|  |
| --- |
| Wintersemester 2019/2020 |
| Visualisierungen für IT-Konsolidierungsprojekte |
|  |
| Thesis  zur Erlangung des Grades  Bachelor of Science  im Studiengang WirtschaftsNetze  an der Fakultät Wirtschaftsinformatik  der Hochschule Furtwangen University |
| vorgelegt von  Katharina Schemel |
| Referenten: Prof. Dr. Marianne Andres  Martin Jelen  Eingereicht am 17. Februar 2020 |

|  |
| --- |
| Eidesstattliche Erklärung  Ich Katharina Schemel erklären hiermit an Eides statt, dass ich die vorliegende Bachelorthesis selbständig und ohne unzulässige fremde Hilfe angefertigt habe.  Die verwendeten Quellen sind vollständig zitiert.  Furtwangen, den 17. Februar 2020  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Inhaltsverzeichnis

[1 Einleitung 1](#_Toc32528352)

[1.1 Motivation 2](#_Toc32528353)

[1.2 Ziel der Arbeit 3](#_Toc32528354)

[2 Grundlagen 5](#_Toc32528355)

[2.1 Enterprise Architecture Management 5](#_Toc32528356)

[2.2 Visualisierungskonzepte 8](#_Toc32528357)

[2.3 Funktionsweise einer Graph-Datenbank 12](#_Toc32528358)

[2.4 Cypher 14](#_Toc32528359)

[2.5 Node.js 16](#_Toc32528360)

[2.6 Webserver 16](#_Toc32528361)

[2.7 Webtechnologien zur Visualisierung (Überdenken ob passend) 18](#_Toc32528362)

[3 Konzeption 19](#_Toc32528363)

[3.1 Infrastrukturaufbau 19](#_Toc32528364)

[3.2 Datenmodellkonzept 20](#_Toc32528365)

[3.3 Visualisierungskonzept für Graphen 22](#_Toc32528366)

[3.4 Umsetzungsmöglichkeiten für künftige IT-Systementwürfe 26](#_Toc32528367)

[3.5 Konzeption der Entwicklungsumgebung 29](#_Toc32528368)

[4 Implementierung 32](#_Toc32528369)

[4.1 Verwendete Technologien 33](#_Toc32528370)

[4.2 Vorgehen bei der Visualisierung 34](#_Toc32528371)

[4.3 Interpretation und Ergebnisse 34](#_Toc32528372)

[5 Evaluation (Fazit) 42](#_Toc32528373)

[6 Zusammenfassung und Ausblick 44](#_Toc32528374)

[6.1 Zusammenfassung 44](#_Toc32528375)

[6.2 Ausblick 45](#_Toc32528376)

[Glossar 48](#_Toc32528377)

[Anhang 49](#_Toc32528378)

**Abbildungsverzeichnis**

[Abbildung 1: Brücke zwischen Business und IT (Business-IT) 6](#_Toc32528379)

[Abbildung 2: Bebauungsplan Grafik 9](file:///D:\Thesis\Datenmodell\Thesis_Abschlussarbeit.docx#_Toc32528380)

[Abbildung 3: Funktionales Referenzmodell 10](file:///D:\Thesis\Datenmodell\Thesis_Abschlussarbeit.docx#_Toc32528381)

[Abbildung 4: Portfolio-Grafik 11](file:///D:\Thesis\Datenmodell\Thesis_Abschlussarbeit.docx#_Toc32528382)

[Abbildung 5:Zusammenhang Entitäten-Beziehungsmodell, Relationenmodell, Graphenmodell 13](#_Toc32528383)

[Abbildung 6: Ergebniss Relationenabfrage 15](file:///D:\Thesis\Datenmodell\Thesis_Abschlussarbeit.docx#_Toc32528384)

[Abbildung 7: Ergebniss Relationenabfrage mit Einschränkungen auf Attributwerte 15](file:///D:\Thesis\Datenmodell\Thesis_Abschlussarbeit.docx#_Toc32528385)

[Abbildung 8: Client-Server Modell 17](#_Toc32528386)

[Abbildung 9: Infrastrukturkonzept 19](#_Toc32528387)

[Abbildung 10: Datenmodell 20](#_Toc32528388)

[Abbildung 11: Mockup Grad der Vernetzung 23](file:///D:\Thesis\Datenmodell\Thesis_Abschlussarbeit.docx#_Toc32528389)

[Abbildung 12: Mockup Beziehung zu 23](file:///D:\Thesis\Datenmodell\Thesis_Abschlussarbeit.docx#_Toc32528390)

[Abbildung 13: Mockup Ampelsystem 24](file:///D:\Thesis\Datenmodell\Thesis_Abschlussarbeit.docx#_Toc32528391)

[Abbildung 14: Mockup Nutzeranzahl 24](file:///D:\Thesis\Datenmodell\Thesis_Abschlussarbeit.docx#_Toc32528392)

[Abbildung 15: Technologie Radar 26](#_Toc32528393)

[Abbildung 16: Konzept der Entwicklungsumgebung 29](#_Toc32528394)

[Abbildung 17: Windows PowerShell 36](#_Toc32528395)

[Abbildung 18: Neo4j Datenbankauszug 36](file:///D:\Thesis\Datenmodell\Thesis_Abschlussarbeit.docx#_Toc32528396)

[Abbildung 19: Visual Studio Code 37](#_Toc32528397)

[Abbildung 20: Notepad++ 38](#_Toc32528398)

[Abbildung 21: Excel Daten 39](#_Toc32528399)

[Abbildung 22: Windows PowerShell 40](file:///D:\Thesis\Datenmodell\Thesis_Abschlussarbeit.docx#_Toc32528400)

[Abbildung 23: Daten Webansicht 41](#_Toc32528401)

# Einleitung

In der Realität ist es oft so, dass innerhalb eines Unternehmens viele verschiedene IT-Systeme verwendet werden, welche schnell schwierig zu überblicken sind. Diese Systeme beinhalten Programme, welche auch gepflegt werden müssen. Dies resultiert beispielsweise daraus, dass diese immer auf dem aktuellen Stand sein sollten, um Sicherheitslücken zu vermeiden. In großen Unternehmen hat man kaum einen Überblick, welche IT-Systeme in den einzelnen Abteilungen verwendet werden. Hier empfiehlt sich eine Konsolidierung vorzunehmen. Das bedeutet, dass die IT-Systeme sozusagen verschlankt werden.

Ein Aspekt der Konsolidierung ist die Ineffizienz von IT-Systemen zu eliminieren. Oft werden verschiedene Systeme verwendet, welche im Prinzip, das exakt gleiche tun. Daher wäre es von Vorteil eine einheitliche Struktur festzulegen, um den Bestand der Software zu vereinfachen.

Ein weiterer Aspekt bei der Konsolidierung ist die Entscheidung, welche Komponenten konsolidiert werden sollten. Zu beachten ist jedoch, dass durch die Konsolidierung nie eine konkrete Entscheidung gefällt wird welche Komponenten konsolidiert werden, sondern diese lediglich eine Entscheidungsgrundlage darstellt. Der Anstoß für eine IT-Konsolidierung kann aus aktuellen Anlässen erfolgen, strategischen Nutzen für das Unternehmen bieten oder einfach von der Unternehmensführung gewünscht sein.

All diese Gründe führen zu der Konsolidierung, um beispielsweise den Geschäftswert bzw. die Betriebskosten des Unternehmens zu optimieren oder zur Zuordnung von Geschäftsprozessen, welche zu Outsourcing führen können. Somit ist deutlich zu erkennen, dass die Konsolidierung ein wichtiger Aspekt in Unternehmen darstellt.

Das Problem bei der Konsolidierung ist jedoch, die einzelnen Komponenten zu identifizieren. Eine Datenbeschaffung ist recht zeitaufwendig und kann fast nie vollständig erfolgen. Hieraus entsteht meist ein Vorgehen auf Basis von Teilbeständen, was letztlich nicht dem Ideal entspricht.

Die Komponente Visualisierung wird sowohl für die Präsentation der Analyseergebnisse wie auch als Werkzeug für die Analyse eingesetzt. Um die Analyse zu verbessern ist es vorstellbar Graphdatenbanken in diesen Prozess mit einzubinden, da ein Graph schnell einen Überblick über Beziehungen und Zusammenhänge der einzelnen Komponenten vermittelt. Daraus kann eine Entscheidungsgrundlage gebildet werden, welche Komponenten entsprechend eliminiert oder verändert werden sollten.

## Motivation

Die Thematik der IT-Konsolidierung ist noch recht jung und sehr aktuell, beispielsweise hat die IT-Konsolidierung des Bundes erst 2015 begonnen. Daher ist es auch sehr spannend diese Thematik weiter voran zu bringen. Auslöser für eine IT-Konsolidierung kann z. B. eine vorherige Fusion darstellen. Laut (Bundesministeriums des Innern, für Bau und Heimat, 2020) sind die folgenden Punkte Ziele einer IT-Konsolidierung:

* Informationssicherheit trotz steigender Komplexität zu gewährleisten
* die eigene IT soll zu jederzeit Souverän und beherrscht sein
* flexibel auf innovative technologische Trends reagieren zu können
* ein zukunftsfähiger Betrieb, welcher leistungsfähig und stabil ist
* für das IT-Fachpersonal ein attraktiver Arbeitgeber bleiben

IT-Konsolidierung: Eine einfache Definition

*„Konsolidierung bedeutet einfach übersetzt „Zusammenführung“. Im Bereich der IT sollen also Infrastrukturen, Datenbestände und Anwendungen zusammengeführt und idealerweise auch vereinheitlicht werden. Dadurch werden Kosten gespart, Abläufe vereinfacht sowie beschleunigt und die IT-Qualität erhöht sich insgesamt.“* (Klein, 2017)

Diese Definition zeigt, das IT-Konsolidierung sehr wichtig ist, da neben einer Kosteneinsparung auch die Qualität der IT erhöht werden soll, was insgesamt eine Bereicherung für die heutige Zeit darstellt, in welcher die Themen Digitalisierung und Industrie 4.0 immer allgegenwärtiger werden. Auch die Zusammenführung von Infrastrukturen, Datenbeständen und Anwendungen erleichtern das tagtägliche Arbeiten, sowie die Pflege der Software erheblich. Die Konsolidierung wird im Enterprise Architecture Management (EAM) durchgeführt, da diese einen konzeptionellen und organisatorischen Rahmen bietet, damit die Architektur zielgerichtet aufgebaut und erweitert werden kann. Zudem kann die Architektur Zusammenhänge z. B. mittels Diagramme sehr gut darstellen. Besonders bei komplexen Sachverhalten ist diese Weise deutlich effizienter.

Die Thematik der Visualisierung im Kontext IT-Konsolidierung ist ein wichtiger Bestandteil dieser Arbeit, da eine Präsentation der Analyseergebnisse mittels Excel-Tabellen, wie es häufig von IT-Beratern praktiziert wird, nur einen geringen Mehrwert bietet.

## Ziel der Arbeit

Das Ziel der Arbeit besteht sozusagen in der „Grundsteinlegung“ auf dem Gebiet der Visualisierung von IT-Konsolidierungsprojekten mit Anbindung an eine Graphdatenbank für die Firma ISB AG. Die Arbeit gliedert sich somit in einen theoretischen wie auch einen praktischen Teil. Der theoretische Teil umfasst die Einarbeitung auf den Themengebieten EAM, Graphdatenbanken und Visualisierung mittels D3. Neben der Einarbeitung stellt sich die zentrale Frage, wie Graphdatenbanken und die Visualisierung auf dem Gebiet der IT-Konsolidierungsprojekte zusammen zu bringen sind. Hierfür sollten prototypische Visualisierungskonzepte erarbeitet werden, unter Berücksichtigung von typischen Visualisierungskonzepten aus dem Bereich des EAM.

Es sei jedoch zu beachten, dass nicht jeder Datensatz, aufgrund seiner spezifischen Eigenschaften, zu jedem Visualisierungskonzept passt. Aus diesem Grund sollten Datenbestände zunächst analysiert werden und dementsprechend eine geeignete Visualisierung aufgrund der Daten gewählt werden. Bezüglich einer Graphdatenbank folgt aus einer derartigen Analyse der Daten in der Regel ein Datenmodell, welches im Zuge dieser Arbeit für eine fiktive IT-Landschaft prototypisch zu erstellen war. Dieses wiederum bildete die Grundlage für die Erstellung der Visualisierungskonzepte.

Die Grundlagenlegung für den praktischen Teil resultierte wiederum in der Erweiterung des theoretischen Teils. Es ging im Zuge dessen darum ein Konzept für eine Infrastruktur für den Betrieb des im praktischen Teil entwickelten Prototyps zu generieren. Weiterhin musste ein Konzept für die Entwicklungsumgebung geschaffen und entsprechend umgesetzt werden, um eine nachhaltige und einfache Organisation des zu erstellenden Softwareprojekts zu gewährleisten. Folglich umfasst das Ziel des praktischen Teils die Entwicklung eines Prototyps, welcher diverse Kernfunktionalitäten exemplarisch implementiert. Diese sind unteranderem die Anbindung an eine Graphdatenbank sowie die Visualisierung der angefragten Daten mittels der JavaScript-Bibliothek D3.

# Grundlagen

## Enterprise Architecture Management

Die folgenden Definitionen von Enterprise Architecture und Enterprise Architecture Management zeigen die konkreten Abgrenzungen voneinander auf. In der Praxis werden diese Begriffe oft als Synonym verwendet, jedoch gibt es gravierende Unterscheidungen zwischen diesen beiden Begriffen.

