen, dass der Drittanbieter sich auch zeitnah um Updates bemüht und der Server so zur Verfügung steht, wie er für die Anwendung benötigt wird.

Ein weiterer Nachteil ist die Einarbeitung in die jeweilige Funktionalität der Drittanbieter. Jeder der Drittanbieter hat ein eigens entwickeltes System, welches einen jeweils unterschiedlichen Aufbau hat. So kann ein Programmierer, der sich in System A eingearbeitet hat nicht auch mit System B oder C arbeiten. Dies ist jedoch hinderlich in der Hinsicht, dass die Plattformen stetig weiterentwickelt werden und sich dementsprechend die Wahl der Drittanbieter ändert und sich die Entwickler immer neu einarbeiten müssen in den Aufbau. Weiterhin können unterschiedliche Anwendungen verschiedene Sprachen benötigen und nicht jede Sprache wird überall unterstützt. Auch da ist ein Wechsel der Plattform als hinderlich anzusehen. Ebenfalls kann es auch passieren, dass ein Kunde eines Unternehmens auf die Verwendung einer bestimmten Plattform besteht da er da schon andere Anwendungen laufen hat oder er dem anderen Drittanbieter

nicht vertraut. Auch hier müssen sich die Entwickler erst einarbeiten wenn ihnen das System unbekannt ist oder es sind weitere Mitarbeiter nötig, die sich jeweils auf eine Plattform spezialisiert haben.

Für Entwickler ergibt sich ein Nachteil aus der Verfügbarkeit von Literatur zum Thema Serverless Programmierung. Auch wenn dieser Bereich immer mehr an Bedeutung gewinnt und die Zahl der Anwendungen, die mit Hilfe der Möglichkeiten erstellt werden stetig ansteigt, so ist dies doch ein relativ neues Gebiet. Einhergehend damit ist auch nur wenig Literatur dazu vorhanden. Teilweise sind derzeit Bücher in der Vorbereitung, die sich näher gehend mit der Thematik des Serverless Programmierung beschäftigen. Wer sich allerdings damit einarbeiten will hat zumeist nur die Dokumentationen der Drittanbieter zur Verfügung. Dies ist in sofern problematisch, als dass die Entwicklung einer Anwendung so unter Umständen mehr Zeit benötigt. Trifft ein Entwickler auf ein Problem, so kann er bei den bisher integrierten Systemen und deren Möglichkeiten sehr leicht Hilfe finden, indem er entweder Bücher dazu findet, er andere Programmierer um Hilfe fragen kann oder er sich in einem der zahlreichen Foren informieren kann. Er hat demnach recht viele Möglichkeiten eine schnelle und effiziente Lösung für sein Problem zu erhalten. Durch das Fehlen dieser Varianten beim Serverless Programmierung ist die Fehlerbehebung auf Grund fehlender Erfahrung recht schwierig und zeitaufwändiger. Auch die nur langsam anlaufende Verbreitung der Methoden tritt dazu bei, dass sich eine Community ebenso langsam entwickelt. Gerade Programmierer sind eher dazu geneigt altbekanntes zu verfolgen und sich nicht in Neues einarbeiten zu wollen. Die schlechte Verbreitung von Literatur hilft dabei nicht wirklich.

Die Auslagerung auf die Server eines Drittanbieters hat eine Unsicherheit zur Folge, die für manche Anwendungen nicht akzeptabel ist. So können die Daten eines Unternehmens in falsche Hände geraten, was

gerade bei sensiblen Daten wie bei Banken eine Katastrophe wäre. Zwar sind die Drittanbieter natürlich bemüht eine hohe Sicherheit für ihre Kunden zu gewährleisten, jedoch kann dies nicht zu 100% garantiert werden. Dies führt dazu, dass die Methode des Serverless Programmierung schlicht zu unsicher ist für Anwendungszwecke und Unternehmen, die mit sehr sensiblen Daten arbeiten. Deshalb muss ein möglicher Hackerangriff auf den Drittanbieter einkalkuliert und das Risiko mit dem Nutzen abgewägt werden.

