Masterarbeit

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Thema: Serverless Programmierung: State of the Art und Evaluation

Abgabetermin Version:	02.01.2017
Abgabetermin Masterarbeit:	29.05.2017
Version:	14
Bearbeiter:	Svenja Haberditzl
Matrikelnummer:	830791
Betreuer:	Prof. Dr. Stefan Edlich

Inhaltsverzeichnis

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.

2. Einleitung

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

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 Programming 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 Perfomancemessungen 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 Messungsergebnisse und deren Bedeutung. Um die These, dass Serverless Programming 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 Programming 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 Programming gearbeitet und wie diese Möglichkeit etabliert werden kann.

3. 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 Programming, 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.

3.1 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.

3.2 Architekturarten

Im Laufe der letzten Jahrzehnte haben sich immer mehr unterschiedliche Architekturarten entwickelt. Um einen kleinen Einblick über die heute viel genutzten Möglichkeiten dieser gewähren zu können, werden diese im Folgenden kurz erläutert und hinsichtlich ihrer jeweiligen Vor- und Nachteile betrachtet.

3.2.1 Server/Client

Diese Variante der Programmierung besteht im wesentlichen daraus, dass die Komponenten aus einem oder mehrere Servern und einem Client bestehen. 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 Perfomance 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.

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	X x
Rich Client	X x
Thin Client	X x
Ultra-Thin Client	X x

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.

3.2.2 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.

3.2.3 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.

3.3 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.

3.4 Middleware

4. Serverless Programming

Der Bereich des Serverless Programming stellt das Herzstück der vorliegenden Dokumentation dar und soll deshalb nachfolgend erläutert und vertieft werden. Dabei wird auf die Definition ebenso wie auf die Funktionsweise Bezug genommen. So wird der Unterschied zu den bereits im vorherigen Kapitel erwähnten Möglichkeiten deutlich und die Vor- und Nachteile, die nach Erstellung des Prototypen betrachtet werden, können argumentatorisch untermauert werden.

4.1 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 Programming 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 Programming 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 Programming 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 Programming 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 Programming nicht mehr vorhanden, da die Skalierung automatisch geschieht. Wird mehr Serverleistung benötigt, so gibt der Drittanbieter automatisch mehr frei.
Geeignet ist Serverless Programming besonders für Nanoservices.

Das Ziel von Serverless Programming 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.

4.2 Funktionsweise

Die Funktionsweise beim Serverless Programming 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.

4.3 Etablierung am Markt

Obwohl das Prinzip des Serverless Programming kein gerade erst neu erfundene Variante der Softwareenticklung dartellt, 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 angemelerkt, 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.

5. Vorstellung der Möglichkeiten Serverless Programming

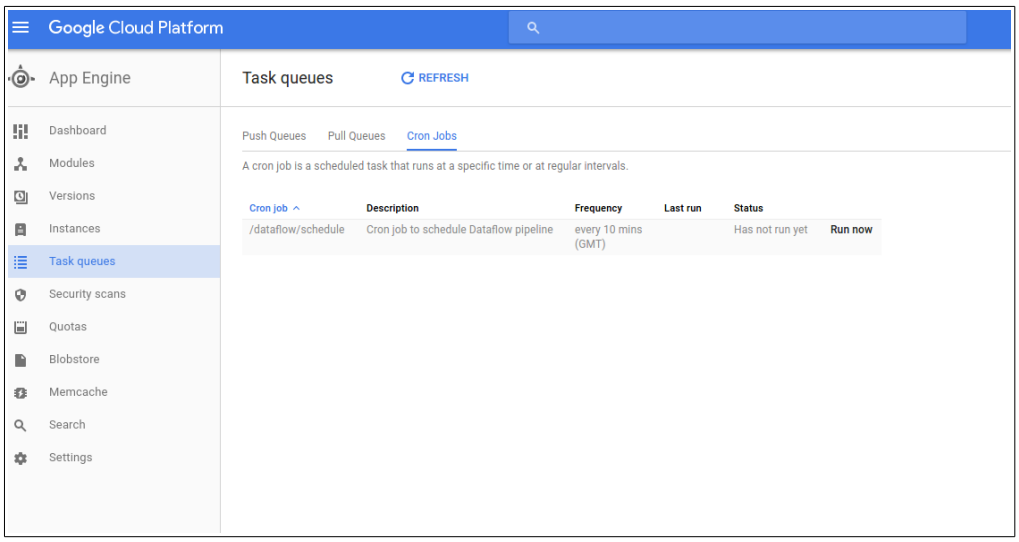
Im folgenden Kapitel sollen nun einige der Plattformen und deren Möglichkeiten des Serverless Programming vorgestellt werden. Dabei wird eine kurze Erläuterung über die jeweilige Plattform und ein Einblick und dessen Funktionsweise gegeben.