Definition Unternehmensarchitektur (EA):

*„Eine Unternehmensarchitektur (Enterprise Architecture) schafft eine gesamthafte Sicht auf das Unternehmen. Sie legt die wesentlichen fachlichen und IT-Strukturen fest und verknüpft sie miteinander. Auf dieser Basis lassen sich das Business und die IT und ihre Zusammenhänge beschreiben. Eine gemeinsame Sprachbasis, „eine Brücke“ zwischen Business und IT, wird geschaffen. So kann die strategische Weiterentwicklung von Business und IT aktiv gesteuert werden.“ (Hanschke, 2013)*

Definition EAM:

*„EAM ist ein systematischer und ganzheitlicher Ansatz für das Verstehen, Kommunizieren, Gestalten und Planen der fachlichen und technischen Strukturen im Unternehmen. Es hilft dabei, die Komplexität der IT-Landschaft zu beherrschen und die IT-Landschaft strategisch und businessorientiert weiterzuentwickeln. EAM ist ein wesentlicher Bestandteil des strategischen IT-Managements und beinhaltet alle Prozesse für die Dokumentation, Analyse, Qualitätssicherung, Planung und Steuerung der Weiterentwicklung der IT-Landschaft und der Geschäftsarchitektur.“ (Hanschke, 2013)*

Einfach gesagt, fungiert die Enterprise Architecture als eine Art Vermittlerrolle zwischen der IT und den entsprechenden Fachbereichen eines Unternehmens, so (BITKOM, 2011). Man könnte es auch bildlich als eine Brücke beschrieben, welche diese Bereiche verbindet.



**Business**

**IT**

**EA**

Abbildung 1: Brücke zwischen Business und IT (Business-IT)

Durch diese Verbindung wird ein Rahmen für den Ausbau der IT-Landschaft zur Verfügung gestellt. Dieser Rahmen umfasst laut (BITKOM, 2011) strategische, konzeptionelle und organisatorische Aspekte. Weiter führt (BITKOM, 2011) aus, dass besonders die Methoden der Umsetzung und die Überprüfung im Hinblick einer Kosten-Nutzen-Betrachtung im Fokus stehen.

Das Enterprise Architecture Management wiederum „*umfasst die Aufgaben zur Erstellung, Pflege und Umsetzung einer EA*“ (BITKOM, 2011). Man kann also sagen, dass das EAM ein strukturierter Ansatz für die Erstellung, Verwaltung und Nutzung der von EA bereitgestellten Modelle ist.

Je größer ein Unternehmen wird, desto bedeutender ist die Rolle des EAM. Durch neue Technologien und Schnittstellen steigt die Komplexität und das Bewältigen der IT-Landschaft wird immer schwieriger. Durch fehlendes EAM können Redundanzen und inkonsistente Daten auftreten. Diese Problematik ist unter anderem die Basis für die IT-Konsolidierung. Aber auch mit EAM kann über einen längeren Zeitraum eine IT-Konsolidierung notwendig werden.

Durch das Bereitstellen diverser Hilfsmittel durch EAM, kann die Kontrolle über die IT-Landschaft wiederhergestellt werden und sie kann strategisch und businessorientiert weiterentwickelt werden.

Zudem wird EAM oft eingesetzt, da es Transparenz schafft. Diese ist maßgeblich für die Bewertung der IST-Situation und zeigt bei der IT-Landschaft schon nach kurzer Zeit, in welchem Bereich sich Kosten einsparen lassen.

Laut (Hanschke, 2013) kann in Problemfällen durch entsprechende Visualisierungen schneller Klarheit verschafft werden, z. B. welche Geschäftsprozesse von einem Ausfall des IT-Systems betroffen sind, wo die entsprechenden Verantwortlichkeiten liegen oder welche Abhängigkeiten zwischen den IT-Systemen bestehen.

Um die Leistungsfähigkeit der IT-Leistung einordnen zu können, werden in der Praxis oft relative Bewertungen wie z. B. sehr hoch, hoch, mittel und gering verwendet. Diese Bewertung ist einfach und verständlich, damit sie von Fachbereichen, Anwendern und Managern angewendet werden kann. Die Fachbereiche und die Fachfremden Personen sollten jeweils eine solche Bewertung vornehmen, um festzustellen in wie weit sich das Eigen- und Fremdbild unterscheidet und anschließend sollte entsprechend vorgegangen werden, um diese beiden Ansichten anzugleichen, führt (Hanschke, 2013) weiter aus.

## Visualisierungskonzepte

Farben und Muster sprechen den Menschen schneller an als eine einfache Tabelle mit Informationen. Daher ist eine Visualisierung von Daten wichtig, um das Interesse schneller auf relevante Aspekte zu lenken. Bei der Darstellung von Inhalten ist es wichtig sich vorab zu überlegen, welcher Sachverhalt näher dargestellt werden soll und entsprechend können hierfür unterschiedliche Visualisierungsarten mit passenden Eigenschaften herangezogen werden.

Grundsätzlich sei an dieser Stelle darauf verwiesen, dass nicht jede Visualisierungsart für die Darstellung eines beliebigen Sachverhalts geeignet ist. Beispielsweise kann ein Ampel Diagramm sehr gut verwendet werden, um einen Fortschritt in einem Prozess oder die Kritikalität einer Eigenschaft darzustellen. Es ist jedoch nicht geeignet komplexere Beziehungen zwischen Personen, Prozessen oder Softwareprodukten darzustellen.

Auch für den Bereich des EAM gibt es mehrere gängige Visualisierungsarten, welche laut (Hanschke, 2013) die Aussagekraft der Informationen unterstreicht. Auch auf diesem Gebiet gestaltet sich der Einsatzbereich solcher Grafikkonzepte analog zum Einsatzberiech der Ampel Grafik. Sie sind jeweils Anhand ihrer Eigenschaften für die Darstellung gewisser Sachverhaltet besser geeignet und für andere schlechter. Beispiele für Visualisierungsarten aus dem Bereich des EAMs seien gegeben durch folgende Grafikkonzepte:

* Bebauungsplan-Grafik
* Cluster-Grafik
* Portfolio-Grafik

Die drei genannten Grafikkonzepte wurden exemplarisch aus dem Best-Practice-Visualisierungskatalog gewählt. Diese zeigen durch ihre unterschiedlichen Ausprägungen die verschiedenen Möglichkeiten, welche eine Visualisierung im Bereich des EAM leisten kann. Diese drei Grafikkonzepte werden folgend näher erläutert und zur Veranschaulichung nach Vorlage von (Hanschke, 2013, 2016) nachgestellt.

* Bebauungsplan-Grafik

Diese Grafik stellt die Zusammenhänge zwischen Elementen der Unternehmensarchitektur in Form einer Matrix dar. Somit können u.a. Informationssysteme zu Geschäftsprozessen und Geschäftseinheiten in Beziehung gesetzt werden. Durch die flexible Struktur können viele Fragestellungen beantwortet werden. Der Einsatz von Farben und diversen Linientypen unterstreicht diese Visualisierung nochmals. Die Bebauungsplan-Grafik unterteilt sich in drei Unterprunkte. Für das EAM ist besonders die Ausprägung der „typischen“ Bebauungsplan-Grafik interessant, da diese eine verbreitete Form zur IT-Unterstützung des Unternehmens ist. In einem Bebauungsplan der IT-Branche werden laut (BITKOM, 2011) gegenwärtige und zukünftige Infrastrukturen, als auch Anwendungssoftware definiert. Diese sollen die Geschäftsprozesse eines Unternehmens unterstützen.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 2: Bebauungsplan Grafik

Diese Abbildung zeigt die Geschäftsprozesse auf der X-Achse und die Geschäftseinheiten des Vertriebs auf der Y-Achse. Die einzelnen Bebauungselemente in mitten dieser Grafik geben einen guten Überblick, welche Geschäftseinheiten für welche Geschäftsprozesse zuständig sind.

Diese Art der Grafik ist jedoch nicht zwingend kompatibel zu größeren Datenmengen mit einer netzwerkartigen Struktur, da Abhängigkeiten unter den Daten deutlich schwieriger dargestellt werden können. Dies resultiert aus der blockartigen Anordnung, welche ab einer gewissen Datenmenge eine optische Clusterung nur bedingt zulässt.

* Cluster-Grafik

Das Konzept der Cluster-Grafik beschreibt das Aufteilen und Gruppieren von Bebauungselementen, anhand von Eigenschaften, Beziehungen oder definierten Kriterien. Bekannte Ausprägungen der Cluster-Grafik sind z. B. das Fachliche Domänenmodell, welches die Ausprägung einer Prozesslandkarte oder eines funktionalen Referenzmodells hat. Im Fachlichen Domänenmodell werden die Kernstrukturen der Geschäftsarchitektur festgelegt, daher gibt es laut (Hanschke, 2013) innerhalb eines Unternehmens auch nur ein Fachliches Domänenmodell.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 3: Funktionales Referenzmodell

Die Abbildung 3 zeigt ein funktionales Referenzmodell. Dieses Eignet sich bis zu einem gewissen Grad, Abhängigkeiten zwischen Bebauungselementen darzustellen. Dies wird in diesem Fall auf Basis einer verschachtelten Darstellung realisiert. Somit eignet sich diese Darstellungsart, um die Zusammengehörigkeit zwischen Elementen der IT-Landschaft eines Unternehmens darzustellen. Die Grenzen werden jedoch erreicht, sobald Elemente in mehreren Gruppierungen zugehörig sind, da dies maximal durch eine redundante Aufführung kompensiert werden kann. Dies kann jedoch bei größeren Darstellungen wiederum die Lesbarkeit bzw. Interpretierbarkeit der Grafik beeinträchtigen. Es kann somit auch nicht auf einfachem Wege dargestellt werden wie stark ein Softwareprodukt in ein Unternehmen integriert ist.

* Portfolio-Grafik

Bei dieser Grafik können besonders gut sogenannte Wertigkeiten von Bebauungselementen oder auch Strategien für Bebauungselemente visualisiert werden. Ebenso eignet sie sich für die Abbildung von Nutzenpotenzialen oder Risiken. Hierbei kann der Ist-Zustand oder auch der Soll-Zustand bzw. deren Differenz dargestellt werden. In einer Portfolio Grafik können maximal fünf verschiedene Kriterien abgebildet werden. Diese sind jeweils die Achsen, sowie die Größe, Farbe und der Kantentyp der Füllelemente.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 4: Portfolio-Grafik

Durch diese Grafik kann schnell ein Überblick anhand der Positionierung verschafft werden. Neben der Position, gibt auch die Größe der einzelnen Elemente Aufschluss über beispielsweise Kosten. Diese Grafik eignet sich besonders gut, um messbare Eigenschaften oder eine kategorische Einordnung von Elementen darzustellen. Abhängigkeiten zwischen Elementen können maximal über die kategorische Einordung realisiert werden.

## Funktionsweise einer Graph-Datenbank

Jeder ist bereits mit einem Graphen in Berührung gekommen. Sei es bei der Erstellung eines Mindmaps oder beim Skizzieren von Symbolen und Linien auf einer Tafel. Graphen sind sehr einfach, verständlich und vielseitig einsetzbar, nach (Hunger, 2014). Eine der bekanntesten Graphdatenbanken ist Neo4j. Der Name Neo4j leitet sich aus „Network Engine for Objects“ ab. Die Zahl „4“ hat in diesem Zusammenhang keine Verbindung zur Versionsnummer. Das „j“ steht dafür, dass die frühere Java-API entsprechend weiterentwickelt wurde. Zu Beginn war die Bedeutung von „4j“ „für Java“, führt (Trelle, 2017) aus. Mittlerweile kann, neben Java, auch in der Programmiersprache Scala entwickelt werden. Daher ist die Bedeutung „for Java“ nicht mehr ganz aktuell.

Eine Graphdatenbank gehört zu der Gruppe der NoSQL Datenbanken. Dies bedeutet, dass die Datenbank in der Regel einen nicht relationalen Ansatz verfolgt. Jedoch wird die Verwendung von SQL nicht gänzlich ausgeschlossen, da NoSQL für „Not only SQL“ steht.

Eine Relationale Datenbank verwendet Tabellen, welche Spalten und Zeilen für die Speicherung der Daten nutzt. Die NoSQL Datenbank hingegen nutzt für die Organisation der Daten beispielsweise Attribut-Wert-Paare (Properties), Objekte, Dokumente oder Listen und Reihen für die Organisation der Daten. Ein großer Vorteil von NoSQL Datenbanken ist, dass sie dort ansetzen, wo SQL-basierte relationale Datenbanken an ihre Grenzen stoßen. NoSQL Systeme eignen sich besonders gut für große, exponentiell wachsende Datenmengen, um diese performant zu verarbeiten, wie beispielsweise im Bereich von Big Data. Dies resultiert aus der einfachen Unterstützung der Datenbankfragmentierung, welche durch den Verzicht auf ein Datenbankschema und auf die referenzielle Integrität ermöglicht wird. Im speziellen besagt die referenzielle Integrität, dass Datensätze nur auf bestehende Datensätze verweisen dürfen. Entsprechend kann auch nur ein Datensatz gelöscht werden, wenn auf diesen kein anderer Datensatz verweist.

Graph Datenbanken werden nicht tabellarisch geführt, sondern in Form von Labeln organisiert. Diese sind zwar vergleichbar mit den Tabellen der relationalen Datenbanken, haben jedoch den Unterschied, dass diese ohne eine feste Vorgabe der enthaltenen Attribute auskommt. Die Speicherung der Daten erfolgt somit Schemafrei. Dies erlaubt zusätzliche Informationen wie mangelnde Attribute zu einem späteren Zeitpunkt problemlos nachzutragen. Ein implizietes sturkturierndes Schema sei dennoch durch das Graphenmodell innerhalb einer Graphdatenbank gegeben. Das Graphenmodell kann als Äquivalent zum Relationenmodell der Relationalen Datenbanken interpretiert werden. Es resultiert analog zum Relationenmodell aus dem Entitäten-Beziehungsmodell. Dieser Zusammenhang sei mit Abbildung 5 nach (Meier et al., 2016) näher verdeutlicht.