Die Methode des Serverless Programmierung ist nicht für jegliche Art von Anwendung gedacht. Es gibt durchaus Anwendungsfälle wo es eher ratsam ist einen eigenen Server angemietet zu haben und nicht auf einen Drittanbieter auszuweichen. Deshalb sollte vor der Wahl der Methode immer erst bedacht werden um was für ein Projekt es sich handelt und wie genau sich die Funktionalität dessen gestaltet. Wenn dieser Punkt nicht genau ausgearbeitet wird, ist es im Verlauf der Erstellung der Anwendung unter Umständen zu spät, um die Methode zu wechseln und das Projekt kann scheitern.

Erschwerend für die Entwicklung von Anwendungen ist es, dass die Drittanbieter nicht jede Programmiersprache unterstützen und sich Entwickler so nach den Möglichkeiten der Anbieter richten müssen. So kann es vorkommen, dass eine Sprache verwendet werden muss, die für die jeweils vorliegende Anwendung nicht die beste ist, da sie sich nicht optimal dafür anbietet. Das volle Potenzial einer freien Wahl kann hier demnach Einbußen bei der Qualität mit sich bringen. Weiterhin bieten die Plattformen jeweils unterschiedliche Sprachen an, die sie unterstützen. Dies ist gerade bei einem möglichen Wechsel der Anbieter hinderlich wenn die neue Wahl die angedachte Sprache nicht unterstützt.

Generell birgt es einen gewissen Nachteil mit den Plattformen der

Drittanbieter zu arbeiten, da diese stetig weiterentwickelt werden und so ein Wechsel zu einem anderen Anbieter als sinnvoll erachtet werden kann. So ist es notwendig, dass die Entwickler sich stets und ständig über Neuerungen informieren und vor allem auf diese auch reagieren müssen. Sie müssen sich demnach darauf einstellen, dass sie mit einem sich ändernden System arbeiten und so eine Änderung der Funktionen erfolgen kann, auf die sie in der Lage sein müssen zu reagieren.

Tritt ein Problem bei einem Drittanbieter auf, so mitunter sämtliche Anwendungen, die dort gespeichert sich, nicht aufrufbar. Sollte es zu einem Ausfall des Servers kommen, so ist das eine Katastrophe für unzählige Unternehmen und bedeutet einen enormen Ausfall von Einnahmen. Zwar ist ein Szenario dieses Ausmaßes relativ unwahrscheinlich, jedoch kann es immer mal zu kleineren Teilausfällen führen. Bei einem eigenen Server ist ein Unternehmen auch nicht davor geschützt einen Ausfall oder andere tiefgreifendere Probleme haben zu können, jedoch ist es dann möglich direkt vor Ort und relativ schnell nach einer Lösung suchen zu können. Ist eine Firma auf den Drittanbieter angewiesen, so muss er mit diesem die schnelle Kommunikation suchen und ist darauf angewiesen, dass dieser schnell kooperiert und in der Lage ist den Fehler zeitnah zu finden und zu beheben.

9.3 Zusammenfassung des Kapitels

Die Auflistung und anschließende Erläuterung der Argumente der Vor- und Nachteile bezüglich der Serverless Programmierung hat gezeigt, dass es wie bei fast allem zwei Seiten der Medaille gibt. Jedoch ist die Gewichtung der Vorteile als stärker anzusehen, da es sich dabei um wichtigere Aspekte handelt, die für Anwendungen relevanter sind und höhere Auswirkungen auf eben diese haben, als die Nachteile. Im Vergleich zu den bisherigen derzeit

vorherrschenden Möglichkeiten, die die Softwareentwicklung bietet, sind die Vorteile der Serverless Programmierung stärker und besser für Unternehmen und Anbieter von Anwendungen.

