

# DIPLOMARBEIT

## Visualisierung der Ergebnisse des Stromnetzmodells

Ausgeführt im Schuljahr 2023/24 von:

Felix Schneider  
Clemens Schlipfinger

5AHIT-19  
5AHIT-17

Betreuer:

Ing. Jürgen Katzenschlager MSc, *Red*  
Ing. Jürgen Katzenschlager MSc

Krems, am 02. April 2024

### EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Krems, am 02. April 2024

Verfasser/innen:

---

Felix Schneider

---

Clemens Schlipfinger

# DIPLOMARBEIT

## DOKUMENTATION

Namen der Verfasser/innen	Clemens Schlipfinger Felix Schneider
Jahrgang / Klasse Schuljahr	5AHIT 2023/24
Thema der Diplomarbeit	Visualisierung der Ergebnisse des Stromnetzmodells
Kooperationspartner	Siemens AG Austria

*Nicht erkannt was das ist*

Aufgabenstellung	Siemens entwickelt ein neues Programmpaket zur Echtzeitberechnung von Stromnetzwerken. Für diese Berechnung ist die Qualität des Netzmodells von höchster Wichtigkeit, aber aufgrund seiner Größe sind Fehler unvermeidlich. Aktuell werden Fehler in unübersichtlichen Log Files gespeichert. Ein ausfallsicheres System mit strukturierter Visualisierung wird für einfache Auswertungen benötigt.
------------------	--

Realisierung	(Realisierung)
--------------	----------------

Ergebnisse	Das Projektziel ist die Entwicklung einer benutzerfreundlichen Webanwendung mit Filtermöglichkeiten, Graphen und Diagrammen. Dabei wird ein dezentrales Backend-System verwendet, welches reibungslos in die Siemens-Infrastruktur integriert werden kann und gleichzeitig höchste Stabilität gewährleistet. Dadurch wird es den Siemens-Ingenieur:innen erleichtert, Netzmodellfehler zu analysieren.
------------	--

*Würde Kefka er mögen*

Typische Grafik, Foto etc. (mit Erläuterung)	<p>content/img/Bild.png</p>
---	-----------------------------

Diese Grafik zeigt ...

Teilnahme an Wettbewerben, Auszeichnungen	Bosch Innovationspreis 2024
---	-----------------------------

Möglichkeiten der Einsichtnahme in die Arbeit	...
--	-----

Approbation (Datum / Unterschrift)	Prüfer/in	Abteilungsvorstand / Direktor/in
---------------------------------------	-----------	-------------------------------------

# DIPLOMA THESIS

## Documentation

Authors	Clemens Schlipfinger Felix Schneider
Form	5AHIT
Academic year	2023/24
Topic	Application for a company
Co-operation partners	Siemens AG Austria

Assignment of tasks	Siemens is developing a new programme package for the real-time calculation of power grids. The quality of the network model is of the utmost importance for this calculation, but due to its size, errors are unavoidable. Currently, errors are stored in confusing log files. A fail-safe system with structured visualisation is required for simple evaluations.
---------------------	---

Realization	(Realisierung)
-------------	----------------

Results	The aim of the project is to develop a user-friendly web application with filter options, graphs and diagrams. A decentralised backend system is used, which can be smoothly integrated into the Siemens infrastructure and at the same time guarantees maximum stability. This makes it easier for Siemens engineers to analyse network model errors.
---------	--

Illustrative graph, photo (incl. explanation)

content/img/Bild.png

This graphic shows ...

Participation in competitions  
Awards

Bosch Innovation Award 2024

Accessibility of diploma thesis

...

Approval  
(Date / Sign)

Examiner

Head of Department / /  
College

# Inhaltsverzeichnis

1. Präambel	9
1.1. Team . . . . .	9
1.2. Kurzfassung . . . . .	9
1.3. Danksagung . . . . .	9
1.4. Gendererklärung . . . . .	9
2. Einleitung	10
2.1. Ausgangssituation und Problemstellung . . . . .	10
2.2. Forschungsfragen . . . . .	10
2.2.1. Message Propagation . . . . .	10
2.2.2. Optimale Darstellungsmethoden . . . . .	10
2.3. Strukturierung der Arbeit . . . . .	11
3. Message Propagation	12
3.1. Einleitung . . . . .	12
3.1.1. Service-Oriented Architecture . . . . .	12
3.1.2. Schlüsselfaktoren für verteilte Systeme . . . . .	12
3.1.3. <del>Andere</del> Kommunikationstechniken in verteilte Systeme . . . . .	13
3.2. Enterprise Messaging Systems . . . . .	15
3.2.1. Eigenschaften . . . . .	15
3.2.2. Enterprise Service Bus . . . . .	16
3.2.3. <del>6</del> Grundlegende Konzepte . . . . .	16
3.2.4. Komponenten eines Messaging Systems . . . . .	17
3.2.5. Messaging Models . . . . .	18
3.2.6. Consumption Mode . . . . .	19
3.2.7. Quality-of-Service Garantien . . . . .	20
3.2.8. Protokoll . . . . .	20
3.2.9. Latenz und Durchsatz . . . . .	21
3.3. Vergleich von populären Enterprise Messaging Services . . . . .	22
3.3.1. Apache Kafka . . . . .	22
3.3.2. RabbitMQ . . . . .	24
4. Visualisierung	26
4.1. Benutzerfreundlichkeit und Performance . . . . .	26
4.1.1. Intuitivität . . . . .	26
4.1.2. Interaktivität . . . . .	27
4.1.3. Performance . . . . .	28
4.2. Arten von Datendarstellungen . . . . .	28
4.2.1. Zweidimensionale (2D) und Dreidimensionale (3D) Darstellungen . . . . .	28
4.2.2. Tabellen . . . . .	29
4.2.3. Diagramme . . . . .	32
4.2.4. Graphen . . . . .	33
4.3. Libraries zur visuellen Darstellung . . . . .	39
5. Architektur des Prototypen	40
5.1. Backend . . . . .	40
5.2. Frontend . . . . .	40

