Moderne WebApps aus Basis von RESTful Web-Services

Ahmad Abou Altout

Zusammenfassung
In diesem Artikel werden Ihnen über SPA Single-Page-Architektur, ihr Ansatz und RESTfull Web-Services, die beste Webapp betrachtet werden, informieren, sie hat in WebApps große bewirken.

Keywords
Single-Page Application, RESTfull Web Services

Hochschule Kaiserslautern
Corresponding author: ahab0004@stud.hs-kl.de

Inhaltsverzeichnis

Einleitung

In der Vergangenheit, bevor sich Browser und Javascript heute mit ihnen entwickelten, stammte der Inhalt aller Webseiten vom Server als HTML, und der Browser las nur diesen HTML-Code und zeigte ihn dem Benutzer an. Dieser Vorgang wird jedes Mal wiederholt, wenn wir auf einen bestimmten Link klicken aber heutzutage mit der Entwicklung die Programmiersprache Javascript, PHP und der verbesserten Browser Leistung sind heute die meisten bekannten Webanwendungen Single-Page-Architektur oder zumindest ein großer Teil davon. Also Moderne Anwendungen gibt es in vielen Formen, sie unterstützen mehrere Plattformen und installieren unzählige Gerätetypen. Diese Anwendungen verpflichten ihre Programmierer, zusätzlich zu den herkömmlichen Webseiten mehrere und unterschiedliche Clients wie Smartphones und Tablets zu unterstützen. Der Entwickler kann viel Aufwand und Zeit sparen, sei es während der Entwicklung oder Wartung, er muss die Logik der Programmarbeit an einem Ort sammeln und diese Logik mit anderen teilen Clients, die auf verschiedenen Plattformen arbeiten. Die beste moderne Methode, die der Entwickler befolgen sollte, besteht darin, eine Programmierschnittstelle für seine Idee zu erstellen, um den Clients mit seinem unterschiedlichen Programm eine gemeinsame Plattform verfügbar zu sein und die Verwaltung dieser Plattform zu vereinfachen und sie weniger mühelos zu warten.

1. Moderne Single-Page-Architektur

Single-Page Anwendungen vermitteln dem Benutzer den Eindruck, eine vollwertige Anwendung zu bedienen. Single Page-Anwendungen mit einem REST-Backend sind ein sehr beliebter Architekturstil. In Single-Page-Architektur laden Browser HTML-, CSS- und Javascript-Codes nur einmal. Anstatt darauf zu warten, dass der Server das Ganze ausführt, fragen wir (als Clients) nur nach den gewünschten Daten (z. B. Json-Format), und der Browser gezeigt der gewünschte Angeordnet (im HTML-Format) für den Benutzern an. Früher zeigte der Browser nur seinen HTML-Code an, aber jetzt generiert er diesen HTML-Code und bestimmt seine Struktur über Javascript, Aber nicht direkt, sondern eine ihre Bibliotheken oder ein damit erstelltes Framework wird verwendet, zum Beispiel Knockoutjs, Emberjs, BackboneJS, AngularJs und viele andere. Hier ist ein Link, der einige davon erklärt.[1]anschließend wird die Benutzeroberfläche schnell und ohne die Seite neu zu laden also ohne Postback. Die Technologie zum Anfordern und Senden von Daten zum und vom Server ohne die Seite neu zu laden – heißt Asynchrones Javascript und XML und wird einfach als Ajax bezeichnet.[2] frühen bei Anfängen von Ajax waren es nur die Unternehmen, die Entwickler mit einer guten Leistungsfähigkeit und hohen Kompetenz hatten, die nur Single-Page-Application aufbauen können. Mit dem Auftreten Javascript-Frameworks wird diese Arbeit für alle Webentwicklern verfügbar, da sie eine starke Plattform und Grundlage bereitstellen, damit die Webentwicklern starke Anwendung ohne viele Komplexität in ihrer Architektur erstellen. Erfolgreiche und bekannte Single-Page-Architektur Webanwendungen sind zahlreich und es ist nicht möglich, sie alle zu erwähnen, aber wir erwähnen nur einige von denen, die man jeden Tag benutzen: Gmail, Google Maps, Twitter etc.