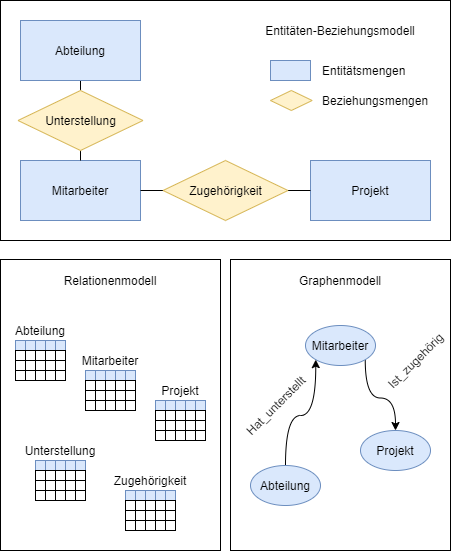


Abbildung 5:Zusammenhang Entitäten-Beziehungsmodell, Relationenmodell, Graphenmodell

Laut (Maier et al., 2016) werden die zu speichernden Daten mithilfe von so genannten Knoten und Kanten dargestellt. Hier werden Objekte in Knoten und Beziehungen zwischen Knoten in Kanten abgebildet. In einem Knoten ist die Beziehung nicht in ihrer Art oder Anzahl beschränkt, ergänzt (Matzer, 2019).

In der Literatur werden Knoten häufig auch als Vertex (V) und Kanten als Edges (E) benannt. Dies sei an dieser Stelle bezüglich der Vollständigkeit erwähnt. Folgend wird diese Terminologie jedoch nicht weiterverwendet.

Die Kante stellt eine Verbindung zwischen den Knoten dar. Diese Verbindung kann als eine Linie oder als ein Pfeil visualisiert werden. Eine Kante ist gerichtet, das bedeutet, dass sie einen Start- sowie einen Endpunkt besitzt. Die Beziehungen zwischen den Knoten können Eigenschaften besitzen. Diese Daten können analog zu den Attributen der Knoten abgefragt werden. Nach (Luber et al., 2017) verzichtet eine Graphdatenbank auf verschachtelte Beziehung, was die hohe Performance für die Speicherung, sowie Abfrage der Informationen erklärt.

Die Information, welche Beziehungen zwischen Daten vorliegen, ist unter gewissen Umständen besonders förderlich. Diese Beziehungen kann ein Graph ausgesprochen gut darstellen. Hieraus ergibt sich der hohe Nutzen der Verwendung einer Graphdatenbank für Daten, welche eine netzwerkartige Struktur aufweisen.

Eine Besonderheit der Graphdatenbanken ist nach (Maier et al., 2016), die Eigenschaft der indexfreien Nachbarschaft. Das Datenbanksystem kann somit die direkten Nachbarn eines Knotens ermitteln, ohne sämtliche Kanten berücksichtigen zu müssen. Dies ist beispielsweise in relationalen Datenbanksystemen nötig, welche sogenannte Beziehungs- oder Verknüpfungstabellen verwenden. Aus diesem Sachverhalt ergibt sich, dass die Abfrage von Beziehungen auf Knoten unabhängig von der gespeicherten Datenmenge sind und somit eine konstante Geschwindigkeit besitzen.

Laut (Matzer, 2019) nutzen bereits große Unternehmen wie z. B. Google, Facebook, Microsoft und LinkedIn die Graph Datenbank, ebenso haben auch Unternehmen wie die NASA oder die Schweizer Bank UBS die Graph Datenbank für sich entdeckt.

Die gängigsten Abfragesprachen für Graph Datenbanken sind Apache TinkerPop Gremlin, SPARQL und Cypher. Da, in dieser Arbeit mit der Graph Datenbank Neo4j gearbeitet wird, betrachtet diese Arbeit folgend nur die Abfragesprache Cypher näher. Cypher wurde von Neo4j entwickelt und erfüllt somit die besten Voraussetzungen was Kompatibilität angeht, nach (Matzer, 2019). Zudem ist Cypher sehr verbreitet und in seiner Struktur der Abfragesprache SQL von Relationalen Datenbanken recht ähnlich. Durch diese ähnliche Struktur ist eine Umstellung von SQL auf Cypher nicht zu zeitaufwendig und komplex. In Kapitel 2.4 wird auf die Abfragesprache Cypher näher eingegangen.

## Cypher

Cypher gehört zu der Gruppe der „deklarativen Abfragesprachen“. Die bekannteste Abfragesprache dieser Gruppe ist SQL. Der Fokus der deklarativen Abfragesprachen ist die Beschreibung eines Problems. Gegenüber den deklarativen Abfragesprachen steht die Gruppe der „imperativen Sprachen“, wie beispielsweise Java oder C++. Nach (Böhm, 2005) ist die imperative Programmierung ein Programmierstil, nach welchem „ein Programm aus einer Folge von Anweisungen besteht, die vorgeben, in welcher Reihenfolge was vom Computer getan werden soll“.

Wird folgend Cypher als problembeschreibende Sprache in Bezug auf Graphen betrachtet, kann von der Beschreibung von Mustern innerhalb von Graphen gesprochen werden. An folgendem Beispiel wird dieser Sachverhalt verdeutlicht. Es wird ein Graph angenommen, bestehend aus zwei Klassen von Knoten und einer gerichteten Beziehung zwischen den Knoten. Exemplarisch seien dies für die Knoten „Filme“ und „Schauspieler und für die Beziehung „spielt\_in“. Entsprechend beschreibt der exemplarische Graph, welcher Schauspieler in welchem Film spielt. Die Beschreibung des Musters des dargelegten Graphen mittels Cypher stellt sich wie folgt dar.

Match (s:Schauspieler) – [spielt\_in] -> (f:Film) Return s, f

Die entsprechende Rückgabe sind alle Knoten und deren Verbindungen untereinander, wie in Abbildung 6 dargestellt. Filme, zu denen keine Schauspieler in der Datenbank hinterlegt sind, werden nicht ausgegeben, da sie dem abgefragten Muster nicht entsprechen.

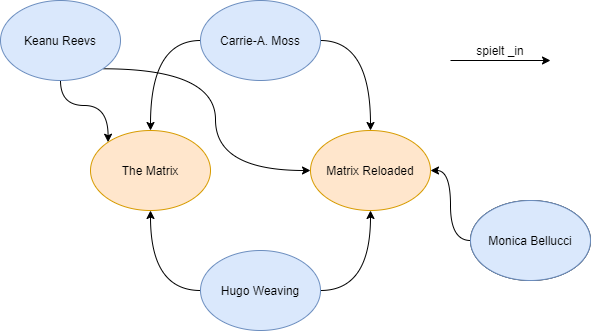


Abbildung 6: Ergebniss Relationenabfrage

Analog zu SQL bietet Cypher die Option, anhand von Attributen der Knoten und Kanten mittels einer Where-Bedingung das Muster zu Filtern. Folgend sei dies mittels Cypher dargestellt.

Match (s:Schauspieler) – [spielt\_in] -> (f:Film)

Where (s.name = „Carrie-A. Moss)

OR (s.name = „Monica Bellucci“)

Return s, f

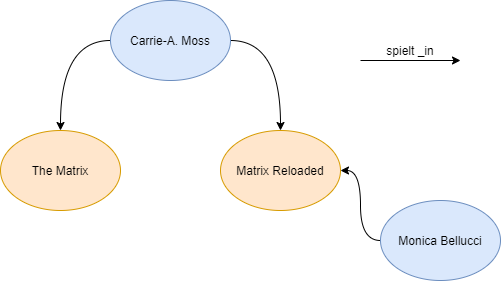


Abbildung 7: Ergebniss Relationenabfrage mit Einschränkungen auf Attributwerte

## Node.js

Node.js ist eine plattformübergreifende Laufzeitumgebung basierend auf JavaScript. Verwaltet wird das Open-Source-Projekt NodeJS durch die OpenJS Foundation mit Unterstützung der Linux Foundation. Die ursprüngliche Hauptaufgabe von NodeJS besteht in der Ausführung von JavaScript-Code außerhalb eines Browsers. Es wird somit möglich serverseitig dynamische Webseiteninhalte zu erstellen, bevor die Seite an den Benutzer übertragen wird. Entsprechend wird deutlich, dass NodeJS primär für die Entwicklung hoch performanter Webserver verwendet wird. Nativ wird von NodeJS nur JavaScript unterstützt. Es gibt jedoch Compiler, die es ermöglichen andere Sprachen, wie zum Beispiel CoffeeScript oder TypeScript in JavaScript zu übersetzen und somit für NodeJS nutzbar zu machen.

Auf der funktionalen Ebene ist NodeJS bezüglich der dynamischen Erstellung von Webseiteninhalten am ehesten mit PHP zu vergleichen, welches dies analog zu NodeJS ebenso unterstützt. Auf der technischen Ebene unterscheidet sich NodeJS jedoch deutlich von PHP. NodeJS führt Funktionen in der Regel parallel aus und verwendet sogenannte „Callbacks“ um die Fertigstellung oder das Scheitern einer Funktion zu signalisieren. PHP hingegen blockiert die meisten Funktionen bis zur Fertigstellung der vorherigen.

Neben der beschriebenen Hauptaufgabe wird NodeJS jedoch auch sukzessive als Werkzeug für diverse andere Aufgaben eingesetzt. In dieser Arbeit wurde NodeJS beispielsweise indirekt als Bestandteil der Entwicklungs- und Deploymentumgebung verwendet. Dies wird ermöglicht, durch den Node Package Manager (npm), welcher ein integrierter Bestandteil von NodeJS ist. Der npm ist ein Paketmanager für die Laufzeitumgebung von NodeJS, was letztlich als freies Repository für Paketerweiterungen für NodeJS interpretiert werden kann. Am Beispiel dieser Arbeit wurde eine Reihe von Paketerweiterungen wie beispielsweise „webpack“ oder „babel“ für die Entwicklung und das Deployment eingesetzt. Diese Erweiterungen reichen von Compileren bis hin zu Webservern für die lokale Entwicklung. Die im Rahmen dieser Arbeit eingesetzten Paketerweiterungen sind in Kapitel 4.2 im Detail aufgelistet.

## Webserver

Ein Webserver ist schlicht eine Software, welche serverseitig installiert wird. Die Hauptfunktion eines Webservers ist nach (Killelea, 2002) das Speichern, Verarbeiten und Übermitteln von Webseiten an Clients. Er operiert somit im sogenannten „Client-Server Modell“, welches in Abbildung 8 näher veranschaulicht ist. Ein Webserver kann im Allgemeinen eine oder mehrere Webseiten enthalten. Bei den bereitgestellten Seiten handelt es sich am häufigsten um HTML-Dokumente, die neben dem Textinhalt auch Bilder, Stylesheets und Skripte enthalten können.

Für Clientanfragen und Serverantworten ist das Hypertext Transfer Protocol (HTTP) von zentraler Bedeutung. Der Client stellt eine HTTP-Anfrage mit einem Uniform Resource Locator (URL), durch welchen die angeforderte Ressource serverseitig identifiziert werden kann. Anschließend antwortet der Webserver nach der Anfragenverarbeitung dem Client mit einer entsprechenden Nachricht. Diese enthält wiederum Informationen zum Status der Verarbeitung (erfolgreich oder gescheitert) und gegebenenfalls den angeforderten Inhalt beziehungsweise die angeforderte Ressource.

Während die Hauptfunktion darin besteht, Inhalte bereitzustellen, umfasst eine vollständige Implementierung von HTTP auch Möglichkeiten zum Empfangen von Inhalten. Diese Funktionalität wird zum Senden von Webformularen verwendet bis einschließlich des Hochladens von Dateien.

Viele generische Webserver unterstützen wie schon in Kapitel 2.4 aufgegriffen das serverseitige Skripting mit beispielsweise PHP (Hypertext Preprocessor) oder anderen Skriptsprachen. Dies bedeutet letztlich, dass das Verhalten des Webservers mittels separater Skripte erweitert beziehungsweise definiert werden kann, während die tatsächliche Serversoftware komplett unberührt bleibt. Die dynamische Generierung von Webseiteninhalten wird überwiegend zum Abrufen oder Ändern von Informationen aus Datenbanken verwendet.

Die aktuell am weitesten verbreiteten Webserver nach (w3techs.com, 2020) sind der Apache HTTP Server und der Nginx Webserver.

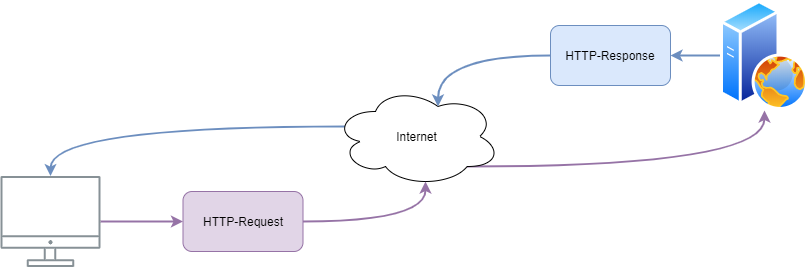


Abbildung 8: Client-Server Modell

## Webtechnologien zur Visualisierung

Der Sinn einer Visualisierung ist, dass diese auch gesehen wird. Der schnellste und einfachste Weg eine Visualisierung für mehrere Menschen zugänglich zu machen ist über eine Webseite. Ein Webbrowser hat laut (Murray, 2017) den Vorteil, dass er nicht an ein Betriebssystem wie beispielsweise Windows, Mac OS, Linux, Android oder iOS gebunden ist, sondern für die Anwendung nur einen Internetzugang und einen Browser benötigt. Ein weiterer Vorteil ist, dass diese Variante kaum beziehungsweise keine Lizenz- und Schulungskosten mit sich bringt.

Für die Visualisierung von Graphen auf Webseiten gibt es diverse Bibliotheken. Unter anderem sei an dieser Stelle Data-Driven Documents (D3), sigmajs und Cytoscape genannt. D3 ist jedoch die am weitesten verbreitete der genannten Bibliotheken und auch Bestandteil der Aufgabenstellung dieser Arbeit. Aufgrund dieses Sachverhalts wird folgend nur D3 näher betrachtet.

D3 bietet neben der Möglichkeit Graphen zu visualisieren noch zahlreiche weitere Anwendungsfälle. Alle gängigen Diagramme werden von D3 bereitgestellt, um Daten zu visualisieren. Die Bibliothek setzt hierbei auf standardisierte und bewährte Technologien wie Hypertext Markup Language (HTML), Scalable Vector Graphics (SVG) und Cascading Style Sheets (CSS). Der Einsatz von Webstandards bietet dem User, nach eigener Aussage, die Möglichkeit mit D3 zu entwickeln, ohne ein proprietäres Framework nutzen zu müssen.