5.1 Google Cloud Functions

Google hat seine eigenen Plattform für Serverless erstellt, die im Jahre 20xy vorgestellt wurde. 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

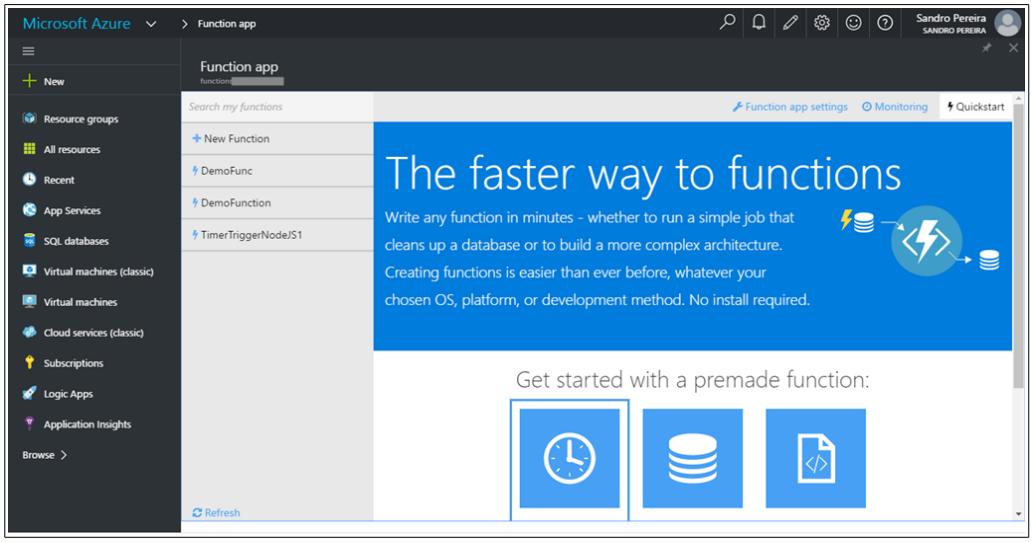
5.2 Windows Azure Functions

Windows Azure Functions ist die Möglichkeit des Serverless Programming, 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 Programming 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.

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.



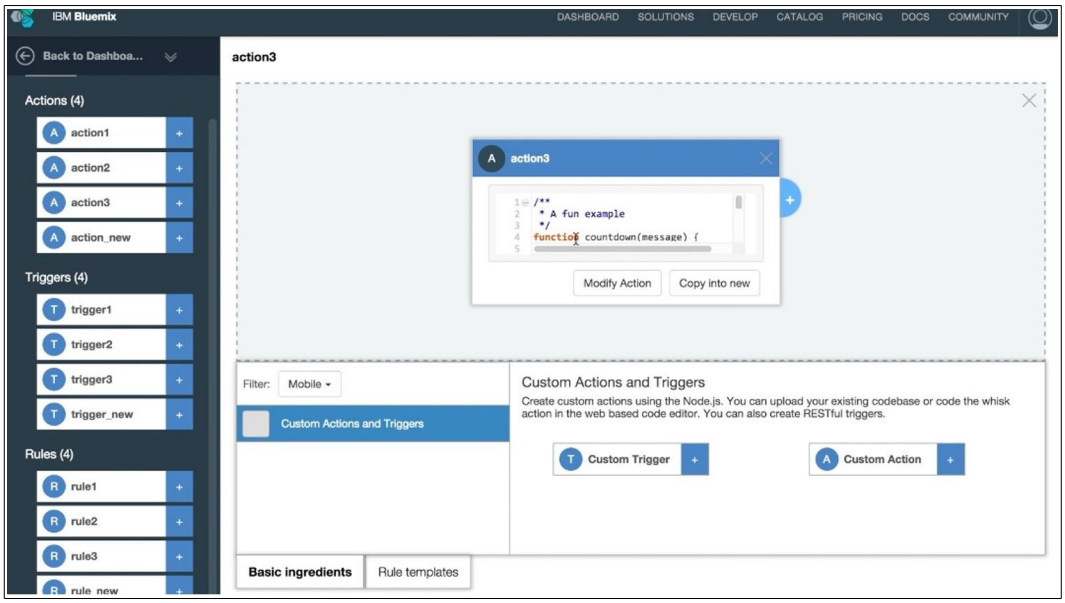
5.3 IBM Open Whisk

Auch IBM hat sich in die Reihe der Plattformen integriert, die Serverless Programming 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 serverlosen
Umgebung gewährleistet.