10. Fazit

Die Ausarbeitung hat gezeigt, dass es zwar einige Möglichkeiten der regulären und bisher umgesetzten Möglichkeiten gibt in der Softwareentwicklung, jedoch sind diese auf Grund ihres Alters und der Dauer, die sie bereits auf dem Markt integriert sind, mitunter nicht mehr zeitgemäß. Die Technik hat sich in den letzten Jahren extrem verändert und deshalb ist es nötig sich auch im Bereich der Entwicklung von Anwendungen auf diese Neuerungen einzurichten und sich an sie anzupassen. Die Anforderungen an Software haben sich zudem verändert. Die Anwendungen sollen immer kleiner werden und auch auf mobilen Geräten aufrufbar sein. Viele existierende Programme werden nach und nach auch auf mobile Plattformen ausgeweitet und dort sind bisherige Verfahren nicht ohne weiteres übertragbar. Deshalb werden heute Methoden benötigt, die darauf basieren die Anfragen in einer Anwendung möglichst schnell und präzise abarbeiten zu können und dementsprechend schnell die Ergebnisse zu präsentieren. Heute ist insbesondere der mobile Anwendungsmarkt hart umkämpfr und jeden Tag schaffen es unzählige neue Apps in eine der verschiedenen App-Stores der unterschiedlichen Betriebssysteme. Wer sich dort dauerhaft etablieren möchte, der muss gut hinhören und sich die Kritik der Anwender zu Herzen nehmen. Auch wenn es immer mehr Mobilfunkanbieter gibt, die Flatrates für die mobile Datennutzung zur Verfügung stellen, so haben diese durch das tägliche Verwenden von diversen Apps meist keine allzu lange Lebensdauer und die anschließende Drosselung der Geschwindigkeit führt dazu, dass einige

Anwendungen für den User nicht mehr verwendbar sind. Bei wichtigen Apps, die auch unterwegs ohne W-LAN jederzeit aufrufbar sein müssen, ist dieser Zustand unvorteilhaft.

Die Vorteile der Serverless Programmierung haben deutlich gemacht welchen Nutzen die neue Möglichkeit der Softwareumsetzung bieten. Daher ist es als sinnvoll anzusehen, wenn Programmierer und Unternehmen sich nicht nur an alt bekannte Varianten einer Umsetzung halten, sondern sich an die aktuellen Gegebenheiten des Markts anpassen und sich nach der jeweils passendsten Methode für das jeweilige Projekt richten und vorher die Möglichkeiten gegeneinander abwägen, um eine möglichst effiziente Anwendung zu erhalten.

Unabhängig von der Tatsache, dass ich in der vorliegenden Ausarbeitung mit AWS als Plattform gearbeitet habe, weil diese recht weit verbreitet ist und sie gute Einarbeitungsmöglichkeiten für Anfänger bietet, empfinde ich AWS als sehr sinnvoll aufgebaut. Ich habe mich im Zuge von Kapitel 5 mit anderen Anbietern solcher Plattformen auseinandergesetzt und mir diese näher betrachtet und der Aufbau dieser war teilweise sehr unübersichtlich und durcheinander, so dass es oft nicht gut ersichtlich war wo sich welcher Service verbirgt und wie sich mit diesen arbeiten lässt. Zwar gab es oftmals Einführungen in Form von Dokumentationen dazu, jedoch sollte der Aufbau eigentlich nahezu selbsterklärend sein, um unnötig viel Zeit zu vermeiden und somit auch das Fehlerrisiko zu verringern. Bei AWS werden alle Services übersichtlich auf einen Blick dargestellt und sie sind schnell zu finden. Weiterhin werden die meisten der Services über eine vorgefertigte Maske erstellt, auf der alle relevanten Einstellungen und Codeeingabemöglichkeiten vorhanden sind, so dass der Anwender genau sehen kann, was von ihm erwartet wird. Bei anderen Plattformen werden die näheren Einstellungsmöglichkeiten erst später angezeigt und als Anwender fällt es schwer herauszufinden welche Einstellungen nun zwingend

notwendig sind und vor allem welche bei dem jeweiligen Anwendungsfall am sinnvollsten sind. AWS arbeitet auch viel mit Beispieleintragungen in den Textfeldern, so dass der Anwender schnell eine Vorstellung von dem bekommt, was er dort eintragen soll bzw. in welchem Format dies erfolgen soll. Auch die Tatsache, dass ich bei AWS nicht mit den Worten „Sie können 30 Tage testen“ begrüßt werde empfand ich als angenehm, da ich so nicht das Gefühl hatte ich muss mich gezwungener Maßen nun irgendwo anmelden und darauf achten etwas zu kündigen. Da ich beim Versandhandel Amazon bereits seit vielen Jahren einen Account habe, konnte ich mich mit den Daten von eben diesem auch bei AWS einloggen und musste so kein zusätzliches Konto erstellen. Weiterhin war die Auflistung meines Verbrauchs in den einzelnen Diensten in der Rechnungsübersicht sehr angenehmen. Zeitgleich wurde bereits ausgerechnet wie viel Prozent des freien Kontingents ich bei gleich bleibender Nutzung am Ende des Monats zur Verfügung haben werde, so dass ich damit besser planen konnte wie viel ich noch frei erstellen kann und wann ich aufpassen sollte. Die Einarbeitung in AWS wird durch die ausführliche Dokumentation der Plattform selber erleichtert.