Das zentrale Prinzip von D3 ist das Binden von Daten an Objekte des Document Object Model (DOM). Diese wiederum können selektiert, manipuliert und transformiert werden. Der Fokus der Bibliothek liegt auf Flexibilität, Performance und Wiederverwendbarkeit von Code. Sie eignet sich zudem sehr gut für große Datenmengen, was bezüglich der Thematik dieser Arbeit nicht gänzlich vernachlässigt werden darf. Für einen weiterführenden, tieferen Einblick in D3 sei auf die Online Dokumentation dieser verwiesen.

# Konzeption

Dieses Kapitel befasst sich mit den theoretischen Überlegungen, welche in dieser Arbeit erarbeitet wurden. Konzepte bezüglich der Infrastruktur, Datenmodell, Entwicklungsumgebung und Visualisierung werden thematisiert. Es bildet somit die theoretische Basis, auf welcher der praktische Teil dieser Arbeit aufsetzt. Dieser wird in Kapitel 4 näher betrachtet.

## Infrastrukturaufbau

Die folgende Grafik zeigt den Aufbau der Infrastruktur für diese Arbeit. Als Server kann hier beispielsweise ein Windows Server verwendet werden. Innerhalb dieses Servers befinden sich die Datenbank und der Webserver. Als Datenbank wurde hier Neo4j angedacht und als Webserver beispielsweise der Apache HTTP-Server. Der Webserver beinhaltet den entsprechenden Code mit den dazugehörigen Skripten.

Um auf diesen Inhalt zuzugreifen, muss zunächst der User an seinem Computer einen Browser öffnen. Dies kann wie hier abgebildet der Firefox sein. Anschließend wird über das Internet bzw. ein Intranet auf die Webseite zugegriffen.

Die einzelnen Komponenten sind in dieser Grafik nur beispielhaft aufgeführt. Diese können durch vergleichbare Komponenten ersetzt werden. Hier ein kleiner Auszug, welche gängigen Komponenten dies sein könnten:

* Server: Windows, Linux, Unix
* Datenbank: Neo4j, MySQL, PostgreSQL, CouchDB
* Webserver: Apache HTTP-Server, Tomcat, Nginx
* Internet / Intranet
* Browser: Firefox, Google Chrome, Internet Explorer, Apple Safari

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung 9: Infrastrukturkonzept

Das Internet kann durch ein Intranet ersetzt werden, wenn es sich beispielsweise um eine geschlossene Anwendung handelt. Das Intranet ist ein vom Internet unabhängiges Rechennetz, welches nicht öffentlich zugänglich ist.

## Graphenmodell

Ein Graphenmodell ist bei der Erstellung einer Datenbank sehr wichtig, da dies die einzelnen Verbindungen und Abhängigkeiten abbildet. Die folgende Grafik zeigt ein Beispiel für ein Graphenmodell einer abstrahierten IT-Landschaft. Das Graphenmodell wurde so erweitert, dass es möglichst realistisch ist und viele eventuelle Szenarien abdecken kann, welche die Entscheidungsfindung unterstützen.

Der Aufbau des Graphenmodells ist in drei Spalten gegliedert. Die linke Spalte, welche in der Farbe Blau dargestellt ist, zeigt die Unternehmensstruktur. Bei der mittleren Spalte handelt es sich um das Informationssystem. Dieses ist die Verbindung zwischen der Unternehmensstruktur und der in der Farbe Grün dargestellten rechten Spalte, dem technischen Aspekt. In manchen Fällen kommt es vor, dass der Name der Relation sich wiederholt. Dies ist der Fall, wenn Verbindungen dieselbe Art der Beziehung aufweisen. Beispiele hierfür können der Abbildung 10 entnommen werden. Beziehungen zwischen Elementen innerhalb der Kategorie der Unternehmensstruktur (blau) sowie der Kategorie des technischen Aspekts (grün) sind zulässig. Des Weiteren werden auch Beziehungen innerhalb einer Objektklasse (Knoten) erlaubt. Als Beispiel sei hierfür eine Beziehung zwischen zwei Informationssystemen an dieser Stelle genannt. Diese könnte beispielsweise durch eine implementierte Schnittstelle hervorgerufen werden. Weiterhin sei erwähnt, dass eine Objektklasse sowie eine Relation mit Attributen erweitert werden kann. Dies hängt in der Realität von der entsprechenden Datengrundlage ab.

**Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung**

Abbildung 10: Graphenmodell

Graphenmodelle sind sehr aussagekräftig. Durch das visuelle Verständnis fungiert das Modell als eine Art Kommunikationstool zwischen Entwickler und Endanwender. Wenn man ein konzipiertes Graphenmodell sieht, ist es zumeist sehr selbsterklärend und man kann die Zusammenhänge des Modells schnell erkennen, ohne thematisch tiefe Kenntnisse des Projektes zu haben.

Laut (Wirtschaftslexikon Gabler, 2018) beschreibt ein Graphenmodell die zu verarbeitenden Daten eines Anwendungsbereichs mittels Grafik. Zudem zeigt dieses die Beziehung, welche die Daten untereinander aufweisen.

Durch ein Graphenmodell können Redundanzen aufgedeckt werden. Ebenso wird dadurch ersichtlich, ob die Daten eindeutige Informationen aufzeigen. Unter Berücksichtigung dieser Punkte, können Rückschlüsse über die Datenqualität getroffen werden. Ein qualitativ hochwertiges Graphenmodell ist somit ein essenzieller Bestandteil eines Softwareprojekts, welches Datenbanken als Ressource einbindet. Aus diesem Grund ist es wichtig sich vor der Implementierung mit dem Graphenmodell auseinander zu setzen. Zusätzlich unterstützt der visuelle Entwurf der Datenstruktur, den Prozess der Implementierung.

## Visualisierungskonzept für Graphen

Neben den in Kapitel 2.2 genannten Best-Practice-Visualisierung ist nach (Hanschke, 2013) eine weitere der Graph. Ein Graph stellt die Beziehung zwischen mehreren Informationen visuell dar. Diese Verbindungslinien, auch Kanten genannt, können durch verschiedene Farben und Lienentypen beschaffen sein. Außerdem können diese auch mit einem Label versehen werden und somit weitere Informationen liefern. Die dargestellten Knoten zeigen Beziehungen und mögliche Abhängigkeiten untereinander auf, welche visuell schneller wahrgenommen werden als durch eine Liste, so (Hanschke, 2013) weiter.

Ein Graph ist im Gegensatz zu den in Kapitel 2.2 vorgestellten Visualisierungskonzepten deutlich agiler in der Darstellung. Er unterliegt keiner vorgegebenen Struktur wie es beispielsweise bei der Cluster-Grafik der Fall ist. Das einzige Strukturierende Element des Graphen sind die Beziehungen der Knoten. Es ist liegt somit eine datengetriebene Struktur vor, mehr jedoch nicht. Des Weiteren impliziert die Verwendung einer Graphdatenbank zur Datenverwaltung auch die Visualisierung mittels eines Graphen näher zu betrachten.

Die Darstellung von Graphen innerhalb einer Webanwendung erwies sich als sehr performant. Eine Darstellung des gesamten Graphen bei einer großen Datenmenge kann jedoch schnell unübersichtlich werden und ist somit nicht zwingend für eine Entscheidungsgrundlage geeignet. Es empfiehlt sich den Graphen in Subgraphen zu unterteilen. Die folgenden Mockups stellen mögliche Subgraphen dar, welche in dem Szenario der IT-Konsolidierungsprojekten einen Mehrwert liefern können. Wie sich dieser Mehrwert gestaltet ist folgend auf die Mockups näher beschrieben. Exemplarisch wurden zwei der Konzepte in dem praktischen Teil der Arbeit realisiert.

Ein Bild, das Vektorgrafiken enthält.

Automatisch generierte Beschreibung

Abbildung 11: Mockup Grad der Vernetzung

Das Mockup aus Abbildung 11 stellt den Grad der Vernetzung eines Knotens dar. Die Abbildung zeigt dies für vier unterschiedliche Knoten eines Typs. Der Grad der Vernetzung beschreibt, wie viele Beziehungen ein Knoten besitzt. In diesem Szenario werden alle Arten von Beziehungen berücksichtigt. Eine Filterung nach einem speziellen Typ von Beziehung wird somit nicht vorgenommen. Entsprechend kann dieses Kriterium herangezogen werden, um zu beurteilen, wie stark ein Bestandteil der IT-Landschaft in ein Unternehmen eingebunden ist.

Der orange farbige Knoten könnte beispielsweise ein Informationssystem darstellen, welches diverse Relationen zu Technologien, Organisationseinheiten und Fachprozessen besitzt. Möchte man nun das Informationssystem durch ein anderes ersetzen, kann diese Darstellung als Indikator für die Komplexität dieses Prozesses herangezogen werden. Sie gibt entsprechend direkt Aufschluss über den Aufwand bzw. die Machbarkeit ein Element der IT-Landschaft zu ersetzen. Ein Weiterer Vorteil dieser Grafik wird in Bezug auf die Analyse einer IT-Landschaft deutlich. Der Grad der Vernetzung gibt auch eine Tendenz für die Bedeutung eines Elements der IT-Landschaft in einem Unternehmen.

Ein Bild, das Vektorgrafiken enthält.

Automatisch generierte Beschreibung

Abbildung 12: Mockup Beziehung zu

Ein weiteres Beispiel, wie Graphen die Analyse einer IT-Landschaft unterstützen können, sei durch das Mockup in Abbildung 12 dargestellt. Hier wird im Gegensatz zum Grad der Vernetzung nur ein Typ der Beziehung eines Knoten visualisiert. Dies wird beispielsweise relevant, wenn nicht vorhandene Redundanzen oder spezielle Beziehungen aufgedeckt werden sollen. Ein passendes Beispiel hierfür sei ein Mitarbeiter mit diversen Verantwortlichkeiten für mehrere Informationssystem für den kein adäquater Ersatz im Unternehmen existiert. Dies würde zweifelsfrei zu Problemen führen, wenn dieser Mitarbeiter länger aus dem Unternehmen ausscheidet. Daher müssen seine Aufgaben auf andere Mitarbeiter umgelegt werden.

Ein weiterer Fall könnte das Aufdecken von Systemen sein, welche eine veraltete Technologie einsetzen. Diese sollten entsprechend zeitnah ersetzt werden. Gründe hierfür seien zum Beispiel die Sicherheit oder die Lauffähigkeit eines Systems. In diesem Szenario würden die weißen Knoten die Informationssysteme repräsentieren, welche diese Technologie verwenden. Durch diese Abfrageart wird schnell deutlich, welche Informationssysteme davon betroffen sind.

Ein Bild, das Vektorgrafiken enthält.

Automatisch generierte Beschreibung

Abbildung 13: Mockup Ampelsystem

Das Mockup in Abbildung 13 stellt ein Ampelsystem dar. Es dient der Gliederung in Klassen. Es könnte zum Beispiel für die Darstellung von Risikopotentialen zum Einsatz kommen. Ein passendes Beispiel sei durch das Risikopotential gegeben, welches durch das Supportende einer Technologie entsteht. Sollte eine Technologie keinen Support mehr erfahren, ist dies ein hohes Sicherheitsrisiko für das Unternehmen und die Technologie könnte entsprechend mit der Farbe Rot gekennzeichnet werden. Eine Technologie, welche ein Supportende innerhalb einer definierten Zeitspanne aufweist, würde folglich mit der Farbe Gelb gekennzeichnet werden. Technologien ohne ein bekanntes Risiko bezüglich eines Supportendes entsprechend mit der Farbe Grün.

Ein Bild, das Vektorgrafiken enthält.

Automatisch generierte Beschreibung

Abbildung 14: Mockup Nutzeranzahl

Das Mockup in der Abbildung 14 zeigt ein Visualisierungskonzept, um messbare Quantitäten zu verdeutlichen. Zum Beispiel könnte die Anzahl der Benutzer eines Informationssystems veranschaulicht werden. Je größer ein Knoten dargestellt wird, desto mehr Nutzer verwenden ein Informationssystem. Dies kann beispielsweise eine Tendenz über die verursachten Lizenzkosten eines Informationssystems geben oder Aufschluss über den potentiellen Aufwand dieses zu ersetzen.

Dieses Visualisierungskonzept und das des Ampelsystems sind sehr gut geeignet, um sie mit anderen Graphvisualisierungen zu kombinieren, da sie die Hauptinformation nicht durch den Graphen darstellen, sondern jeweils durch eine weitere Dimension wie Größe und Farbe von Objekten.

## Kombiniertes Visualisierungskonzept

Eine Realisierung eines Ampelsystems, welches aktuell in Unternehmen eingesetzt wird, sei durch den Tech Radar gegeben. Dieses ordnet die Knoten in einzelne definierte Phasen ein. Dies kann eine gute Entscheidungsgrundlage darstellen, wenn es beispielsweise darum geht, welche Systeme Risiken aufweisen oder welche Technologien für neue Projekte verwendet werden sollten. Unternehmen wie Zalando haben bereits eine eigene Visualisierung ihrer Daten anhand dieses Konzeptes im Web veröffentlicht. Abbildung 15 zeigt diese Umsetzung des Unternehmens Zalandeo. Dieser Tech Radar besitzt insgesamt 4 Quadranten, welche unterschiedlichen Themengebieten zugeordnet sind. So repräsentieren alle Objekte des oberen linken Quadranten Frameworkbestandteile, welche im Unternehmen Zalando zum Einsatz kommen. Eine Aufteilung des Tech Radars in mehr oder weniger Quadranten ist möglich. Es sollte jedoch die Lesbarkeit der Darstellung berücksichtigt werden.

Analog zu einem klassischen Ampelsystem wird beim Tech Radar die Zugehörigkeit zu einer bestimmten Klasse mittels verschiedener Farben realisiert. Folgend seien die Klassen aus Abbildung 15 näher beschrieben und mit Beispielen aus dem Tech Radar von Zalando verdeutlicht.