5/6/7 sollte wir noch besprechen → Struktur

6.	Aufsetzen eines Kafka-Systems	41
6.1.	Herausforderungen einer GraphQL API mit relationalen Datenbanken . . . . .	41
7.	Webapplikation	42
7.1.	Dashboard mit Kennzahlen des Netzmodells . . . . .	42
7.2.	Tabellen mit Filter- und Suchfunktionen . . . . .	42
7.3.	Graph des Stromnetzmodells . . . . .	42
7.3.1.	Einfache Integration von Cytoscape in Angular . . . . .	42
8.	Bewertung	45
I.	Literaturverzeichnis	46
II.	Abbildungsverzeichnis	50
III.	Tabellenverzeichnis	51
IV.	Quellcodeverzeichnis	52
V.	Abkürzungsverzeichnis	53
VI.	Übersetzungsverzeichnis	54
A.	Zusammenfassung und Ausblick	55
A.1.	Zusammenfassung . . . . .	55
A.2.	Ausblick . . . . .	55
B.	Anhang	56
B.1.	Kapitelverzeichnis . . . . .	56
B.2.	Besprechungsprotokolle . . . . .	57
B.2.1.	Begleitprotokoll 1 . . . . .	57
B.2.2.	Begleitprotokoll 2 . . . . .	60
B.2.3.	Begleitprotokoll 3 . . . . .	63
B.2.4.	Begleitprotokoll 4 . . . . .	66
B.2.5.	Begleitprotokoll 5 . . . . .	66
B.3.	Pflichtenheft . . . . .	67
B.4.	Arbeitspakte . . . . .	77
B.5.	Projekttagebücher . . . . .	78
B.5.1.	Projekttagebuch Clemens Schlipfinger . . . . .	78
B.5.2.	Projekttagebuch Felix Schneider . . . . .	79
B.6.	Datenträgerbeschreibung . . . . .	81

# 1. Präambel

## 1.1. Team

Das Projektteam besteht aus dem Projektleiter Felix Schneider und Kollegen Clemens Schlipfinger. Betreuer des Projekts ist MSc Jürgen Katzenschlager und Auftraggeber ist Siemens AG Austria.

*nichtige Schreibweise*

## 1.2. Kurzfassung

Siemens entwickelt ein neues Programmpaket zur Echtzeitberechnung von Stromnetzwerken. Für diese Berechnung ist die Qualität des Netzmodells von höchster Wichtigkeit, aber aufgrund seiner Größe sind Fehler unvermeidlich. Aktuell werden Fehler in unübersichtlichen Log Files gespeichert. Ein ausfallsicheres System mit strukturierter Visualisierung wird für einfache Auswertungen benötigt.

## 1.3. Danksagung

Wir möchten uns insbesondere bei unserem Betreuer MSc Jürgen Katzenschlager bedanken. Dieser hat uns während der gesamten Umsetzung unserer Arbeit mit seiner Expertise unterstützt und uns wertvolle Ratschläge gegeben.

Des Weiteren möchten wir uns besonders bei unseren Familien und unseren Freunden bedanken, welche uns in vielerlei Hinsicht bei unserem Projekt unterstützt haben. Wir fanden bei ihnen immer ein offenes Ohr, welches uns emotional unterstützte, indem sie für uns da waren und uns zuhörten, wenn wir uns über unsere Herausforderungen und Sorgen im Zusammenhang mit unserem Projekt unterhalten haben. Ihre Unterstützung hat geholfen, die emotionalen Belastungen, die mit dem Projekt einhergingen, besser zu bewältigen.

## 1.4. Gendererklärung

Zur besseren Lesbarkeit der Diplomarbeit wurde ausschließlich die männliche Form verwendet. Da Begriffe wie "Benutzerinnen und Benutzer" den Text unleserlich machen, wurde es schlicht auf "Benutzer" gekürzt, dies soll jedoch keine Geschlechterdiskriminierung zum Ausdruck bringen.