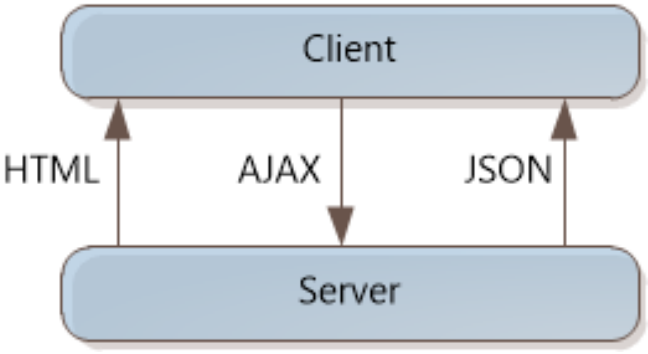

Abbildung 1. Beispiel für AJAX

1.1 Microprofile

Während Java EE eine solide Grundlage für die Erstellung von Microservices bietet, wurden Technologien und Programmiermodelle benötigt, um besser für Microservices-Anwendungen geeignet zu sein. IBM® und andere Unternehmen haben zusammengearbeitet, um MicroProfile[3] zu starten, eine offene Zusammenarbeit zwischen Entwicklern, der Community und Anbietern. Die microprofile.io-Community widmet sich schnellen Innovationen um Microservices und Enterprise Java. Diese Community erstellt und integriert Technologien, die am besten für native Java Cloud-Anwendungen geeignet sind, die den Architekturmustern von Microservices folgen. Mitarbeiter identifizieren, implementieren, verfeinern und verbessern Technologien, die üblicherweise in leichten Mikrodiensten verwendet werden. Mitarbeiter zeigen auch, wie diese Technologien in verschiedenen Laufzeitumgebungen verwendet werden. Innerhalb der microprofile.io-Community definiert jede MicroProfile-Version eine Reihe von Technologien zu einem bestimmten Zeitpunkt im Verlauf der Zusammenarbeit. Microservice-Anwendungen zeichnen sich in der Regel dadurch aus, dass viele Services, die jeweils in eigenen Prozessen laufen, miteinander interagieren und so ein perfekter Ablauf ergeben. Oft werden diese Services zusätzlich in Containern verpackt und in der Cloud bereitgestellt, also wir verfahren mit einem hochdynamischen, und stark verteilten System. MicroProfile 3.3 Abbildung 2 ist die 13. Plattformversion für das Eclipse MicroProfile-Projekt. Basierend auf dem time-boxed Release-Prozess von MicroProfile handelt es sich um eine inkrementelle Version, die ein Update auf MicroProfile Config 1.4, MicroProfile Fault Tolerance 2.1, MicroProfile Health 2.2, MicroProfile Metrics 2.3 und MicroProfile Rest Client 1.4 enthält. Die (CDI 2.0,JSON-p1.1, JSON-B1.0 und JAX-RS2.1)dargestellten Spezifikationen wurden von der Java-EE-8-Spezifikation[4] übernommen und sind mit dieser identisch (Abb.2). So verwendet MicroProfile ebenso wie Java-EE 8 eine Implementierung von CDI für die Dependency-Injection. Die(JWT Propagation 1.1, OpenAPI 1.1, OpenTracing 1.3,Rest Client 1.4,Config 1.4,Fault Tolerance 2.1,Metrics 2.3,Health 2.2)dargestellten Spezifikationen wurden speziell für MicroProfile [5] entwickelt. Diese acht speziellen Spezifikationen unterscheiden sich von ihren Java-EE-Pendants unter anderem dadurch, dass es für sie keine Referenzimplementierungen gibt.