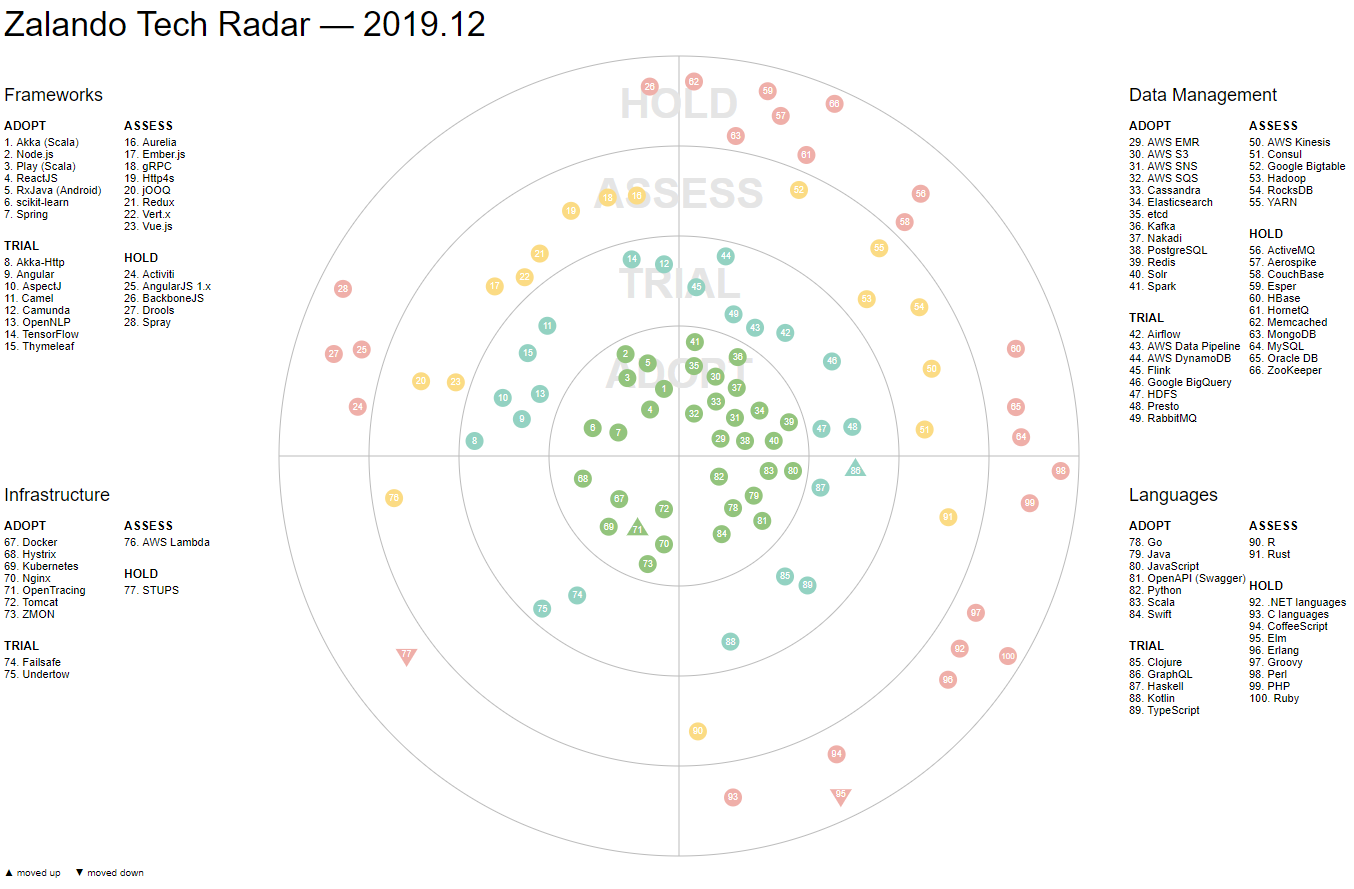


Abbildung 15: Technologie Radar

* **Adopt = Übernehmen**

In diesem Feld befinden sich Technologien, auf welche großes Vertrauen gesetzt wird, um langfristig mit diesen zu arbeiten. Ebenso sind diese weitestgehend risikofrei und sind recht verbreitet. Ein Auszug dieser am Beispiel von Zalando sind u.a.:

* Frameworks: Node.js, ReactJS, Spring
* Infrastructure: Docker, Nginx, Tomcat
* Data Management: Cassandra, Kafka, PostgreSQL, Spark
* Languages: Java, JavaScript, Python, Scala
* **Trial = Test**

Technologien werden in diesem Bereich getestet, u. a. in Projekten. Die Technologie hat sich teilweise bereits in der Problemlösung bewährt. Testtechnologien sind riskanter, da sie unvorhersehbare Auswirkungen haben können. Ein Auszug dieser am Beispiel von Zalando sind u.a.:

* Frameworks: Angular, Camunda, Thymeleaf
* Infrastructure: Failsafe, Underlow
* Data Management: Airflow, AWS DynamoDB, Flink, Google BigQuery, Presto
* Languages: GraphQL, Haskell, Kotlin, TypeScript
* **Assess = Bewertung**

Technologien aus diesem Bereich besitzen ein hohes Potential aber auch ein hohes Risiko. Die Technologien dieser Gruppe sollten näher untersucht werden. Mit einer solchen Technologie hat das Unternehmen beispielsweise nur wenige Erfahrungen gesammelt, welche jedoch als positiv und vielversprechend zu werten sind. Ein Auszug dieser am Beispiel von Zalando sind u.a.:

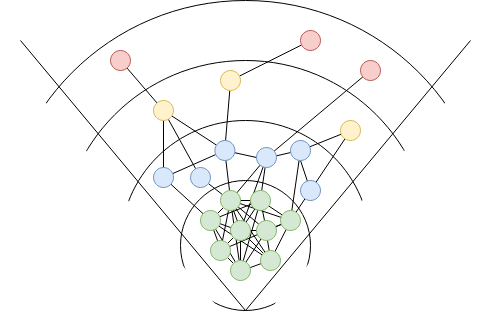
* Frameworks: Aurelia, Ember.js, Redux, Vert.x, Vue.js
* Infrastructure: AWS Lambda
* Data Management: Consul, Google Bigtable, Hadoop, RocksDB, YARN
* Languages: R, Rust
* **Hold = Halten**

In diesem Bereich befinden sich Technologien, welche nicht für neue Projekte präferiert werden. Diese sind lediglich für aktuell bestehende Projekte notwendig. Daher sind diese Technologien keine weiteren Investitionen mehr wert. Ein Auszug dieser am Beispiel von Zalando sind u.a.:

* Frameworks: Activiti, AngularJS 1.x, BackboneJS, Drools, Spray
* Infrastructure: STUPS
* Data Management: ActiveMQ, CouchBase, Esper, MongoDB, MySQL, Oracle DB
* Languages: .NET languages, Elm, Groovy, Perl, PHP, Ruby

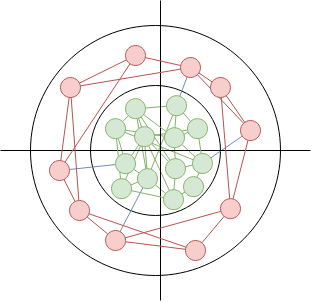
Eine naheliegende Erweiterung des zuvor beschriebenen Tech Radars sei durch das Ergänzen der Beziehungen unter den dargestellten Elementen gegeben. Das Sichtbarmachen der Beziehungen lässt in Bezug auf die eingesetzten Technologien diverse Rückschlüsse über die Gesamtsituation eines Unternehmens auf diesem Gebiet zu.

Unter der Annahme die Gesamtsituation des zugrundeliegenden Unternehmens ist positiv, so sollte der durch die Ergänzung der Beziehungen entstandene Graph, im Zentrum des Tech Radars sehr stark vernetzt sein. Entsprechend hierzu sollte die Vernetzungsdichte des Graphen nach außen hin deutlich abnehmen. Dieses Schema sei in Abbildung XX visualisiert.



Existieren zu dem beschrieben Schema Abweichungen, kann dies direkt mittels einer solchen Grafik deutlich gemacht werden. Eine Abweichung sei beispielsweise eine starke Vernetzung in einem Randbereich des Tech Radars. Eine Abweichung innerhalb eines Quadranten würde im Zuge dessen darauf hindeuten, dass in dem Themengebiet des entsprechenden Quadranten verstärkt Technologien zu ersetzen sind.

Ein weiteres denkbares Muster, welches durch ein Tech Radar als Graph vorkommen kann, ist dass die Dichte der Vernetzung innerhalb einer Klasse höher ist als zwischen unterschiedlichen Klassen. Ein einfaches Beispiel hierfür sei der Einsatz einer veralteten Technologie, welche wiederum den Einsatz weiterer veralteter Technologien voraussetzt. Beim Auftreten eines solchen Musters könnten Rückschlüsse über die Kompatibilität der Technologien zu neueren Technologien gezogen werden. Des Weiteren lassen sich hierdurch Technologien aufdecken, welche für das Auftreten eines solchen Musters verantwortlich sind. In Abbildung XX sei dieses beschriebene Muster schematisch visualisiert.



Die roten Knoten beischreiben hier die veralteten Technologien und die grünen Knoten die vom Unternehmen bevorzugten Technologien. Es wird ersichtlich, dass die veralteten Technologien überwiegend nur andere veraltete Technologien referenzieren. Entsprechend lässt sich hieraus ableiten, dass die meisten der veralteten Technologien obsolet werden, insofern ein Ersatz für diverse veraltete Technologien eingeführt wird. Weiterführend, wird jedoch auch deutlich, dass bei einer Ersetzung einer veralteten Technologie gegebenenfalls weitere veraltete Technologien aufgrund ihrer Abhängigkeiten ersetzt werden müssen.

## Konzeption der Entwicklungsumgebung

Bei der Erstellung des Konzeptes für die genutzte Entwicklungsumgebung, war neben den technischen Anforderungen dieser Arbeit auch die Nachhaltigkeit, die einfache Verwaltung wie auch die einfache Portierung des Systems auf andere Geräte von Bedeutung. In der Abbildung 16 sind die einzelnen Komponenten der Entwicklungsumgebung und deren Verknüpfungen untereinander visualisiert.

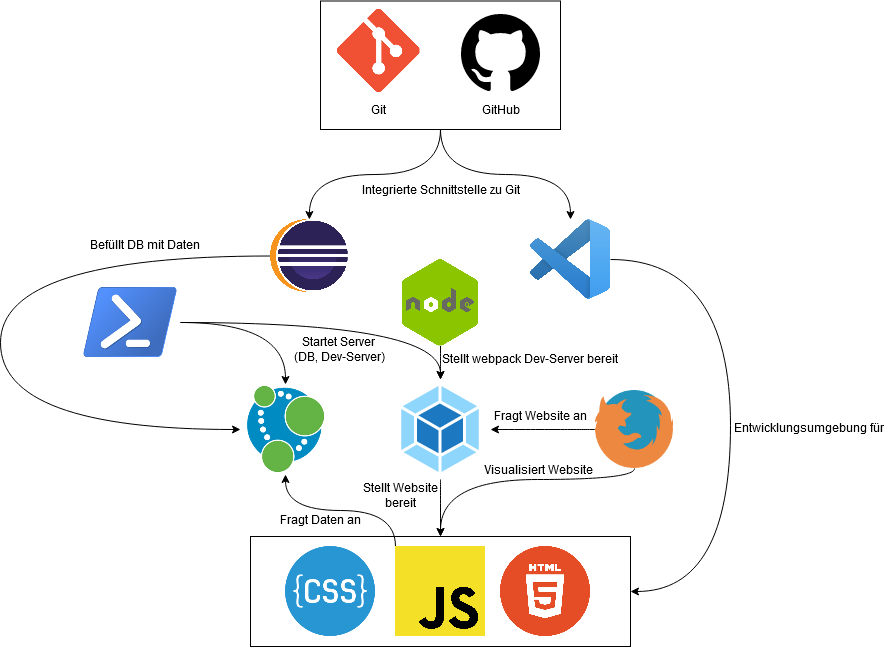


Abbildung 16: Konzept der Entwicklungsumgebung

Als zentraler Ablageort für den Code wurde GitHub verwendet. Hinter GitHub steckt das Versionsverwaltungssystem Git, welches sich auch in der Namensgebung von GitHub wiederfindet. Die Wahl fiel auf Git, da es diverse Schnittstellen zu populären integrierten Entwicklungsumgebungen (IDE) bietet und zudem den aktuellen Standard im Bereich der Versionsverwaltungssysteme definiert. Der Einsatz eines Repositories innerhalb eines Softwareprojekts bietet neben der Datensicherheit auch noch den Vorteil der einfachen Verteilung des Systems auf andere Geräte.

Als IDE wurde Visual Studio Code für die Entwicklung für CSS, JavaScript und HTML5 und für die Entwicklung für Java wurde Eclipse gewählt. Visual Studio Code zeichnet sich durch eine sehr schlanke IDE aus, welche modular durch zahlreiche Erweiterungen ergänzt werden kann. Erweiterungen sind für diverse Sprachen vorhanden. Eclipse ist eine sehr weitverbreitete IDE für Java und aufgrund ihrer langjährigen Historie sehr bewährt.

Der Code des Softwareprojekts wird mittels eines Webservers bereitgestellt und über einen Browser visualisiert. Der Webserver wird durch die Paketerweiterung webpack zur Verfügung gestellt, welche wiederum aus der Laufzeitumgebung NodeJS stammt. Die Installation von webpack erfolgt über den in NodeJS integrierten Node Package Manager. Dieser bietet auch die Option in einer zentralen Konfigurationsdatei (package.json) Abhängigkeiten des Softwareprojekts zu anderen Paketerweiterungen zu definieren. Diese können dann wiederum mittels des Node Package Managers und der Konfigurationsdatei auf einem anderen System automatisiert installiert werden. Entsprechend trägt der Node Package Manager erheblich zur einfachen Verteilung des Systems auf anderen Geräten bei.