## 2. Einleitung

### 2.1. Ausgangssituation und Problemstellung

*3x  
des  
selbe*

Siemens entwickelt ein neues Programmpaket zur Echtzeitberechnung von Stromnetzwerken. Für diese Berechnung ist die Qualität des Netzmodells von höchster Wichtigkeit, aber aufgrund seiner Größe sind Fehler unvermeidlich. Aktuell werden Fehler in unübersichtlichen Log Files gespeichert. Ein ausfallsicheres System mit strukturierter Visualisierung wird für einfache Auswertungen benötigt.

Das Projektziel ist die Entwicklung einer benutzerfreundlichen Webanwendung mit Such- und Filtermöglichkeiten sowie Graphen. Dabei wird ein dezentrales Backend-System verwendet, welches reibungslos in die Siemens-Infrastruktur integriert werden kann und gleichzeitig höchste Stabilität gewährleistet. Dadurch wird es den Siemens-Ingenieuren erleichtert, Netzmodellfehler zu analysieren.

### 2.2. Forschungsfragen

#### 2.2.1. Message Propagation

Die Ausfallsicherheit moderner Systeme ist von höchster Wichtigkeit. Entkoppelte Systeme fördern die Last unter welcher Applikationen operieren können. Ein Service ist mit anderen niedrig gekoppelt, wenn wenig Abhängigkeiten gegeben sind. Die Messagepropagation unterstützt ein System dahingehend, die Kommunikation zwischen den einzelnen Softwarekomponenten in ein eigenes Service auszulagern. Welche Form der Messagepropagation am besten für verschiedene Anwendungszwecke geeignet ist, ist demnach eine Frage von hoher Bedeutung.

Demnach wird folgendes Gebiet erforscht:

*Vergleich der verschiedenen Formen der Messagepropagation in Enterprise Service Bus Technologien*

#### 2.2.2. Optimale Darstellungsmethoden

Soziale Netzwerke, biologische Systeme, Nahrungsketten, Dateisysteme und viele weitere Vorkommnisse stark vernetzter Daten verlangen eine effiziente Visualisierung dieser Datenflut. Die Darstellung dieser Informationen muss deswegen spezifisch an die Benutzerfreundlichkeit optimiert werden. Neben Diagrammen, Tabellen und Graphen müssen außerdem alle Elemente der Applikation benutzerfreundlich gestaltet sein, um die Benutzerzufriedenheit zu fördern.

Daraus ergibt sich folgende Forschungsfrage:

*Visualisierungsmethoden für stark vernetzte Daten*

*in fällen eines solchen  
nicht der Inhalt vorliegt als erwartet*

## 2.3. Strukturierung der Arbeit

Der Inhalt dieser Arbeit unterteilt sich in drei Hauptabschnitte. Jedes Kapitel ist eindeutig einem Abschnitt zuordenbar.

- Recherche
  - 3. Message Propagation
  - 4. Visualisierung
- Empirischer Teil
  -
- Bewertung
  -



Um mit der Materie vertraut zu werden, werden bestehende Forschungen der Literatur zusammengetragen und aufgearbeitet. Anschließend wird die Konzeptionierung des Prototypen niedergeschrieben, welcher zum Beweisen der These herangezogen wird. Schlussendlich wird ein Katalog zur Bestimmung der richtigen Form von Message Propagation und Vor- und Nachteile der verschiedenen Visualisierungsmethoden bei stark vernetzten Daten erstellt.

Mir fehlt ein Teil, was das Netzmodell ist  
 und was Sieben erreichen will.  
 sollte Teil des Kapitel 2 sein.

### 3. Message Propagation

#### 3.1. Einleitung

*Schalt  
da  
dron*

2. so  
heißt

Die Anforderungen an Softwaresysteme wachsen exponentiell. Heutzutage umfassen sie eine Vielzahl an Aspekten, wie eine 24/7 Verfügbarkeit, Echt-Zeit Reaktionszeiten oder die Fähigkeit Millionen Benutzer gleichzeitig zu bedienen. Um diesen Voraussetzungen gerecht zu werden steigt die Komplexität der Anwendungssysteme. Die einst monolithischen Systeme müssen in kleinere Teile aufgebrochen werden und transformieren sich in verteilte Softwaresysteme (*Service-Oriented Architecture*). Diese kleineren Teile oder Dienste (*Services*) müssen untereinander kommunizieren und sich koordinieren, jedoch stellt sich diese Koordination als äußerst herausfordernd dar. Traditionelle Kommunikationstechniken wie *Remote Procedure Call (RPC)* können diesen Kriterien nicht gerecht werden und Anwendungen, welche diese Techniken verwenden werden zu schwer wartbare fehleranfällige Konstrukte. *Enterprise Messaging Systems* bieten eine geeignete Lösung für die Kommunikation in verteilten Softwaresystemen. [1]

##### 3.1.1. Service-Oriented Architecture

In der *Service-Oriented Architecture (SOA)* werden Softwareressourcen als *Services* verpackt, die eigenständige Module darstellen. Diese bieten standardisierte Geschäftsfunktionalitäten und sind unabhängig vom Zustand oder Kontext anderer Dienste. Es werden eine Sammlung von *Services* erstellt, die miteinander kommunizieren können, indem sie Service-Schnittstellen verwenden, um Nachrichten von einem *Service* an einen anderen zu übermitteln oder eine Aktivität zwischen einem oder mehreren *Services* zu koordinieren. Auf diese Weise kann mit SOA ein sehr flexibles System geschaffen werden. [2]

##### 3.1.2. Schlüsselfaktoren für verteilte Systeme

*Wenig  
er/  
faktoren  
+ Hinführen*

Ein verteiltes System hat Faktoren, welche die Zuverlässigkeit, Effizienz und letztendlich die Qualität des Systems beeinflussen.