Die folgende Grafik zeigt den Aufbau der Oberfläche von Bluemix.



5.4 Amazon Webservices

In der vorliegenden Ausarbeitung geht es maßgeblich um AWS Lambda und somit um die Plattform von Amazon, die dafür mit ihren unterschiedlichen Diensten und Möglichkeiten verwendet wird. Deshalb soll an dieser Stelle ein kurzer Einblick in AWS gegeben werden.

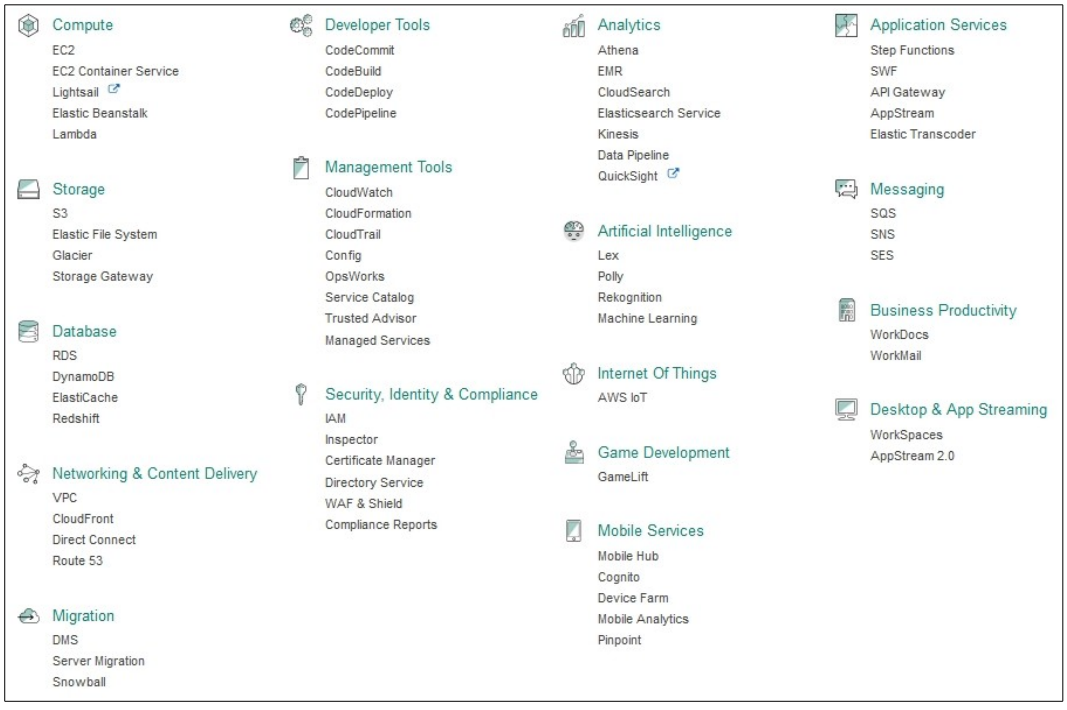
Die Plattform kann von jeglicher Art von Anwendern genutzt werden. Ein Privatnutzer, der nur kleine Anwendungen erstellen oder testen möchte ist ebenso willkommen wie das große Unternehmen, welches einiges an Serverleistung und Dienste nutzen möchte. Gerade Firmen gehen immer mehr dazu über sich an Drittanbieter 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.
Weiterhin können unterschiedliche Sicherheitsgruppen und Usergruppen erstellt werden, die für mehrere Anwendungen verwendet werden können oder jeweils für jede Anwendung neu erstellt werden. Diese und andere Einstellungsmöglichkeiten bieten ein hohes Maß an Flexibilität und spricht somit eine breite Zielgruppe an.

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.