Die eigentliche Hauptaufgabe von webpack liegt jedoch nicht darin einen Webserver für die Entwicklung bereitzustellen, sondern in dem Bereich des Deployments. Ist ein lauffähiger Zustand des Softwareprojekts erreicht, kann mittels webpack ein Build erzeugt werden. Hierbei extrahiert webpack den gesamten projektrelevanten Quellcode, Ressourcen und Stylesheets und führt dies in entsprechenden Dateien zusammen. Dies führt beispielsweise dazu, dass eine komplette Bibliotheksdatei von mehreren Megabyte in Abhängigkeit, des verwendeten Quellcodes auf mehrere Kilobyte reduziert werden kann. Der Build kann somit mit einer Art Standaloneimplementierung verglichen werden.

Als abschließende Komponente der Entwicklungsumgebung sei an dieser Stelle der Datenbankserver von Neo4J erwähnt. Dieser ist für die Datenspeicherung und Datenbereitstelleung verantwortlich. Er kommuniziert direkt mit der Website unter Zuhilfenahme spezieller Datenbanktreiber für JavaScript. Auf eine Middleware zwischen Datenbank und Website wurde bewusst verzichtet, da es das Gesamtsystem spürbar komplexer gemacht hätte, jedoch im aktuellen Zustand des Prototyps keinen erheblichen Mehrwert liefert. Das System kann jedoch ohne größeren Aufwand aufgrund des modularen Aufbaus zu einem späteren Zeitpunkt um eine Middleware nachgerüstet werden.

# Implementierung

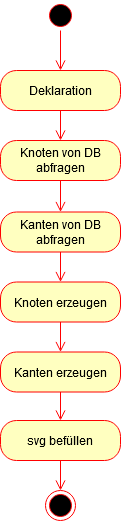
In diesem Kapitel wird gezeigt, was für das Vorgehen bei der Umsetzung der technischen Seite der Arbeit erforderlich ist. Es wird der grobe Ablauf des Programms aufgezeigt, auf spezielle Schwierigkeiten eingegangen, sowie die Inbetriebnahme des Ergebnisses des praktischen Teils erläutert. Abschließend erfolgt eine kurze Auflistung der in dieser Arbeit eingesetzten Ressourcen und Technologien.

## Beschreibung der implementierten Anwendung

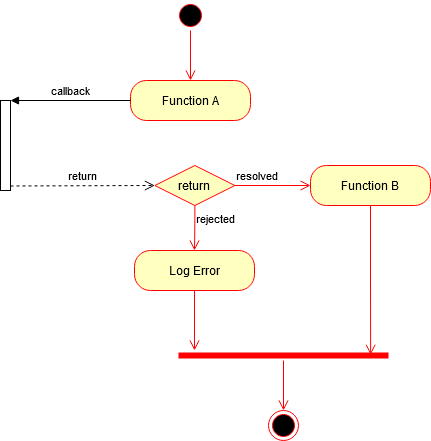
Der grobe Ablauf des Programms sei durch die Abbildung XX skizziert. Der Einstieg in die Anwendung erfolgt durch den Aufruf der implementierten Webseite. Hierdurch erfolgt der Start einer JavaSkript-Routine mit der eigentlichen Programmlogik. Der Start der Routine ist mit dem Lademechanismus der Website codeseitig verknüpft. Sie wird somit immer bei einem Aufruf oder Reload der Webseite ausgeführt. Die Routine lässt sich grob in sechs Abschnitte untergliedern, wie es auch in der Abbildung XX verdeutlicht wird. In dem Bereich der Deklaration werden Abhängigkeiten zu JavaScript-Bibliotheken oder zum verwendeten Stylesheet definiert, globale Programmvariablen angelegt, Verbindungsparameter der Datenbank deklariert und initialisiert, sowie diverse Statements in der Abfragesprache Cypher definiert.

Für die Initialisierung der Datenbankverbindung wird ein spezieller Treiber benötigt. Dieser wird durch die JavaScript-Bibliothek „neo4j-web“ zur Verfügung gestellt. In der Regel wird für jede Datenbank ein eigener spezieller Treiber benötigt. Eine Verwendung dieses Treibers mit einer MySQL Datenbank ist somit nicht möglich. Damit der Treiber eine Verbindung zu einer Datenbank aufbauen kann, benötigt er mindestens zwei Informationen. Zum einen wäre hier die URL, unter welcher die Datenbank zu erreichen ist und zum anderen die Authentifizierungsparameter für die Datenbank. Diese setzen sich zusammen aus dem Namen des Datenbanknutzers, mit welchem die Verbindung hergestellt werden soll, und dessen Passwort. Folgend sei die für die Initialisierung der Datenbankverbindung benötigte Codezeile dargestellt.

var driver = neo4j.driver("bolt://localhost", neo4j.auth.basic("neo4j", "myPWD"));

****

In den Abschnitten „Knoten von DB abfragen“ und „Kanten von DB abfragen“ werden mittels der Initalisierten Datnbankverbindung Cypher-Abfragen an die Datenbank übermittelt. Nach der serverseitigen Verarbeitung der Anfrage durch die Datenbank wird ein entsprechender Rückgabewert an das Programm übergeben. Die Abfrage von Daten einer Neo4J Datenbank mittels JavaScript ist eine sogenannte „non-blocking“ Operation. Dies bedeutet, dass die Operation ausgeführt wird, jedoch folgende Operation zulässt ohne einen Rückgabewert erhalten zu haben. Die Operation der Datenbankabfrage kann somit als asynchron verstanden werden. In dem Fall dieser Arbeit, würde das nun bedeuten, dass die Grafik zusammengebaut wird, ohne jedoch die Daten zu besitzen, da diese schlicht noch nicht von der Datenbank übermittelt wurden. Dies führt zu Fehlern während der Laufzeit. Um diesen Mechanismus zu unterbidnen musste in dieser Arbeit mit sogenannten „Promisses“ und mit „chainnign“ gearbeitet werden, um die asynchronen Prozesse in einen prozeduralen Ablauf zu bringen. Die Promisses sind sogenannte Callback-Methoden. Nach der Übergabe des Rückgabewertes wird eine folgende, definierte Methode ausgeführt. In Abbildung XX sei dieser Sachverhalt visualisiert.



Das channing wird durch den Sachverhalt relevant, dass in der implementierten Anwendung in gewissen Szenarien mehrere Abfragen in Folge an die Datenbank übermittelt werden müssen. In diesen Szenarien setzt sich das zu visualisierende Ergebnis aus mehreren Abfrageergebnissen zusammen. Die technische Umsetzung gestaltet sich wie folgt. Es sei ein Array mit einer definierten Anzahl an Cypher-Abfragen gegeben. Dieses wird an eine Funktion übergeben, welche alle Abfragen abarbeitet und nach erhalt aller Rückgabewerte diese an die Aufrufende Methode übergibt. Das channig erflogt beim Übermitteln der einzelnen Abfragen an die Datenbank. Dies bedeutet letztlich, dass der Prozess aus Abbildung XX1 in Abhängigkeit der Anzahl der Abfragen n-mal prozedural ausgeführt wird. Das channig verhindert damit, dass parallele Abfragen an die Datenbank übermittelt werden und resultierend daraus den Fehlerfall einer Prozessierung der Anwendung ohne den Erhalt aller angefragten Informationen verhindert. Folgend sei dieser Sachverhalt als Codebeispiel dargestellt und in der Abbildung XX2 visualisiert.

function parseStatementArray(stm) {

return new Promise(resolve => {

var test = [];

stm.reduce((chain, currentStatement) => {

var t = chain.then(() => parseCypherToDB(currentStatement));

test.push(t);

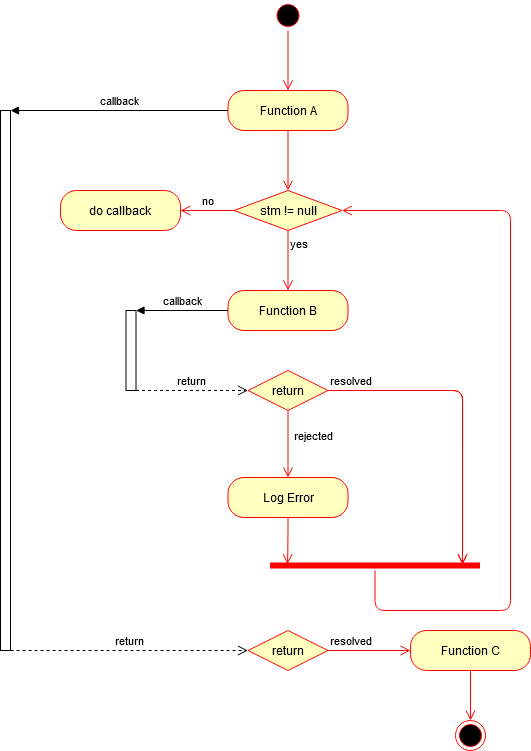
return t;

},Promise.resolve())

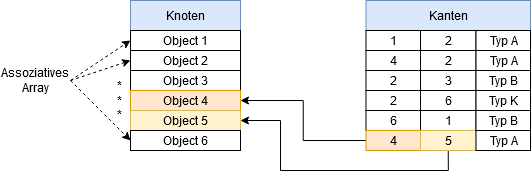
resolve(Promise.all(test))

})

}



In den Abschnitten „Knoten erstellen“ und „Kanten erstellen“ werden die Ergebnisse der Datenbankabfrage entsprechend für die Visualisierung aufbereitet. Alle Knoten und alle Kanten werden in separierten Arrays verwaltet. Der Inhalt des entsprechenden Abfrageergebnisses für einen Knoten wird in ein sogenanntes assoziatives Array konvertiert und die Ergebnisse einer Kante in ein Array, bestehend aus Quelle, Ziel und Typ der Beziehung. Quelle und Ziel bestehen hierbei aus Integerwerten, welche jeweils den Index des entsprechenden Knoten im Array der Knoten repräsentiert. Die Beziehungen zwischen den Arrays der Knoten und Kanten sei in Abbildung XX nochmals verdeutlicht.



Ein assoziatives Array unterscheidet sich zu einem normalen Array in dem Punkt, dass es Key-Value Paare speichert. Der Key übernimmt somit die gleiche Funktion wie der Index. Es muss jedoch gewährleistet sein, dass der Key innerhalb des Arrays eindeutig ist. Durch folgendes Codebeispiel sei ein solches assoziatives Array exemplarisch dargestellt.

i = {

Code: record.properties.Code,

Name: record.properties.Name,

Art: record.properties.Art,

type: "Standard"

};

Der letzte Abschnitt der Anwendung, „svg befüllen“, befasst sich mit dem Styling der darzustellenden Objekte, den Eigenschaften der Objekte und dem hinzufügen der Objekte zu der auf der Webseite visualisierten svg-Grafik. Das Styling wurde aus JavaScript ausgegliedert und erfolgt über ein Stylesheet im Format CSS. Mit den Eigenschaften der Objekte sind unter anderem funktionale Eigenschaften wie ein Drag-Event oder ein Hover-Event gemeint. Beispielsweise ruft der Hover-Event ein Tooltip-Popup auf, welches die Informationen des Knotens anzeigt. Des Weiteren werden in diesem Abschnitt noch Eigenschaften des Forcelayouts von D3 definiert. Das Forcelayout wird auf den Graphen angewendet und beschreibt zum Beispiel welchen Abstand die Knoten zueinander besitzen sollen, oder ob diese sich überlagern dürfen.

## Verwendete Technologien

Um die in der Arbeit verwendeten Sotware- und Entwicklungkomponenten in ihrer Gänze darzustellen, seien diese nachfolgend aufgelistet und mit den entsprechend verwendeten Versionsnummern versehen:

Software:

* Betriebssystem: Windows 10 Enterprise
* Google Chrome Version 79.0.3945.130 (64-Bit)
* Notepad++ Version 6.5.4
* GitHub Desktop Version 2.2.4
* Microsoft Visual Studio Code (User) Version 1.41.1
* Eclipse Java 2019-09
* XAMPP Version 7.3.9-0
* Apache HTTP-Server 2.4.41
* Neo4j Server Version 3.5.9
* Node.js Version 10.16.3
* PowerShell

Bibliotheken:

* Java ™ SE Development Kit 13.0.1.0 (64-Bit)
* D3.js Version 5
* jquery 1.11.0
* neo4j-web 1.0.4
* lodash 4.15.0

NodeJS Erweiterungspakete:

* @babel/core 7.8.4
* babel-loader 8.0.6
* babel-preset-env 1.7.0
* babel-preset-stage-0 6.3.13
* css-loader 3.4.2
* eslint 6.8.0
* file-loader 5.0.2
* html-webpack-plugin 3.2.0
* rimraf 3.0.2
* style-loader 1.1.3
* webpack 4.41.5
* webpack-cli 3.3.10
* webpack-dev-server 3.10.3

Aufgrund des Umfangs und der Relevanz wird an dieser Stelle nicht näher auf die einzelnen Komponenten eingegangen. Es sei diesbezüglich auf die Onlinedokumentation der jeweiligen Produkte verwiesen.