- **Kopplung:** In welchem Maße sind die Komponenten voneinander abhängig oder können unabhängig voneinander arbeiten? *würde diese Theorie nicht das Freig. fernhalten*
- **Latenz:** Wie schnell erfolgt die Kommunikation zwischen den Systemen?
- **Skalierbarkeit:** Wie skaliert das gesamte System, wenn mehr Systeme hinzugefügt werden und die Arbeitslast eine Zunahme erfordert?
- **Wartbarkeit:** Wie aufwendig ist die Wartung der gesamten Systems?
- **Zuverlässigkeit:** Inwieweit kann das System konsistent und vertrauenswürdig seine Funktionen erfüllen und neigt es dazu, Fehler oder Ausfälle zu haben?
- **Verfügbarkeit:** Wie leicht kann das System ausfallen? Wie robust ist das System gegenüber Ausfällen von Komponenten?
- **Erweiterbarkeit:** Wie einfach ist es, neue Geschäftsszenarien (*Business Cases*) umzusetzen oder neue Komponenten zu integrieren?

### 3.1.3. Andere Kommunikationstechniken in verteilte Systeme

~~Entweder Beobachten oder nicht sind alle Leser~~

Es gibt eine Vielzahl an Techniken für die Kommunikation in verteilten Systemen, jedoch haben viele klare Nachteile gegenüber der Verwendung von *Messaging Systemen*.

#### 3.1.3.1. Direkte Kommunikation

Die trivialste Lösung wählt eine Art von *Remote Procedure Calls* zwischen den einzelnen Teile des verteilten Systems, jedoch führt dies zu einer hohen Kopplung, da die einzelnen *Services* direkt miteinander kommunizieren. Wenn diese Systeme eine gewisse Größe erreichen, werden sie zu undurchsichtige Netze an Abhängigkeiten, dies für zu einer Aufwendigen Wartung, niedrigen Skalierbarkeit und erschwert jede Art von Erweiterung des Systems. Jedoch können die Anforderungen an die Echtzeitkommunikation erfüllt werden. [1, 3]