11. Ausblick in die Zukunft

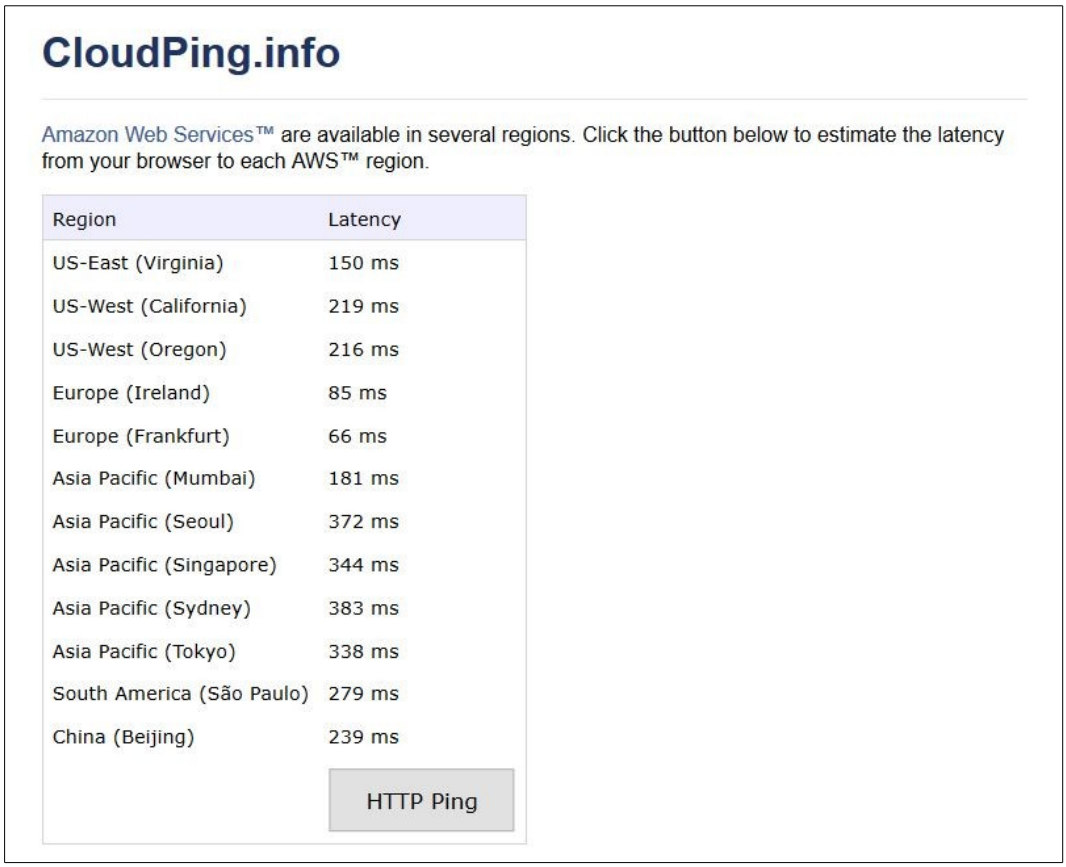
Wie bereits auf den vorherigen Seiten erwähnt, ist der Bereich der Serverless Programmierung keine Erfindung, die erst seit kurzem existiert. Bereits seit Jahren entwickelt er sich stetig weiter. Doch die nur langsam vorangehende Verbreitung dieser Möglichkeit sorgt einen langwierigen Weg.

Es sind derzeit einige Bücher in Planung und Bearbeitung, die sich explizit mit der Thematik der Serverless Programmierung beschäftigen und die dafür sorgen werden, dass dieses Thema mehr Aufmerksamkeit erhalten wird. Allerdings ist dabei festzuhalten, dass es sich dabei um englischsprachige Literatur handelt und es wohl noch einige Jahren dauern wird, bis diese auch in den jeweiligen Landessprachen der einzelnen Länder zur Verfügung stehen wird.