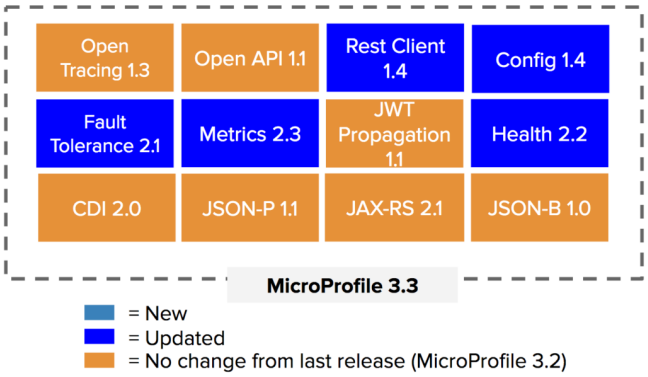

Abbildung 2

1.2 Single-Page Ansatz

Natürlich Clients brauchen schnelle Ladung von Webseiten und schnellste Methode, um Daten von Datenbank zu bringen. Und auf andre Seite benötigen die Programmierern-Webseite Architektur, mit der nicht komplex verfahren. Also alle diese Wünsch findet man in Single-Page-Architektur, die viele Vorteile noch hat. Einige Vorteile SPA-Anwendungen:

- Der Druck auf den Server wird reduziert, da es nur übernimmt, Daten vom Kunden über sogenannte Software-Schnittstellen oder APIs abzurufen oder abzurufen, und es ist ihm egal, wie diese Daten dem Benutzer präsentiert werden

- Single-Page-Architektur Webanwendungen sind sehr praktisch, um als Team zu arbeiten, wobei sich der Backend-Webentwickler auf die Entwicklung von Frontend-Webentwicklern konzentriert, während sich der Frontend-Entwickler darauf konzentriert, diese Programmierschnittstelle zu nutzen, um eine Benutzeroberfläche mit einer hervorragenden Benutzererfahrung zu erstellen.

Die Nachteile oder Herausforderungen von SPA:

- Die wichtigste Herausforderung bei SPA ist die Kompatibilität mit Suchmaschinen und die Möglichkeit, diese Seiten auf Social Media-Plattformen zu teilen. Als Reaktion auf diese Herausforderung wurde in den letzten Jahren der Begriff Server Side Rendering[6], der als Einführung von HTML-Seiten vom Server beim ersten Eintrag in die Anwendung bezeichnet wird, in dem Begriff verwendet.

- Ein großer Teil der einseitigen Webanwendungen besteht aus Javascript, und häufig wird eine große Javascript-Datei geladen. Wenn Sie lange auf der Seite bleiben und die Benutzer intensiv damit interagieren, können einige Probleme im Zusammenhang mit einem übermäßigen RAM-Verbrauch auftreten. Aus diesem Grund ist es

immer gut, vorgefertigte Javascript-Frameworks wie React.js oder Vue.js zu verwenden, da sie statt uns Anstrengungen unternehmen, um Lösungen für Probleme zu finden, die sich auf den Aspekt der Leistung und Geschwindigkeit in der Anwendung auswirken.

Single-Page-Architektur Webanwendungen sind normalerweise schneller als herkömmliche Webanwendungen und bieten Benutzern eine bessere Benutzererfahrung, wenn sie gut erstellt sind, weil Sie die Anwendung zum ersten Mal laden, werden nicht alle Seiten vom Server gerendert. Es wird nur index.html geladen, wenn Sie die Anwendung laden. Da nur eine einzelne Seite geladen wird, wird sie als SPA bezeichnet, aber wie wird das rest von daten, die der Client braucht, versorgt oder Aktionen wie (anmelden registrieren)geschehen? Die Daten werden asynchrone versorgt und die Aktionen asynchron im Hintergrund geschehen, wie? [7] die asynchrone Kommunikation zwischen Server und Client durch XMLHttpRequest stellt einen der wichtigsten Teile von AJAX dar. Damit wird es möglich, Bereiche innerhalb der Seite zu aktualisieren, ohne ein vollständiges Neuladen der Seite anzufordern. Das XMLHttpRequest-Objekt stellt die technische Grundlage für die asynchrone Kommunikation mit Webserver oder -diensten innerhalb von AJAX Applikationen dar. Ein Kommunikationsablauf über XMLHttpRequest stellt sich folgendermaßen dar: Erzeugen des XML pRequest-ObjektsÖffnen einer Verbindung zu einem Dienst oder Prozess:

- Erzeugen des XML pRequest-Objekts
- Öffnen einer Verbindung zu einem Dienst oder Prozess
- Senden einer Nachricht über die erzeugte Verbindung
- Starten eines EventListeners zur Überprüfung, ob eine Antwort auf die Anfrage erfolgt ist
- Integration der Antwort in die aktuelle Seite über das DOM
- Wenn das Objekt einmal erzeugt wird, kann es einheitlich verwenden mit dem Anfragen-Typ (POST oder GET) werden.[8]

1.3 RESTfull Web-Services als Backend

Der Begriff REST, eine Abkürzung für Representational State Transfer, drückt die Architektur aus, die bei der Entwicklung von Webdiensten verwendet wird. Ziel ist es, Standards festzulegen, die das Ressourcenmanagement von Systemen steuern und definieren, wie diese adressiert und über HTTP an eine Vielzahl unterschiedlicher Anwendungen übertragen werden, unabhängig von den Programmiersprachen, in denen sie entwickelt wurden. Für Anwendungen war die REST-Architektur in den letzten Jahren die dominierende Webdesign-Architektur, um die Verwendung und Handhabung zu vereinfachen. Aber was versteht man unter Backend?[9] Alle Anwendung besteht im Allgemeinen aus zwei Teilen: dem Frontend und dem Backend. Viele Menschen verstehen die Konzepte aufgrund Unklarheit. Nehmen wir ein Beispiel, Sie sitzen in einem Auto und verwenden Bremsen, Kupplung, Gaspedal, Gang usw. Diese werden als Frontend-Komponenten bezeichnet. Aber alle Aktionen, die hinter den Kulissen stattfinden, die Sie nicht kennen. Dies ist als Backend bekannt. Die große Rolle, die Backend spielt, dass Sie den Mechanismus der Anwendung verstecken, das bedeutet, dass ein Benutzer die Datenbanken und vertraulichen Inhalte nicht sehen kann, aber wenn es eine Sicherheitslücke gibt, dann er kann. Der Restful ist so aufgebaut, dass es dem Prinzip des Webdienstes entspricht, bei dem die Anwendung aus einer Gruppe von Diensten besteht, die einzeln arbeiten oder verknüpft werden können. Der Verständnisprozess zwischen dem Client und dem Server oder der Ressource wurde durch Senden von XML-Dateien mit bestimmten Typen durchgeführt, und diese Typen wurden als Soap Messages bezeichnet. Mit dem Auftreten des Restful-Prinzips können wir sagen, dass die Soap Messages durch Restful-Prinzips aufgrund ihre Komplexität und der Notwendigkeit Informationen, die durch XML auszutauschen, und den Ressourcenverbrauch erhöht, indem es sich vollständig auf HTTP verlässt, ersetzt wird. Einer der Vorteile dieses Prinzips besteht darin, dass Operationen in Abhängigkeit von den folgenden Anforderungstypen (Requests)[10] angewendet werden:

- GET: Elemente aus der Quelle abrufen (Anzeigevorgang)
- POST: neue Elemente hinzufügen(Hinzufügung)
- PUT: ein Element bearbeiten (den Bearbeitungsprozess)
- DELETE: Elemente löschen (Löschvorgan)

[11] Ein kleines Beispiel Stellen wir uns einen fiktiven Webservice vor, der Zugriff auf eine Produkt-Datenbank bietet. Der Einstiegspunkt für den Webservice soll die URI sein. Ein GET-Request an diese URI sollte also eine Liste aller Produkte zurückliefern:

```
GET /products HTTP/1.0
Accept: application/json
```