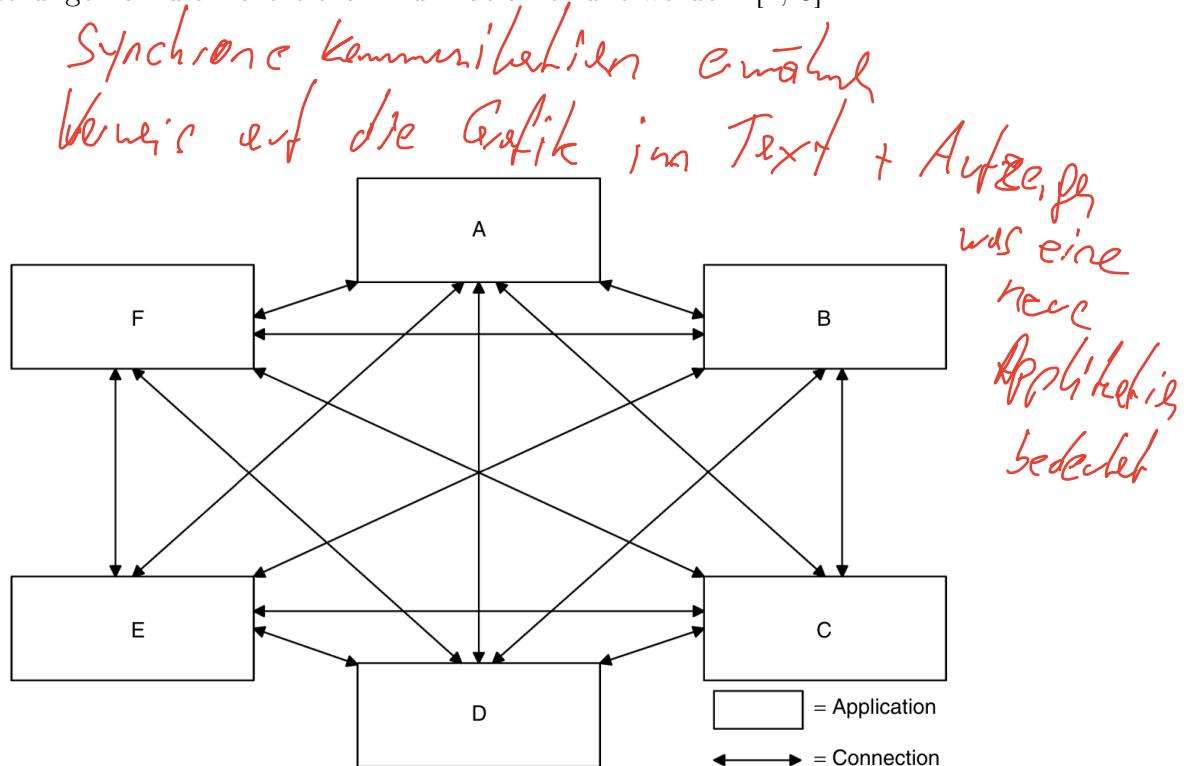


Abbildung 3.1.: Ein Beispiel für ein verteiltes System mit direkter Kommunikation  
[1]

*Plausibilis*

### 3.1.3.2. Verwendung einer Datenbank

Eine weitere Möglichkeit des Informationsaustauschs könnte die Kommunikation über eine gemeinsame Datenbank sein. In dieser Architektur sind alle *Services* auf das gleiche Datenbank-Schema abhängig, welches die Koppelung reduziert. Weitere *Services* können auch leicht integriert werden, da sie nur einen Datenbank-Schema entsprechend kommunizieren müssen. Wenn jedoch das Datenbank-Schema angepasst wird, müssen alle *Services* angepasst werden, dieser Prozess könnte mit großen Aufwand verbunden sein. Eine hohe Skalierbarkeit zu erreichen ist in den meisten Fällen sehr herausfordernd, insbesondere bei relationalen Datenbanken, da diese von Natur aus schwer zu skalieren sind.[1, 3]

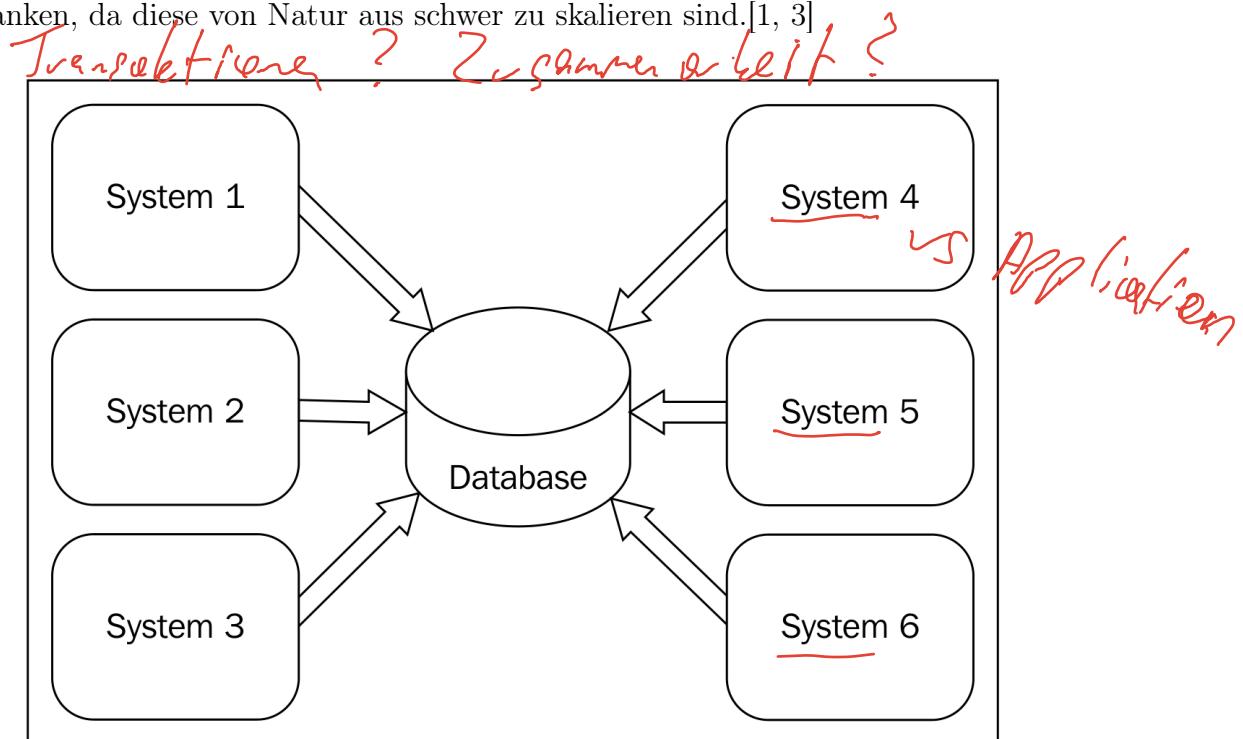


Abbildung 3.2.: System mit einer Datenbank als zentrale Kommunikationskomponente  
[3]