Generell ist es als sinnvoll anzusehen, dass die Anbieter der Plattformen, die Serverless Programmierung bisher anbieten, möglichst große und bekannte Unternehmen für sich gewinnen, damit diese durch die Verwendung der Plattformen Werbung für eben diese machen und so auch andere Firmen auf die Möglichkeit der Serverless Programmierung aufmerksam werden und sich ebenfalls damit auseinandersetzen. Weiterhin werden so auch Programmierer darauf aufmerksam, wenn in der Öffentlichkeit mehr Anwendungen serverless erstellt werden und so ihr Interesse geweckt wird.

Abbildungsverzeichnis

Tabellenverzeichnis



CloudPing.info

Amazon Web Services™ are available in several regions. Click the button below to estimate the latency from your browser to each AWS™ region.

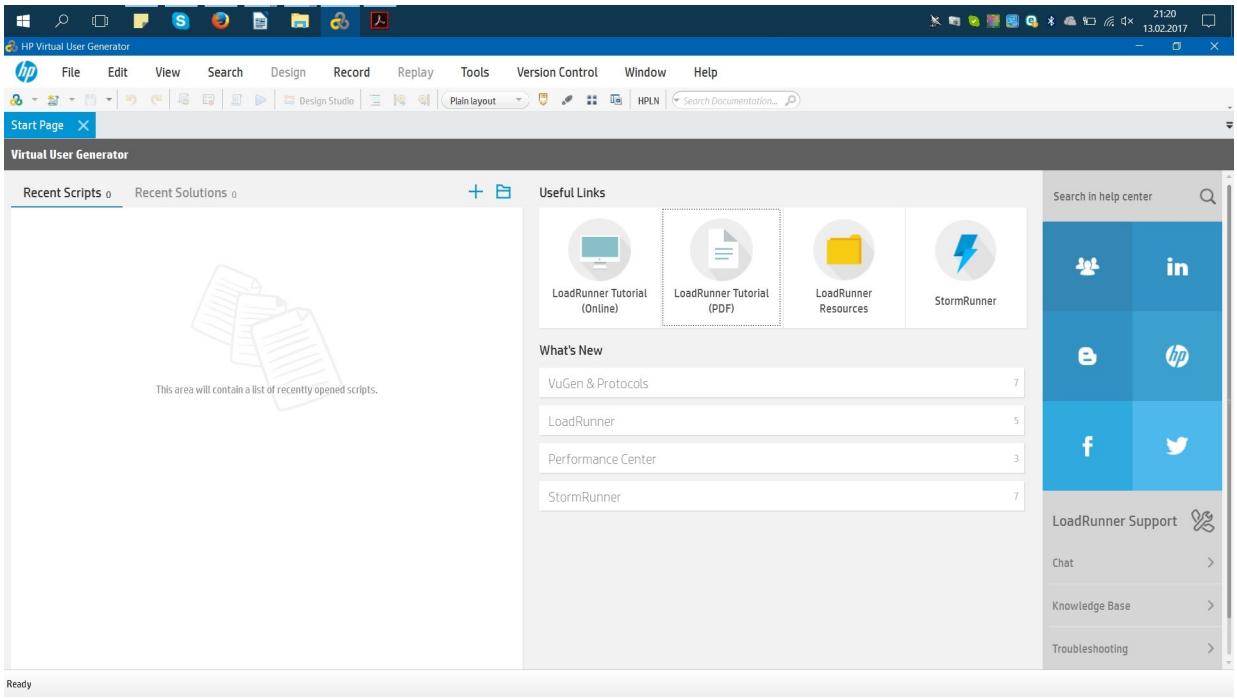
Region	Latency
US-East (Virginia)	150 ms
US-West (California)	219 ms
US-West (Oregon)	216 ms
Europe (Ireland)	85 ms
Europe (Frankfurt)	66 ms
Asia Pacific (Mumbai)	181 ms
Asia Pacific (Seoul)	372 ms
Asia Pacific (Singapore)	344 ms
Asia Pacific (Sydney)	383 ms
Asia Pacific (Tokyo)	338 ms
South America (São Paulo)	279 ms
China (Beijing)	239 ms

HTTP Ping

Auf dieser Seite ist es möglich die Latenz von dem eigenen Standort aus zu den einzelnen Regionen, die in AWS angeboten werden, zu messen. Es ist interessant zu sehen wie sich die Werte verhalten und wie groß die Unterschiede sind.