Listing 1. Ein Request

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 1234

[ { uri: "http://ws.mydomain.tld/products
    /1000",
name: "Gartenstuhl",
price: 24.99 },
{ uri: "http://ws.mydomain.tld/products/1001
    ",
name: "Sonnenschirm",
price: 49.99 } ]
```

Listing 2. Ein Response:

Client schickt Request und der Server gibt Response zurück am wichtigste Punkt bei Ablauf ist, dass Jede Request alle erforderlichen Informationen vom Client zu der Quelle enthält. Daher werden keine anderen Informationen übertragen, es ist beispielsweise nicht zulässig, die Informationen in der Session zu speichern, da dies gegen das Prinzip Restful verstößt wird. Es gibt eine Beziehung zwischen dem Client und der Quelle, daher müssen Anforderungen Staatenlos(Stateless) sein, das bedeutet, dass kein Fall beibehalten wird. Jedes Mal müssen alle Daten in der Request und Response übertragen werden. Vorteile[12]

- Eine vollständige Trennung zwischen Client- und Serverebene
- Responses können gespeichert werden und Client verlässt sich auf diesen Informationen als (Offline-Modus) wie Beispiel(Gmail)
- Die Leichtigkeit der übertragenen Daten, da sie von HTTP abhängig sind,
- Die Quelle hängt nicht vom Client ab, daher kann der Client ein beliebiges Gerät, Programm oder etwas anderes sein

Es gibt einige Nachteile:

- Sie benötigen zusätzliche Arbeiten, um den Schutz zu gewährleisten, da dies von Request abhängt.
- Sie benötigen ein erläuterndes Dokument der sendenden Daten und Referenzen, um zu wissen, wie Sie damit umgehen sollen. Jedes Mal müssen alle Informationen entweder vom Client oder
- vom Server gesendet werden.

2. Beispiel Single-Page Anwendungen

Ein Rateme soll als Beispiel für eine Single-Page Anwendungen in Reteme kann jeder Bewertung von aller Restaurants Zweibrücken und Homburg sehen, und wenn man registriert hat, dann kann man selber Bewertung geben und seine Bewertungen werden in Data-Bank gespeichert und für andre Gäste gezeigt. Wenn man auf einem Icon des Restaurants, der Browser bleibt auf derselben Webseite, aber der JavaScript-Code zeigt die Informationen und Bewertungen des Restaurants ohne die Seite neu zu laden. Dafür benutzt man Restfull Architektur, der vorher erklärt wurde, und mit der HTTP Methoden kann man wie unten im Beispiel Bewertung zeigen oder hinzufügen.

```
Get /rateme/rating/Restaurant-Id
```

Listing 3. Get Anfrage

Dann hier wird ein Request gestellt und der Server gibt Response mit aller Bewertungen von Restaurant-Id als JSON zurück

```
{
"name": "mens Hochschule",
"bewertung": "essen war gut",
}
```

Listing 4. Json

Dann kann man mit B.z javascript die Ergebnisse bearbeiten und in HTML Seite zeigen. Die folgende POST Anfrage fügt Bewertung hinzu