## Inbetriebnahme der erstellten Anwendung

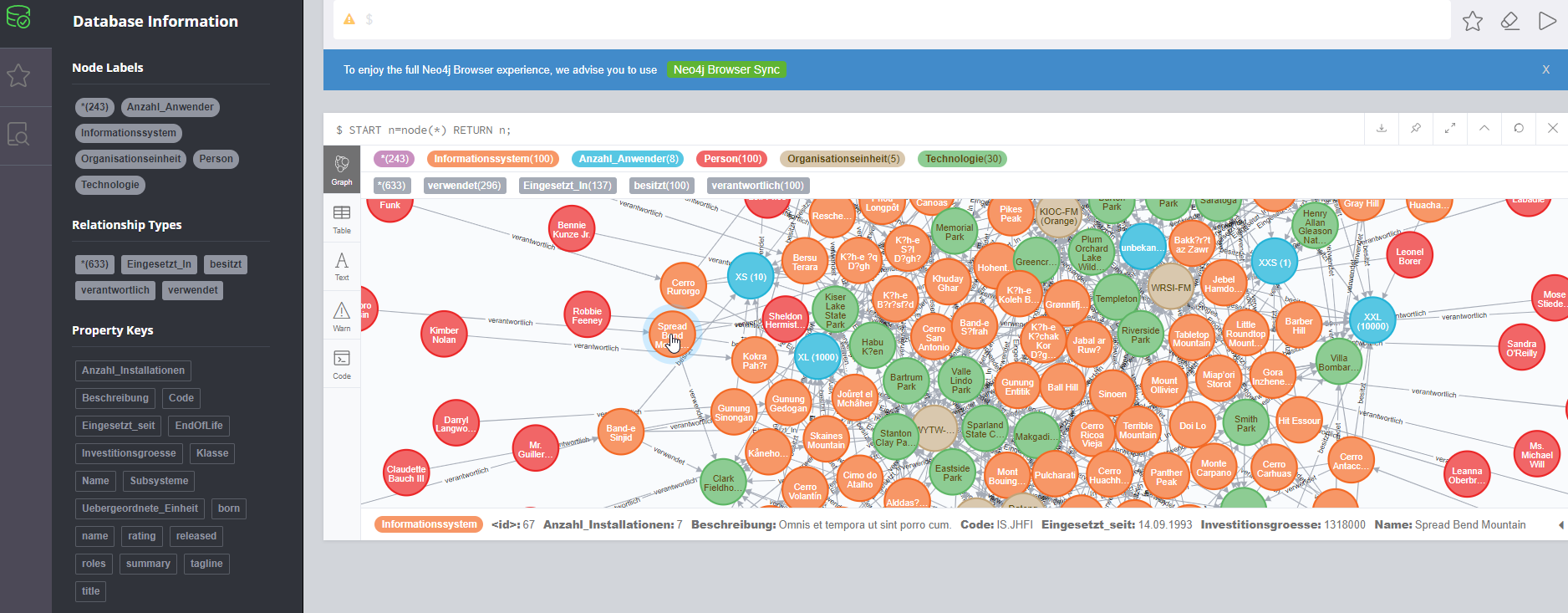
Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 17: Windows PowerShell

Anschließend kann im Webbrowser durch *Localhost:7474* die Webseite der Datenbank aufgerufen werden. An der folgenden Grafik ist deutlich zu erkennen, dass die Knoten die Objekte darstellen. Ebenso wird innerhalb der Datenbank der Inhalt dieser Knoten aufgezeigt. Die Verbindung zwischen den Objekten, also die Kanten, haben jeweils einem entsprechenden Namen der Relation, welcher direkt in der Datenbank und auf der Kante angezeigt wird.

Abbildung 18: Neo4j Datenbankauszug



Besonders von Vorteil ist bei Neo4j, dass innerhalb der Datenbank das Abfrageergebnis direkt visualisiert wird, wie auf der obigen Grafik zu erkennen ist. Somit ist direkt ersichtlich, ob die definierte Abfrage auch das gewünschte Ergebnis erzielt. Abgefragt wurde hier anhand des Befehls „*START n=node(\*) RETURN n;*“. Durch diese Abfrage gibt die Datenbank ihren gesamten Inhalt aus.

Für das Tool zum Programmieren wurde sich für Visual Studio Code entschieden. Zunächst standen verschiedene Programme zur Auswahl wie beispielsweise Notepad++ und Sublime Text 3. Jedoch hat sich schnell herausgestellt, dass diese nicht so komfortabel sind wie Visual Studio Code, daher wurde sich für dieses entschieden. Das bei Visual Studio Code die farbliche Zusammengehörigkeit des Codes übersichtlicher dargestellt ist als bei Sublime Text 3, war ein Grund für die Wahl für Visual Studio Code. Diese farbliche Zusammengehörigkeit ist auch als Syntax-Highlighting bekannt. Zudem hat dieser Editor einen zusätzlich integrierten Debugger und läuft mit verschiedenen Betriebssystemen wie z. B. Windows, Linux und Mac OS. [[1]](#footnote-1) Ein weiterer Grund für Visual Studio Code ist, dass die Hintergrundfarbe schwarz deutlich angenehmer beim Programmieren ist als die Hintergrundfarbe weiß wie bei Notepad++. Zudem befindet sich in Visual Studio Code rechts im Fenster eine optische Übersicht des Codes in Miniatur. Dadurch erhält man einen kleinen Überblick über die Struktur des Codes. In Abbildung 5 und Abbildung 6 sind diese optischen Unterschiede visuell dargestellt.

Abbildung 19: Visual Studio Code

Abbildung 20: Notepad++

Doch um eine Datenbank sinnvoll befüllen zu können, benötigt man selbstverständlich auch Daten. Die Testdaten wurden hier per Excel-Liste vom Betreuer der ISB AG zur Verfügung gestellt. Die Daten sind aus Datenschutzgründen hier fiktiv aufgeführt. Diese Liste umfasst sechs Arbeitsblätter mit mehreren Spalten und Zeilen Inhalt. Die Arbeitsblätter haben folgende Namen: Fachprozesse, Informationssysteme, Organisationseinheiten, Technologien, Standards und Standorte. Um die Daten jedoch in Datenbank zu bekommen, muss zunächst jedes Arbeitsblatt einzeln als eine csv-Datei exportiert werden.

Abbildung 21: Excel Daten

Anschließend wird diese Datei in das Programm Eclipse eingelesen. Innerhalb der Import Datei wurde der Code durch die Abfragesprache Cypher angesprochen. Cypher ist die Abfragesprache für die Graph-Datenbank Neo4j. Durch diese Übermittlung des Inhalts konnte eine Output Datei erzeugt werden, welche die Graph-Datenbank aufrufen und auslesen kann. Somit sind die Daten aus der Excel-Datei in die Graph-Datenbank Neo4j gelangt.

Ein Bild, das Screenshot enthält.

Automatisch generierte BeschreibungUm Einblick in die Daten zu bekommen, wurde ein Webserver aufgebaut, welcher eine Webansicht ermöglicht. Die folgende Grafik zeigt das hier verwendete Windows PowerShell. Nach dem Start des Programms, muss der korrekte Ordner ausgewählt werden und der Befehl *„npm run dev“* löst den Start aus.

Abbildung 22: Windows PowerShell

Wenn es gestartet ist, kann durch den Webseitenaufruf *Localhost:8080* auch schon die Webansicht erfolgen. Die folgende Grafik stellt diesen Webaufruf dar. Die Knoten spiegeln die Objekte wider und die Kanten die Relationen. Die Größe der Knoten ist unterschiedlich, da diese aussagt, wie viele Anwender dieses Produkt verwenden, also die Anzahl der Anwender. Je größer der Knoten, desto mehr Anwender. Je kleiner, desto weniger Anwender. Die Farbe der Knoten ist unterschiedlich, da jede Farbe einen anderen Typ abbildet. Die Typen sind entsprechend der Excel Arbeitsblätter abgeleitet. Durch ein Mouse Over Event können die Informationen zu den entsprechenden Knoten abgefragt werden. Ebenso können die Knoten durch einen Klick bewegt und anders positioniert werden.

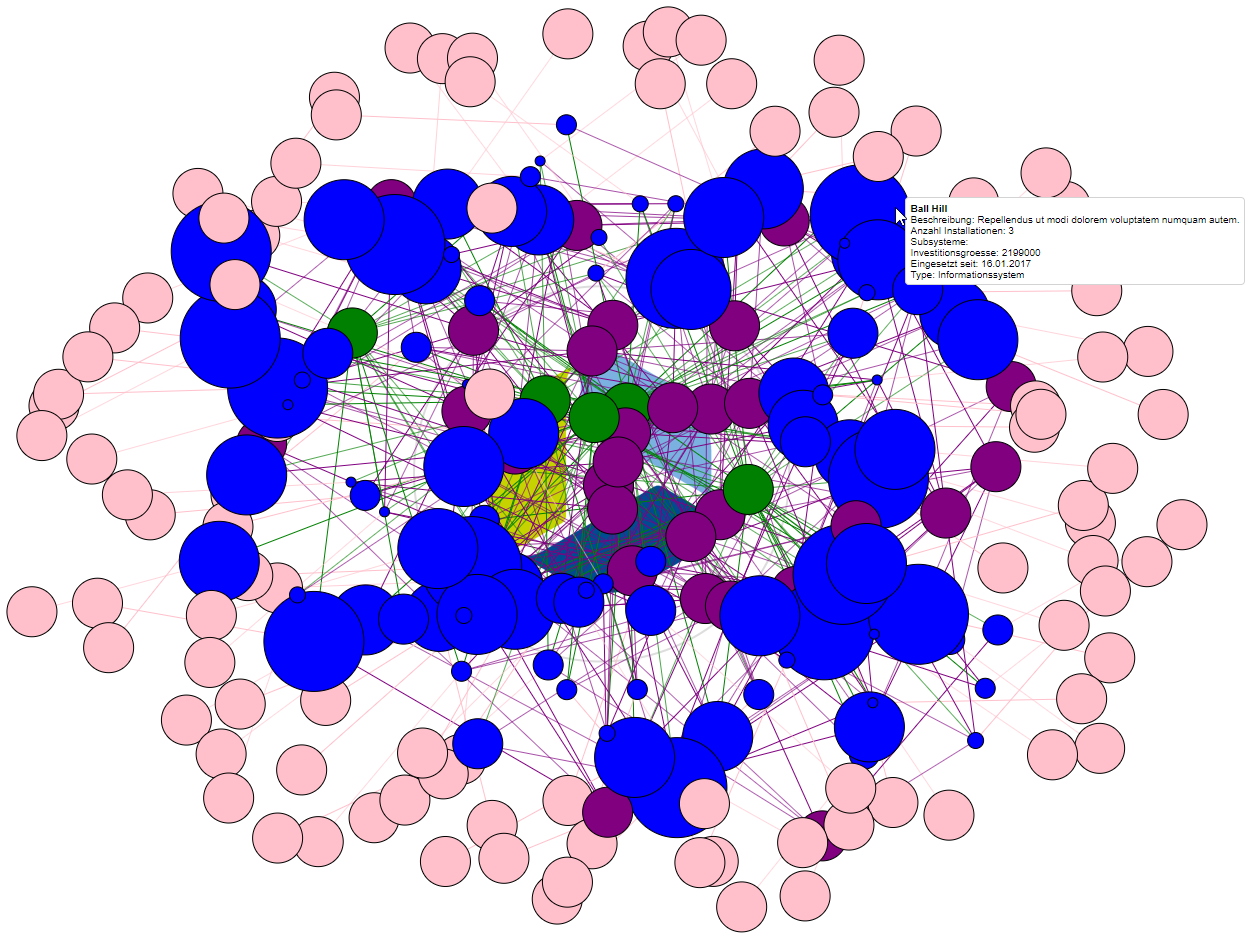


Abbildung 23: Daten Webansicht

Im Zuge des Infrastrukturaufbaus wurde ein „Build“ Ordner angelegt. WAS IST DAS, WAS MACHT DAS. Innerhalb dieses Ordners werden automatisiert alle Komponenten abgelegt, welche notwendig sind, um diesen Stand auf einem anderen Gerät zu erzeugen. Daraus folgt, dass keine essenziellen Bestandteile veraltet bzw. unvollständig sind.

# Evaluation (Fazit)

Bei der Evaluation wird die Arbeit in ihrer Gänze reflektiert. Dabei werden die wichtigsten Ergebnisse prägnant präsentiert und die gesamte Vorgehensweise wird konstruktiv hinterfragt und bewertet.

Der Beginn der Zusammenarbeit mit dem Unternehmen verlief reibungslos. Zunächst wurde diverse Literatur bereitgestellt, um einen Einblick in die Thematik zu erhalten. Bei den wöchentlichen Abstimmungen mit dem Betreuer des Unternehmens wurde auch sehr auf die theoretischen Zusammenhänge eingegangen und durch das Besprechen ein grobes Modell der Datengrundlage erstellt. Mitte Oktober wurden vorläufige Daten erzeugt, mit welchen das Einspielen von Daten in die Datenbank ermöglicht wurde. Die vorab programmierten Zusammenhänge konnten dadurch korrigiert und erweitert werden. Da die Thematik mit Graph Datenbanken kein Bestandteil des Studiums war, war hier sehr viel Einarbeitung und „learning by doing“ notwendig. Die Datenbasis wurde seitens des Betreuers regelmäßig geändert und erweitert, um möglichst praxisnahe Datenbestände zu generieren. Dies hatte zufolge, dass der Aufbau der Datenbank und des Imports, als auch die optische Darstellung abgeändert und teilweise auch neu strukturiert werden musste.

Im Laufe der praktischen Arbeit sind von Zeit zu Zeit einige Schwierigkeiten im Bereich des Programmierens aufgetreten, da teilweise das Know-How noch nicht ausreichend vorhanden war. Hier hieß es wieder tiefer in die Thematik einlesen, um eine Lösung für das vorliegende Problem zu finden. Zu jeder Zeit stand der Betreuer des Unternehmens mit Rat zur Seite, jedoch war der eigene Ehrgeiz da, es selbst lösen zu können.

Der Betreuer hat regelmäßig an Zeitplan und an die Risikoanalyse erinnert, um nicht zu sehr in Verzug zu geraten. Im Zuge dessen wurde ab Dezember auch mehr Energie in den schriftlichen Teil gelegt. Das sah so aus, dass zu Beginn viel praktisch gearbeitet wurde und an einem Wochentag geschrieben und dies nun umgekehrt wurde - sprich an einem Wochentag praktisch gearbeitet und den Rest der Woche an der Arbeit geschrieben.

Nachdem die aktuelle Version der Datenbestände eingespielt war, musste die optische Darstellung des Graphen angepasst werden. Durch die Anzahl der verschiedenen Typen war die Übersichtlichkeit nicht automatisch gegeben, was eine schlechte Entscheidungsgrundlage für den Kunden bedeuten würde. Daher mussten hier noch diverse Änderungen im Bereich der Visualisierung vorgenommen werden.

Durch das Endergebnis des Graphen ist es nun einem Berater möglich mittels Präsentation dem Kunden durch diese Darstellung einen visuellen Einblick in den Datenbestand zu ermöglichen. Anhand dieser Entscheidungsgrundlage wird es möglich sein eine Entscheidung zwecks IT-Konsolidierung zu treffen.