#	Resource	Content Type	Request Start	DNS Lookup	Initial Connection	SSL Negotiation	Time to First Byte	Content Download	Bytes Downloaded	Certificates	Error/Status Code	IP
1	http://ss.symcd.com/	application/ocsp-response	0.216 s	31 ms	42 ms	-	48 ms	1 ms	1.9 KB	-	200	23.63.139.27
2	https://eu-central-1...?region=eu-central-1	text/html	0.267 s	61 ms	31 ms	81 ms	67 ms	2 ms	2.1 KB	2937 B	200	54.239.54.107
3	https://eu-central-1...-1&state=hashArgs%23	-	0.394 s	-	-	-	74 ms	-	1.6 KB	-	302	54.239.54.107
4	https://eu-central-1...azon.com/favicon.ico	image/vnd.microsoft.icon	0.504 s	-	30 ms	48 ms	34 ms	2 ms	1.4 KB	2937 B	200	54.239.54.107
5	http://ss.symcd.com/	application/ocsp-response	0.77 s	-	-	-	50 ms	1 ms	1.9 KB	-	200	23.63.139.27
6	http://ss.symcd.com/	application/ocsp-response	0.821 s	-	42 ms	-	48 ms	1 ms	1.9 KB	-	200	23.63.139.27
7	https://eu-central-1...de&state=hashArgs%23	text/html	0.823 s	218 ms	30 ms	74 ms	42 ms	-	1.4 KB	3021 B	302	54.239.55.66
8	https://www.amazon.c...&forceMobileLayout=0	text/html	1.184 s	41 ms	117 ms	138 ms	180 ms	122 ms	15.8 KB	3049 B	200	54.239.26.128
9	https://images-na.ss...al._CB318885489_.css	text/css	1.916 s	53 ms	32 ms	220 ms	95 ms	-	7.4 KB	2980 B	200	52.222.155.114
10	https://images-na.ss...sl._CB219086192_.css	text/css	1.916 s	-	31 ms	232 ms	105 ms	1 ms	1.1 KB	2980 B	200	52.222.155.114
11	https://images-na.ss...in._CB352383985_.css	text/css	1.916 s	-	31 ms	237 ms	71 ms	2 ms	0.8 KB	2980 B	200	52.222.155.114
12	https://images-na.ss...ew._CB176739463_.css	text/css	1.916 s	-	42 ms	238 ms	92 ms	3 ms	2.7 KB	2980 B	200	52.222.155.114
13	https://images-na.ss...min._CB253690767_.js	application/x-javascript	1.917 s	-	30 ms	369 ms	82 ms	-	16.9 KB	2980 B	200	52.222.155.114
14	https://images-na.ss...gin._CB340498465_.js	application/x-javascript	1.928 s	-	41 ms	394 ms	124 ms	1 ms	0.7 KB	2980 B	200	52.222.155.114
15	https://l0.awsstatic.com/js/urchin.js	application/javascript	1.93 s	80 ms	66 ms	327 ms	108 ms	58 ms	22.0 KB	2863 B	200	52.222.150.163
16	https://images-na.ss...1.1._CB306317608_.js	application/x-javascript	1.989 s	-	-	-	154 ms	-	8.6 KB	-	200	52.222.155.114
17	https://images-na.ss...cim._CB523784584_.js	application/x-javascript	1.999 s	-	-	-	67 ms	582 ms	213.0 KB	-	200	52.222.155.114
18	https://images-na.ss...ogo._V400518270_.png	image/png	2.022 s	-	-	-	121 ms	-	7.9 KB	-	200	52.222.155.114
19	https://images-na.ss...n_amazon_company.gif	image/gif	2.022 s	-	-	-	71 ms	-	1.4 KB	-	200	52.222.155.114
20	https://www.amazon.c...I-1&utmp=/ap/signin/	text/html	2.165 s	-	-	-	190 ms	-	1.3 KB	-	404	54.239.26.128
21	https://d1.awsstatic...S/Prospect_image.jpg	image/jpeg	2.169 s	41 ms	46 ms	78 ms	65 ms	266 ms	50.4 KB	2863 B	200	52.222.152.29
22	https://images-na.ss...e03._V395592492_.png	image/png	2.23 s	-	-	-	135 ms	135 ms	42.9 KB	-	200	52.222.155.114
23	https://fls-na.amazo...Y7TEC017HQXQ4FH2PB:0	image/gif	2.233 s	333 ms	121 ms	183 ms	146 ms	-	0.3 KB	2928 B	200	72.21.206.53
24	https://fls-na.amazo...360746-9229502&csm=1	image/gif	2.788 s	-	-	-	150 ms	-	0.3 KB	-	200	72.21.206.53
25	https://www.amazon.c...017HQXQ4FH2PB&aftb=1	image/gif	2.789 s	-	-	-	192 ms	-	0.5 KB	-	204	54.239.26.128
26	https://fls-na.amazo...H2PB%26aftb%3D1:1364	image/gif	2.789 s	-	142 ms	306 ms	214 ms	-	0.3 KB	2928 B	200	72.21.206.53
27	https://images-na.ss...a53c170e472c._V2_.js	application/x-javascript	2.79 s	-	-	-	93 ms	-	4.6 KB	-	200	52.222.155.114
28	https://www.amazon.com/favicon.ico	image/x-icon	3.159 s	-	-	-	185 ms	-	2.9 KB	-	200	54.239.26.128
29	https://images-na.ss...119:1487013464&vip=1	application/x-shockwave-flash	3.2 s	-	-	-	56 ms	-	5.1 KB	-	200	52.222.155.114
30	https://fls-na.amazon.com/1/batch/1/OE/	text/plain	4.253 s	-	-	-	141 ms	-	0.2 KB	-	204	72.21.206.53