```
Post / rateme/ rating/Restaurant-Id / user-
    Id/ bewrtung/
```

Listing 5. Post Anfrage

Kürze kann man erklären, dass der Entwickler in seinem Projekt ein API (Application Programming Interface), die als Vermittler zwischen Client und Server, hat. Unter API kann man Klasse Controller machen, damit er es, was zum und vom Server geschickt oder gebracht werden will, steuern kann. Er kann auch die Klasse Controller verteilen, damit er die Fehler schnelle korrigieren kann, oder wenn er in Gruppe arbeitet, die Arbeit verteilen können. Zur Bearbeitung der HTML Seite und zur Verbindung mit dem Vermittler (API) kann man Javascript, PHP etc. benutzen.

3. Alternative Ansätze

Es ist wichtig auch, dass man alternative Ansätze kennenlernt. Eine der wichtigsten konkret den Architekturansatz einer klassischen Web-Anwendung ist Sevlets. Sie sind Programme, die auf dem Server ausgeführt werden und als mittlere Schicht zwischen Anforderungen vom Browser oder Client zu HTTP-Server. Servlet wird zum Erstellen interaktiver Webanwendungen (Dynamic Web Application) verwendet. Sevlet nimmt Anforderungen vom Benutzer entgegen und verarbeitet sie auf dem Server (siehe Abbildung 3). Mit dieser Technologie können Sie auch mit Datenbanken arbeiten, Informationen speichern und Seiten (html, json, xml) generieren. [13] Um auf eingehende Anfragen zu reagieren, die doGet() und doPost() sowie ggf. auch doPut() und doDelete() Funktionen werden immer dann aufgerufen, wenn eine Anfrage zum Servlet mit der jeweiligen HTTP-Methode gesendet wird, d. h. sendet man eine GET-Anfrage an ein Servlet, so wird dessen Funktion doGet() aufgerufen.[14] einige Vorteile:

1. Es arbeitet mit sehr hohem Schutz im Bereich der Anwendungen, es ist schwer zu durchdringen, da es von Java Security Manager abhängt.
2. Es bietet volle Unterstützung von Java und eine gute Kommunikation zwischen anderen Java-Sprachtechnologien wie JDBC und Applet.

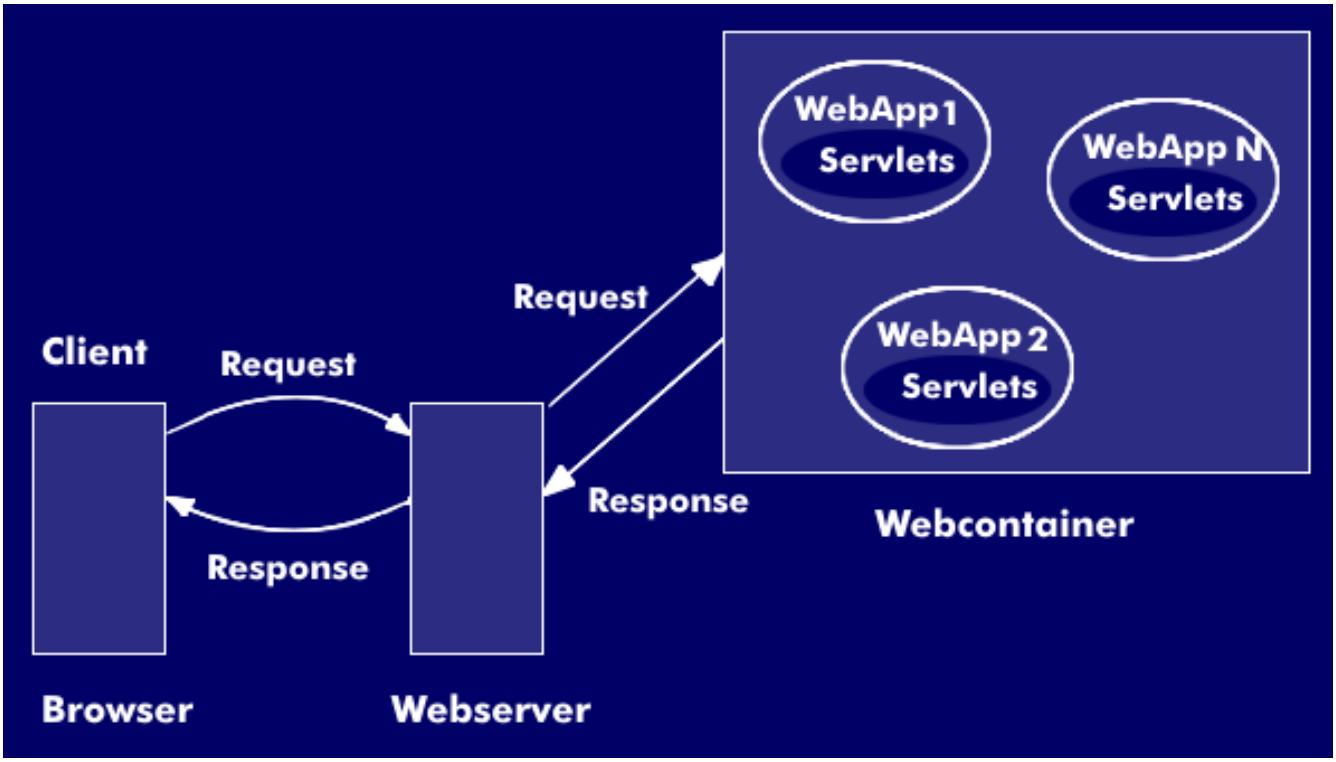

Abbildung 3. kommunikationsstruktur-client-servlet

3. Es funktioniert auf allen Plattformen, da es platform-independent ist und auf Java basiert, das mehrere Plattformen unterstützt.

4. Es zeichnet sich durch eine hohe Leistung aus, da es die Prozesse und Anforderungen von Benutzern als Prozess aufteilt und sie in Threads unterteilt, die gleichzeitig auf Benutzer antworten.

Vergleich von Rest und Servlet

1. Der RESTful-Webdienst kann Servlets als Implementierung verwenden, aber umgekehrt ist dies nicht der Fall.

2. Servlets können nur im Servlet-Container ausgeführt werden, RESTful-Services können jedoch auch im Webcontainer ausgeführt werden.

3. Wenn Sie die RESTful-Services mit anderen Teams / Clients / der Öffentlichkeit teilen möchten, um sie zu nutzen, gibt es zahlreiche Dokumenttools, mit denen Dokumentation für Ihre REST-Services erstellt werden kann (z. B. Swagger). Diese Art von Dokumentationstools finden Sie bei Servlets jedoch nicht.