## Vorgehen bei der Visualisierung

Es war zu Beginn wichtig, die in die Datenbank eingespielten Daten, direkt zu Visualisieren. Dadurch konnten Unstimmigkeiten oder Verbesserungsmöglichkeit schneller festgestellt werden. Ebenso konnte dadurch abgeschätzt werden, ob diese Visualisierung eine Entscheidungsgrundlage bieten kann oder ob es zu unübersichtlich anhand der Datenmenge ist. Zum einen wurden die Daten innerhalb der Datenbank Neo4j visuell dargestellt und zum anderen auf der programmierten Webansicht im Browser.

Zudem wurde bei der Visualisierung der Daten festgestellt, dass einige Knoten sehr dominant wirken und die ganze Aufmerksamkeit auf sich ziehen, obwohl diese nicht relevanter sind als andere Knoten. Daher wurden die Farben der Knoten harmonisch angepasst, im Sinne von Farbintensität. Je heller und kontrastierender, desto stärker werden einzelne Farben hervorgehoben.

## Interpretation und Ergebnisse

Die vorliegende Datenmenge ist restlos in die Datenbank eingebunden. Anhand der Datenvisualisierung konnte die Darstellung weiter optimiert werden. Eine Erkenntnis war, dass bei einer großen Datenmenge, welche viele Abhängigkeiten und Beziehungen aufweist, eine Visualisierung in Form eines Graphen, eine Herausforderung war. Die Herausforderung war hierbei, dies gut sichtbar darzustellen, ohne die Datenmenge zu reduzieren. Ein Graph ist für eine umfangreiche Datenmenge konzipiert und bleibt zu jeder Zeit sehr performant.

Bei der Visualisierung der Daten hat sich gezeigt, dass man bei einer hohen Datenmenge schnell den Überblick verliert. Durch Veränderungen der Darstellung, wie beispielsweise wurde die Anzahl der Anwender nicht mehr als eigener Knoten dargestellt, sondern haben diese Informationen die Größe der Informationssystem-Knoten verändert. Somit wurden einige Knoten aus der Darstellung entfernt und die Größe der Knoten gibt Auskunft über die Anzahl der Anwender. Als Beispiel auf der Datenbasis bedeutet dies, wie viele Personen ein Informationssystem verwenden. Je größer der Knoten, desto mehr Anwender gibt es bei diesem Informationssystem. Auch die Farbgebung der Knoten hat sich erst bei der Visualisierung gezeigt, welche Farben dominanter wirken als anderen. Dementsprechend wurden hier auch Änderungen vorgenommen.

* Gehört zu Kapitel 4 Implementierung

Bei solch einem Projekt ist ein guter Infrastrukturaufbau essenziell. Aus diesem Grund wurde hier zu Beginn das Tool GitHub installiert, um dieses einerseits als Repository, also als Datenspeicherungstool, zu verwenden. Zum anderen war dieses Tool sehr gut, um von mehreren Geräten am selben Projekt arbeiten zu können. Das Besondere an GitHub ist auch, das im Falle eines erheblichen Fehlers auf eine vorherige Version zurückgegriffen werden kann, da es eine Versionsverwaltung bietet.

Als Datenbank wurde hier das Neo4j verwendet. Neo4j ist eine NoSQL Datenbank. Genauer gesagt eine Graph-Datenbank und unterscheidet sich zu einer Relationalen Datenbank insofern, dass diese die Möglichkeit bietet große Datenmengen, welche stark vernetzt sind, effizient zu analysieren. Dies könnten nach (Hunger, 2014) beispielsweise Filmempfehlungen oder Kurierrouten sein. Die Darstellung dieser Datenbank ist so, dass die Neo4j Datenbank aus einer Menge von Knoten besteht, welche die Objekte darstellen. Die Verbindungen zwischen den Knoten sind Kanten. Diese Kanten spiegeln die Beziehungen zwischen den Knoten wider.

Um die Datenbank im Web aufrufen zu können, musste zunächst ein das webpack gestartet werden. Hier wurde das Windows PowerShell verwendet. Nachdem das Programm geöffnet wurde, wählt man zunächst den entsprechenden Ordner aus. Die Steuerung ist so, dass man durch den Befehl „*cd..*“ einen Ordner zurück springt und durch „*cd Ordnername*“ nach vorne zum entsprechenden Ordner springt. Anschließend startet man den Vorgang durch den Befehl *„\bin\neo4j console“*. Dies wird durch die nachfolgende Grafik nochmal visuell veranschaulicht.

# Zusammenfassung und Ausblick

In diesem Kapitel wird die Abschlussarbeit kurz zusammengefasst und legt die wichtigsten Erkenntnisse der Arbeit dar. Der abschließende Ausblick zeigt auf, welches Potenzial noch in der Arbeit steckt und wie diese Arbeit in Zukunft noch weiter ausgebaut werden kann.

## Zusammenfassung

Diese Arbeit ist einerseits in einen konzeptionellen Teil, als auch in einen praktischen Teil untergliedert. Daraus resultiert ein prototypisches Werkzeug zur IT-Konsolidierung. Dieses stellt ein Planungswerkzeug für den Bereich der IT-Konsolidierung dar. Sein Zweck ist es, eine Entscheidungsgrundlage darzustellen. Diese basiert auf einer Datengrundlage, welche zum besseren Verständnis visuell dargestellt wird.

Ein Mensch kann bei einer Anzahl von mehreren 100 Systemen kaum einen Überblick behalten, welche Systeme noch aktuell sind oder welche irrelevant geworden sind. Durch eine visuelle Darstellung kann recht schnell eine Beziehung oder Abhängigkeit zwischen den Daten festgestellt werden und man erhält wieder einen Überblick. Doch von Zeit zu Zeit sollte ein Unternehmen eine IT-Konsolidierung durchführen, um sich von kritischen oder irrelevanten Technologien zu lösen. Durch eine Fusion können Irrelevanzen entstehen und durch veraltete Technologien können Sicherheitsrisiken entstehen.

Genutzt wird dieses Tool voraussichtlich von Beratern und Software Architekten, um den Kunden gewonnene Erkenntnisse visuell zu präsentieren. Anhand dieser Präsentation ist die IT-Konsolidierung für das Unternehmen verständlicher und eine Entscheidung anhand der Visualisierung kann getroffen werden.

Zu Beginn des technischen Teils wurde zunächst eine gut strukturierte Infrastruktur erstellt, welche das Arbeiten erheblich erleichtert hat. Hierbei war besonders hilfreich das verwendete Repository, welches neben einer Versionsverwaltung auch das Arbeiten an verschiedenen Geräten ermöglicht. Während der Einbindung der Daten in die Datenbank wurde das Datenmodell erstellt und nach Datenupdates aktualisiert und erweitert. Das Datenmodell war während des Programmierens hilfreich, um die Beziehungen und Anhängigkeiten der Daten schnell erkennen zu können.

Während der Implementierung war es notwendig die Daten, welche sich in einer Excel-Datei befinden zunächst mittels Importdatei in eine csv Datei umzuwandeln. Ohne diese Umwandlung ist es der Datenbank nicht möglich die Daten einzulesen.

Innerhalb der Graph Datenbank Neo4j konnte direkt visualisiert werden. Das hatte den Vorteil, dass direkt sichtbar wurde, ob die Datenbestände korrekt in ihrer Art und Menge eingespielt wurden. Die zusätzliche Visualisierung per Webseite hat durch die ansprechende Darstellung eventuelle Probleme oder Unstimmigkeiten in der Optik aufgezeigt. Durch diese beiden visuellen Kontrollen war es möglich gezielte Anpassungen vorzunehmen.

## Ausblick

Bezug auf künftige Visualisierung nehmen

**Literaturverzeichnis**

**Andreas Maier, Michael Kaufmann. 2016.** *NoSQL-Datenbanken.* s.l. : Springer Vieweg, 2016. 8. Auflage.

**BITKOM - Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V. 2011.** bitkom. *bitkom.org.* [Online] 2011. [Zitat vom: 11. 10 2019.] https://www.bitkom.org/sites/default/files/file/import/EAM-Enterprise-Architecture-Management-BITKOM-Leitfaden.pdf.

**2020.** Business-IT (Grafik). *business-it.link.* [Online] 2020. [Zitat vom: 07. 01 2020.] https://www.business-it.link/unternehmen/vision-leitbild-strategie.

**Hanschke, Inge. 2013.** *Strategisches Management der IT-Landschaft: Ein praktischer Leitfaden für das Enterprise Architecture Management.* München : Carl Hanser Verlag, 2013. 3. Auflage.

**Hunger, Michael. 2014.** *Neo4j 2.0: Eine Graphdatenbank für alle.* Paderborn : entwickler-press, 2014.

**Innern, Bundesministerim des.** BMI.Bund. [Online] [Zitat vom: 08. 10 2019.] https://www.bmi.bund.de/DE/themen/it-und-digitalpolitik/it-des-bundes/it-konsolidierung/it-konsolidierung-node.html.

**Klein, Manfred. 2017.** eGovernment Computing. *egovernment-computing.de.* [Online] 13. August 2017. [Zitat vom: 11. 10 2019.] https://www.egovernment-computing.de/was-ist-it-konsolidierung-in-der-oeffentlichen-hand-a-741563/.

**Michael Matzer, Big Data Insider (E-Book). 2019.** *Graph-Datenbanken.* Augsburg : Vogel IT-Medien, 2019.

**Murray, Scott. 2017.** *Interactive Data Vizualization for the Web: An introduction to designing with D3.* United States of Amerika : O´Reilly, 2017. 2. Auflage.

Onpulson Wirtschaftslexikon. *onpulson.de.* [Online] [Zitat vom: 13. 11 2019.] https://www.onpulson.de/lexikon/benutzerfreundlichkeit/.

**2019.** Opensource.Zalando. *opensource.zalando.com.* [Online] 12 2019. [Zitat vom: 05. 11 2019.] https://opensource.zalando.com/tech-radar/.

**Richard Lackes, Markus Sieper. 2018.** Wirtschaftslexikon Gabler. *wirtschaftslexikon.gabler.de.* [Online] 2018. [Zitat vom: 08. 11 2019.] https://wirtschaftslexikon.gabler.de/definition/datenmodell-28093/version-251730.

**Stefan Luber, Nico Litzel. 2017.** bigdata-insider. *bigdata-insider.de.* [Online] 12. 06 2017. [Zitat vom: 04. 11 2019.] https://www.bigdata-insider.de/was-ist-nosql-a-615718/.

**Trelle, Tobias. 2017.** Codecentric Blog: IT Expertenwissen von Entwicklern für Entwickler. *blog.codecentric.de.* [Online] 19. 06 2017. [Zitat vom: 16. 10 2019.] https://blog.codecentric.de/2017/06/graphen-visualisierung-mit-neo4j/.

Böhm, Jungkunz: *Grundkurs IT-Berufe*. Vieweg-Verlag, 2005

 Patrick, Killelea (2002). [*Web performance tuning*](https://www.worldcat.org/oclc/49502686) (2nd ed.). Beijing: O'Reilly. p. 264. [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [*059600172X*](https://en.wikipedia.org/wiki/Special:BookSources/059600172X). [*OCLC*](https://en.wikipedia.org/wiki/OCLC) [*49502686*](https://www.worldcat.org/oclc/49502686).

w3techs.com, 2020

**Quellen von Kapitel 3 noch einbauen (bisher Kap 1+2)**

Inge Hanschke S. 615.

Inge Hanschke S. 144 / 592.

<https://www.bitkom.org/sites/default/files/file/import/EAM-Enterprise-Architecture-Management-BITKOM-Leitfaden.pdf>

<https://www.business-it.link/unternehmen/vision-leitbild-strategie>

Inge Hanschke, S. 144f.

Inge Hanschke, S. 27.

Inge Hanschke S. 238ff.

Inge Hanschke S. 246ff.

BITKOM, Enterprise Architecture Management – neue Disziplin für die ganzheitliche Unternehmensentwicklung, 2011, S.22.

Inge Hanschke S. 241.

Inge Hanschke S. 253.

Neo4j 2.0: Eine Graphdatenbank für alle, entwickler-press, 2014, Paderborn.

<https://blog.codecentric.de/2017/06/graphen-visualisierung-mit-neo4j/>, Tobias Trelle, 2017

NoSQL-Datenbanken, Andreas Maier, Michael Kaufmann, 2016, Springer Vieweg, 8. Auflage, S. 237ff.

Ebook Big Data Insider, Graph-Datenbanken, Michael Matzer.

<https://www.bigdata-insider.de/was-ist-nosql-a-615718/>, Stefan Luber / Nico Litzel, 2017.

Ebook Big Data Insider, Graph-Datenbanken, Michael Matzer.

Ebook Big Data Insider, Graph-Datenbanken, Michael Matzer.

Interactive Data Visualization fort he Web S. 3.

<https://www.onpulson.de/lexikon/benutzerfreundlichkeit/> Wirtschaftslexikon.

<https://wirtschaftslexikon.gabler.de/definition/datenmodell-28093/version-251730> , 2018, Richard Lackes, Markus Sieper

Inge Hanschke, S. 239.

Inge Hanschke, S. 195.

<https://opensource.zalando.com/tech-radar/>

# Glossar (notwendig?)

**Begriff Definition / Erklärung**

Ampel

Bebauung

Best Practice

Enterprise Architecture (EA)

Enterprise Architecture Management (EAM)

Fachliches Domänenmodell

Funktionales Referenzmodell

Geschäftsarchitektur

Geschäftsprozess

Informationssystem

IT-Governance

IT-Konsolidierung

IT-Management

Organisationseinheit

Prozesslandkarte

Schnittstelle

Strategie

Unternehmensarchitektur

Zusätzlich: Deployment

Die Erklärungen sind dem Strategisches Management der IT-Landschaft: Ein praktischer Leitfaden für das Enterprise Architecture Management, als auch XX entnommen.

# Anhang

In der angehängten Daten-CD befinden sich folgende Punkte:

* schriftliche Arbeit
* kompletter Quellcode
* aktuelle Datenbestände

1. <https://www.heise.de/newsticker/meldung/Test-Microsoft-Visual-Studio-Code-ein-Editor-mit-integriertem-Debugger-4026553.html>, 2018 🡪 ggf. ersetzen [↑](#footnote-ref-1)