*wirksamere Option?*

### 3.2. Enterprise Messaging Systems

*Übertragung?*

Durch ein *Messaging System* sind Sender (*Publisher*) und Empfänger (*Consumer*) stark entkoppelt, da sie über einen gemeinsamen Mittelmann - dem *Messaging System* - kommunizieren. Die Sender müssen Nachrichten dem *Messaging System* übertragen und die Empfänger erhalten die Nachrichten vom *Messaging System*. In vielen Fällen können die Kommunikationspartner nicht von einander wissen, dies reduziert die Kopplung auf das Minimum. Das *Messaging System* kann Nachrichten validieren, speichern, transformieren und zu den geeigneten Empfängern weiterleiten (*Routen*), dadurch wird ein einheitlichen Weg um die Kommunikation zwischen zwei Systemen zu ermöglichen geboten. [3]

*Asynchron?*

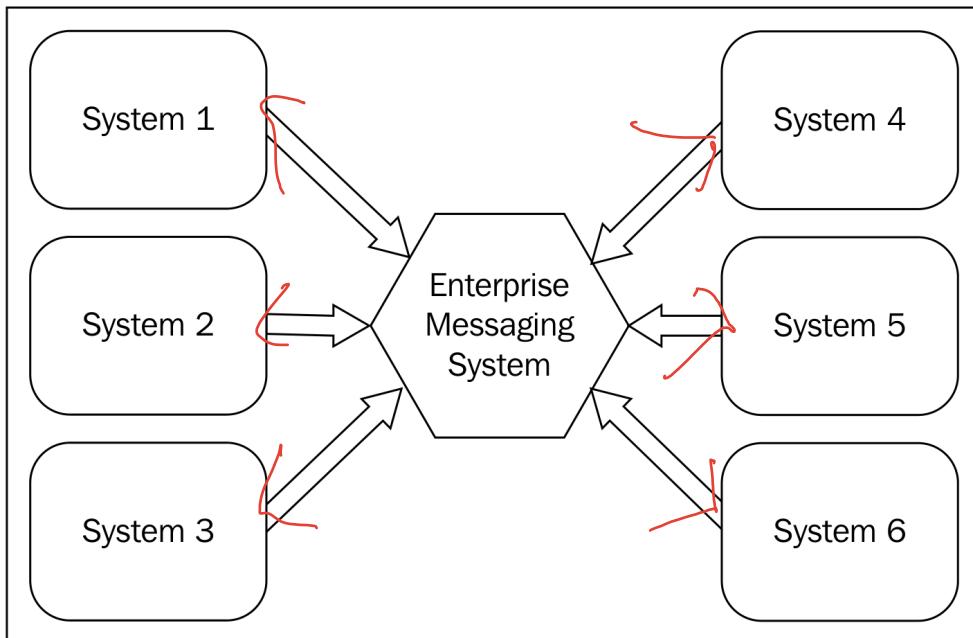


Abbildung 3.3.: Ein Beispiel für ein verteiltes System mit einem zentralen *Messaging System* [1]

#### 3.2.1. Eigenschaften

*vs Fallstudie?*

Die folgenden Eigenschaften können bei verteilten Systemen, welche *Messaging Systeme* als Grundlage der Kommunikation verwenden, beobachtet werden: *Quelle?*

- **hohe Skalierbarkeit** - *Messaging Systeme* können sich an große Auslastungen anpassen durch *Clustering*
- **niedrige Koppelung** - Da das *Messaging System* einen Mittelmann darstellt, sind die Kommunikationspartner entkoppelt.
- **leichte Erweiterbarkeit** - Es besteht die Möglichkeit, dem *Messaging System* ohne Beeinträchtigung der bestehenden Kommunikationskanäle ein weiteres System hinzuzufügen.
- **niedrige Latenz** - *Messaging Systeme* sind auf Echt-Zeit Datenverarbeitung ausgelegt und verwenden viele Techniken um den niedrigsten Overhead wie möglich zu erreichen.

- **gute Wartbarkeit** - Das *Messaging System* bildet den zentralen Punkt der Kommunikation, was zu einer erheblichen Vereinfachung der Übersicht und der Fehlersuche führt.

### 3.2.2. Enterprise Service Bus

*Überleitung + Woraus?*

Wenn in einem System auch noch verschiedene Protokolle wie REST, SOAP, JMX, AMQP oder RPC verwendet werden und diese Protokolle auch in das *Messaging System* integriert werden. Die Integration könnte durch verschiedene Adapter realisiert werden, welche die Protokolle auf andere konvertieren. Wenn ein *Message System* diese weiteren Funktionalitäten besitzt und einer seiner Hauptaufgaben die Schaffung einer Kompatibilitätsschicht wird, dann wird von einem *Enterprise Service Bus (ESB)* gesprochen. Dadurch kann der ESB die Integration von bestehenden Systemen oder Systemen die an ein bestimmtes Protokoll gebunden sind deutlich erleichtern. Allerdings erfordert es erheblichen Aufwand, dass der ESB stets allen Kommunikationsanforderungen (Protokolle usw.) gerecht wird. [3, 4]