4. Servlets sind Java-spezifisch, RESTful-Webdienste jedoch nicht.

4. Zusammenfassung

Mit dieser Entwicklung in Webapp und Javascript auf den verschiedenen Ebenen wird am wichtigsten, dass die Entwickler alle ihre Aufwand in richtige Architektur investieren, deswegen habe ich versucht, auf Moderne WebApps aus Basis von RESTful Web-Services hervorzuheben. Also wie wir gesehen haben, dass SPA verschiedene arten von Anwendungen und Kunden gleichzeitig unterstützt und dient, das heißt, das eine Server kann unsere Webapp, Handy App oder PC App unterstützen. Man muss eigentlich nicht vergessen, dass Resstfull Architektur große Rolle gespielt, indem die HTTP Methode (GET,POST ,PUT,DELETE) benutzen kann und die Responses gespeichert werden kann. Also der Client verlässt sich auf die Information, die als(Offline Mode) gespeichert werden, wie Gmail. Es ist klar, dass die Alternative Ansätze der anderen wie (JEE mit HTML, JavaScript, Sevlets, JSP), die viele Vorteile und Unterstützung haben, geschätzt werden.

Literatur

[1] other-libraries. `http://www.asp.net/single-page-application/overview/introduction/other-libraries`,Zuletzt abgerufen am 26. Juni 2020.

[2] Ajax. `https://docs.microsoft.com/en-us/aspnet/single-page-application/overview/introduction/`

knockoutjs-template,Zuletzt abgerufen am 30. Juni 2020.

[3] Microprofile. https://microprofile.io/, Zuletzt abgerufen am 26. Juni 2020.

[4] Java ee specifications. https://javaee.github.io/javaee-spec/Specifications,Zuletzt abgerufen am 26. Juni 2020.

[5] microprofile-specifications. https://download.eclipse.org/microprofile/microprofile-3.3/microprofile-spec-3.3.html,Zuletzt abgerufen am 30. Juni 2020.

[6] Server-side-rendering. https://blog.searchmetrics.com/de/javascript-seo-server-side-rendering/,Zuletzt abgerufen am 30. Juni 2020.

[7] Vildan Softic Manfred Steyer. *Angular JS: Moderne Webanwendungen und Single Page Applications mit JavaScript*. O'Reilly Media, 2015.

[8] Jan Zahalka. *Web Engineering für asynchrone Anwendungen*. diplom.de, 2006.

[9] Backend. https://medium.com/techloop/an-introduction-to-backend-development-and-rest-apis-b1a1a978821f,Zuletzt abgerufen am 29. Juni 2020.

[10] Mark Masse. *REST API Design Rulebook*. O'Reilly and Associates, 2011.

[11] Restful. https://www.mittwald.de/blog/webentwicklung-design/webentwicklung/restful-webservices-1-was-ist-das-uberhaupt,Zuletzt abgerufen am 29. Juni 2020.

[12] Roy Thomas Fielding. *REST: Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

[13] Servlet. https://www.itwissen.info/Servlet-servlet.html,Zuletzt abgerufen am 30. Juni 2020.

[14] Schnittstellen. https://www.homepage-webhilfe.de/JavaEE/Servlet/,Zuletzt abgerufen am 30. Juni 2020.