Hier wurde die Seite der Konsole von AWS aufgerufen und die Werte für die einzelnen Elemente der Homepage gemessen.



Beim Programm LoadRunner von HP gibt es die Möglichkeit virtuelle User zu generieren und diese in verschiedene Anwendungen zu integrieren. Dabei gibt es ein freies Kontingent von 50 Usern, alles darüber hinaus muss bezahlt werden.

KLE11HEI12BEN14SAN14 et. alEigGi12BAL11WieMe08OST12HIL15SeeGu06StaHr11SCH05BraKo08TAB06RupSoSTE11HAM08GolDa13DIE15CHA16MÜNLOO16ABE16SCH16JAN16LIM16FRO16ROB16OZA16FUL16n.A.16an.A.16bJAN16ORE16JOH16STA14ReuHa06SCH10

Literaturverzeichnis

Abts, Dietmar (2015): Masterkurs Client/Server-Programmierung mit Java - Anwendungen entwickeln mit Standard-Technologien,Wiesbaden: Springer Fachmedien.

Avram, Abel (2016): FaaS, PaaS, and the Benefits of the Serverless Architecture, https://www.infoq.com/news/2016/06/faas-serverless-architecture (07.10.2016)

Balzert, Helmut (2011): Lehrbuch der Softwaretechnik - Entwurf, Implementierung, Installation und Betrieb, Heidelberg: Spektrum Akademischer Verlag.

Bengel, Günther (2014): Grundkurs Verteilte Systeme, Wiesbaden: Springer Vieweg.

Brandt-Pook, Hans / Kollmeier, Rainer (2008): Softwareentwicklung kompakt und verständlich - Wie Softwaresysteme entstehen, Wiesbaden: Vieweg+Teubner.

Eigner, Gerhardt / Gilz, Mogo Nem (2012): Informationstechnologie für Ingenieure, Berlin: Springer Vieweg.

Froehlich, Andrew (2016): Serverless Architecture Pros And Cons, http://www.networkcomputing.com/data-centers/serverless-architecture-pros-and-cons/2090515944 (26.10.2016)

Fuller, Chris (2016): Serverless Architectures: Monoliths, Nanoservices, Microservices & Hybrids, https://blog.gorillastack.com/serverless-architectures-monoliths-nanoservices-microservices-hybrid/ (26.10.2016)

Goll, Joachim / Dausmann, Manfred (2013): Architektur- und Entwurfsmuster der Softwaretechnik - Mit lauffähigen Beispielen in Java, Wiesbaden: Springer Fachmedien.

Hamilton, Patrick (2008): Wege aus der Softwarekrise, Berlin: Springer-Verlag.

Heil, Andreas (2012): Anwendungsentwicklung für Intelligente Umgebungen im Web Engineering, Wiesbaden: Springer Vieweg.