5.5 AWS Lambda

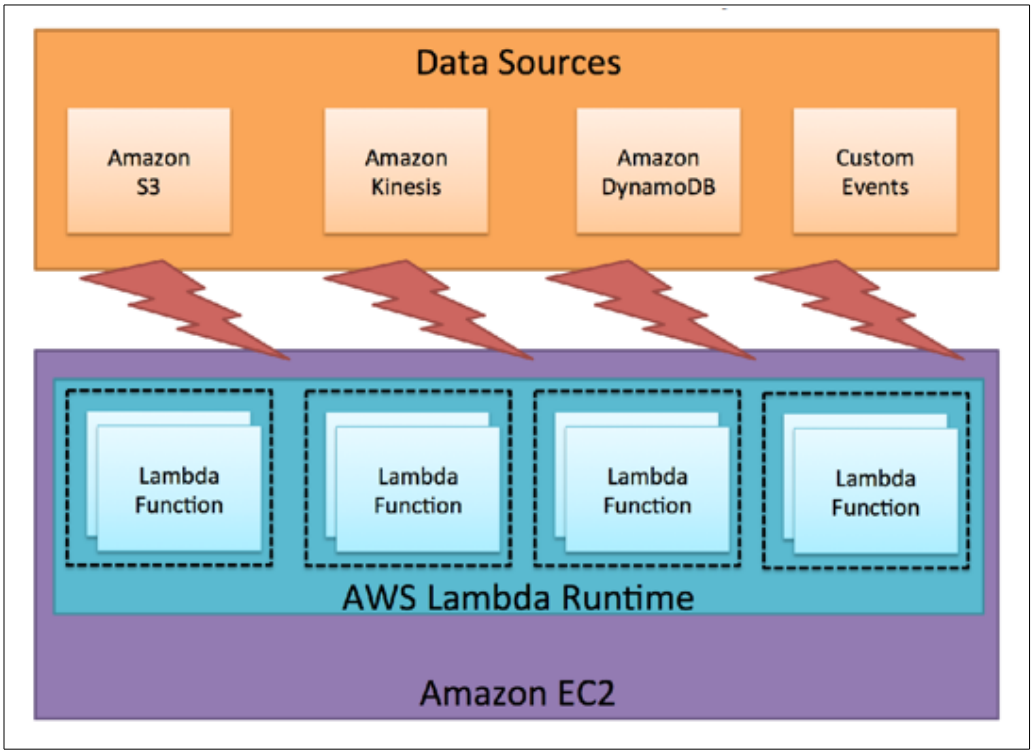
AWS Lambda gehört zum Anbieter Amazon und wurde 2014 vorgestellt. Eine nähere Suche im Internet zum Thema Serverless Programming zeigt, dass AWS Lambda häufig im Zuge von Artikeln verwendet wird. Ebenfalls existieren viele Tutorials dazu, so dass hier eine große Verbreitung dessen zu erkennen ist. Es 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 Programming 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. Zwar hat AWS Lambda nicht wie die drei folgenden Plattformen eine eigene Benutzeroberfläche, mit deren Hilfe Dokumente erstellt werden können, jedoch bietet die Tatsache, dass mit regulären Entwicklungsumgebungen gearbeitet wird den großen Vorteil, dass sich die Entwickler mit dieser auskennen und so effektiver arbeiten können, da sie auch eine Vielfalt an Funktionen und Einstellungen bietet.

AWS Lambda unterstützt Java, Node.js und Paython 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. So ist AWS Lambda nicht eingeschränkt in seiner Verwendung.

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 Programming.



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 Programming 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 Programming 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 also mit einer Plattform beschäftigt hat, der wird sich aller Wahrscheinlichkeit nach nicht auch noch mit einer anderen auseinandersetzen.

6. Erläuterung des Prototypen

Im Zuge der Ausarbeitung soll ein Prototyp erstellt werden, der die Vorteile des Serverless Programming aufzeigt und die Möglichkeiten der Umsetzung einer Anwendung veranschaulicht.

Die Anforderung 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 Programming zu zeigen.

Weiterhin soll der Prototyp so aufgebaut sein, dass es die Möglichkeit gibt für die nachfolgende Perfomancemessung 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. So soll das Gerüst dieser ebenso wie eventuelle Benachrichtigungen so gestaltet werden.

7. Erläuterung Erarbeitung des Prototypen

xxx

8. Performancemessungen

xxx

9. Ergebnis

xxx

10. Vor - und Nachteile Serverless Programming

Der Bereich des Serverless Programming bietet einige Vorteile im Vergleich zu den bisherigen Möglichkeiten. Aber wie bei fast allem existieren im Gegenzug dazu auch Nachteile. Beide Seiten sollen nun im Folgenden näher betrachtet werden. Dies soll insbesondere im Hinblick auf die zuvor ausgearbeiteten Informationen und die Durchführung des Prototypen erfolgen.

10.1 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 Programming 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 Programming 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. Beim Serverless Programming 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 Programming 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 Programming 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 Programming 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 Programming 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 Programming 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 Programming mit verschiedenen Methode gearbeitet werden. So können die unterschiedlichsten Projekte umgesetzt und gestaltet werden, je nach Belieben des Unternehmens oder des Kunden.

10.2 Nachteile

Einer der größeren Nachteile des Serverless Programming 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 Programming. 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 Programming 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 Programming 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 Programming 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 Programming 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.

11. Fazit

xxx

12. Ausblick in die Zukunft

xxx