Hilbrich, Robert (2015): Platzierung von Softwarekomponenten auf Mehrkernprozessoren - Automatisierte Konstruktion und Analyse für funktionssichere Systeme, Wiesbaden: Springer Fachmedien.

Janakiram (2016): Five Serverless Computing Frameworks To Watch Out For, https://www.janakiram.com/posts/blog/five-serverless-computing-frameworks-to-watch-out-for (07.10.2016)

Johnson, Eric (2016): Part 1: Building a Serverless Architecture Wi AWS - Hosting on S3, https://blog.rackspace.com/part-1-building-server-less-architecture-aws (07.10.2016)

Kleuker, Stephan (2011): Grundkurs Software-Engineering mit UML, Wiesbaden: Vieweg+Teubner Verlag.

Limpalair, Christophe (2016): Serverless Architecture, https://linuxacademy.com/blog/amazon-web-services-2/serverless-architecture/ (07.10.2016)

Looi, Mark (2016): AWS Lambda and Serverless Computing, https://www.linkedin.com/pulse/aws-lambda-serverless-computing-mark-looi (26.10.2016)

Müns, Philipp (2016): AWS fundamentals: What is Lambda?, https://medium.com/just-serverless/aws-fundamentals-what-is-lambda-32d17a89dda2#.8otqy3blp (26.10.2016)

n.A. (2016): What is Amazon Lambda, https://www.iopipe.com/what-is-amazon-lambda/ (04.09.2016)

n.A. (2016): Using AWS Lambda and API Gateway to create a serverless schedule, https://www.import.io/post/using-amazon-lambda-and-api-gateway/ (07.10.2016)

O'Reilly, Dennis (2016): The Benefits of Serverless Applications in a Data-Rich World, https://dzone.com/articles/the-benefits-of-serverless-applications-in-a-data (26.10.2016)

Osterhage, Wolfgang W. (2012): Performance-Optimierung, Berlin: Springer-Verlag.

Ozar, Brent (2016): Serverless Projects - The Pros and Cons of Serverless Architecture, https://serverless-developers.com/2016/06/the-pros-and-cons-of-serverless-architecture/ (26.10.2016)

Patra, Chandan (2016): A Crash Course in Amazon Serverless Architecture: Discover the Power of Amazon API Gateway, Lambda, CloudFront, and S3, http://cloudacademy.com/blog/amazon-serverless-api-gateway-lambda-cloudfront-s3/ (07.10.2016)

Reussner, Ralf / Hasselbring, Wilhelm (2006): Handbuch der Softwarearchitektur, Heidelberg: dpunkt.verlag.

Roberts, Mike (2016): Serverless Architectures, https://martinfowler.com/articles/serverless.html (04.09.2016)

Rupp, Chris / die SOPHISTen (2013): Systemanalyse kompakt, Berlin: Springer-Verlag.

Sandhaus, Gregor / Berg, Björn / Knott, Philip (2014): Hybride Softwareentwicklung - Das beste aus klassischen und agilen Methoden in einem Modell vereint, Berlin: Springer-Verlag.

Schäfer, Werner (2010): Softwareentwicklung,München:Pearson Studium.
Scholz, Peter (2005): Softwareentwicklung eingebetteter Systeme, Berlin: Springer-Verlag.

Schott, Wendy (2016): Five advantages of severless architecture for every app, http://www.developer-tech.com/news/2016/oct/05/five-advantages-serverless-architecture-every-app/ (26.10.2016)

Seemann, Jochen / von Gudenberg, Jürgen Wolff (2006): Software Entwurf mit UML 2, Berlin: Springer-Verlag.

Starke, Gernot (2014): Effektive Softwarearchitekturen - Ein praktischer Leitpfaden, München: Carl Hanser Verlag.

Starke, Gernot / Hruschka, Peter (2011): Software-Architektur kompakt - angemessen und zielorientiert, Heidelberg: Spektrum Akademischer Verlag.

Stender, Peter (2011): Webprojekte realisieren nach neusten OOP-Kriterien - Ein Workshop über die Kooperation von PHP/XLS/JavaScript, Wiesbaden: Vieweg+Teubner Verlag.

Tabeling, Peter (2006): Softwaresysteme und ihre Modellierung, Berlin: Springer-Verlag.

Wieczorrek, Hans W. / Mertens, Peter (2008): Management von IT-Projekten, Berlin: Springer-Verlag.