*Eingehen auf Grafik*

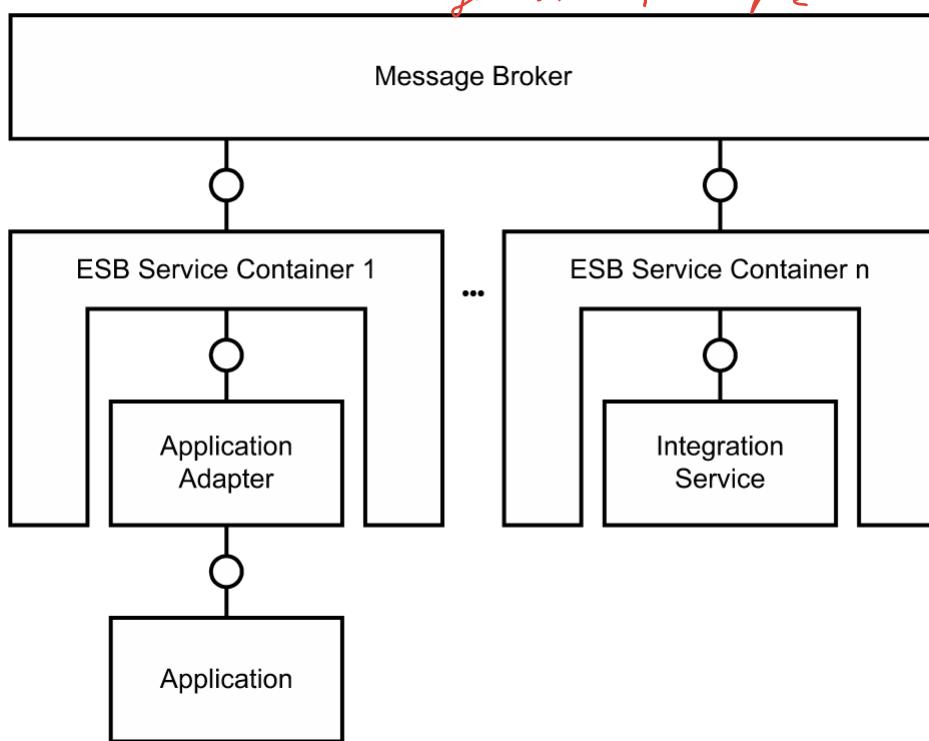


Abbildung 3.4.: Ein einfacher *Enterprise Service Bus* [4]

### 3.2.3. grundlegende Konzepte

*L RxT*

#### 3.2.3.1. Das zentrale Element - Nachrichten

Die Nachricht (*Message*) ist das zentrale Element der Kommunikation im *Messaging System*. Eine Nachricht hat typischerweise folgende Bestandteile: *[3]*

*wer mehrl. was mit der Message?*

- *Header* - enthält Metadaten, wie das *encoding*, *routing information* und sicherheitsrelevante Daten
- *Body* - enthält die eigentlichen Daten

[3]

### 3.2.3.2. Queues

Die *Message Queue* ist ein essenzieller Teil des *Messaging Systems*. Sie bieten die Funktionalität Nachrichten im *Messaging System* zu speichern. Die Sender reihen Nachrichten in die *Queue* ein und die Empfänger nehmen die Nachrichten über die *Queue* entgegen. Die standardmäßigen *Queues* folgen den *First-In First-Out (FIFO)* Prinzip. Die Sender und Empfänger können dabei völlig unabhängig voneinander mit der *Queue* interagieren.

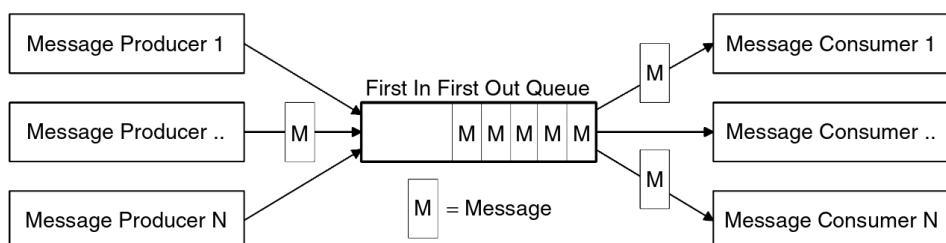


Abbildung 3.5.: Eine *Queue* [1]

Es können viele Parameter von einer *Queue* konfiguriert werden, hierzu zählen unter anderen:

- Der Name der *Queue*
- Die Größe der *Queue*
- Der Sortieralgorithmus für die Nachrichten
- Der Schwellenwert für die Sicherung von Nachrichten

Eine spezielle Art der *Queue* ist die *Dead-Message Queue*, sie beinhalten Nachrichten, welche abgelaufen oder nicht Zustellbar (z.B. nicht gültiger Name für die Ziel-Queue) sind. [1]

### 3.2.4. Komponenten eines Messaging Systems

Die Kernkomponente eines Messaging Systems ist der *Message Broker*, dieser erhält die Nachrichten der *Services* und verschickt sie zu den *Services*. Der *Broker* übernimmt unter anderen folgende Aufgaben:

- Erhalt und Zustellen der Nachrichten
- Speichern der Nachrichten
- *Routing* von Nachrichten

*Wodder?* Der *Cluster* ermöglicht den *Messaging System* ausfallsicher und skalierbar zu sein, da er aus einem Verbund von mehreren *Broker* besteht. Im *Cluster* werden die Nachrichten repliziert und verteilt, dafür gibt es zwei Architekturen:

*Ander Schreiber.* *Single Point of Failure* → *Replizitheit* → *Cluster*

- **Master-Slave** - Die *Broker* sind in *Master* und *Slave* unterteilt. Der *Master* stellt den Dienst zur Verfügung und die *Slaves* dienen nur als Backup.
- **Peer-to-Peer** - Die *Broker* haben alle den selben Status, jede Nachricht ist in mehreren *Brokers* abgespeichert.

Manche *Messaging Systeme* unterstützen auch das Zusammenspiel von mehreren *Clustern*, dass hat folgende Vorteile:

*Relevanz für ReSet?*

- Segregation der Daten
- Isolierung gemäß Sicherheitsanforderungen
- Mehrere Rechenzentren - Notfallwiederherstellung (*disaster recovery*)

[5, 6]

### 3.2.5. Messaging Models

Ein *Messaging System* verwendet eines oder mehrere Kommunikationsmodelle, welche die Grundlage für die Kommunikation der zu verbindenden Systeme definiert. [1]

*Was wird vorgestellt?*

#### 3.2.5.1. Point-to-Point

In einer *Point-to-Point* Kommunikationsmodell gibt es nur einen Sender und einen Empfänger. Im Fall das es mehrere Empfänger für eine bestimmte Nachricht gibt, kann nur ein einziger sie erhalten, solche Empfänger werden auch *Competing Consumers* genannt. Diese Technik kann für eine einfache Art von *Load Balancing* verwendet werden, denn es werden die Nachrichten auf mehrere Empfänger aufgeteilt. [1]

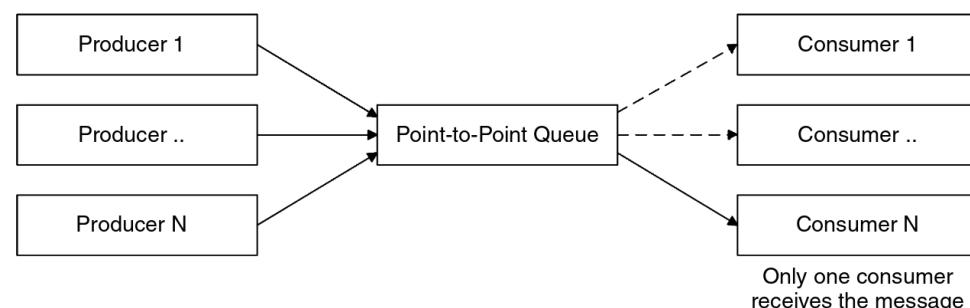


Abbildung 3.6.: Das *Point-to-Point Messaging Model* [1]

*Beispiel?*

#### 3.2.5.2. Publish-Subscribe

*vor Subscriber*

In der Form von *Publish-Subscribe* gibt es mehrere Sender und Empfänger. Die Nachrichten werden immer zu allen Empfängern geschickt. Das *Messaging System* leitet die Nachrichten an *Clients* weiter, basierend auf den *Topics*, für die sie sich angemeldet haben und an denen sie interessiert sind. Empfänger können sich für den Erhalt von den Nachrichten eines bestimmten *Topic* melden (*subscribe*) und Sender können Nachrichten an ein *Topic* senden (*publish*).

*wie erklärt*

Jedoch gibt es keine Einschränkung für die Rolle des *Clients*; ein Client kann ein Empfänger und Sender eines *Topic* sein. [1]

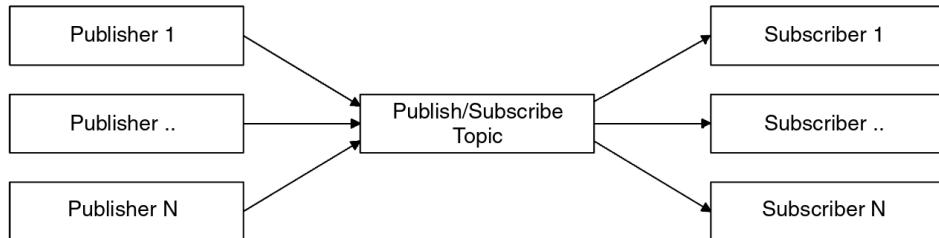


Abbildung 3.7.: Das *Publish-Subscribe Messaging Model* [1]

*Beispiele?*

### 3.2.5.3. Request-Response

In einem Request-Response Kommunikationsmodell gibt es einen Sender und einen Empfänger. Wenn der Sender eine Nachricht schickt, muss er auf die Antwort des Empfängers warten, welches zu einer höheren Kopplung und weiteren Nachteilen führt. Durch diese Nachteile findet das Request-Response Kommunikationsmodell nur in speziellen Fällen Anwendung in *Messaging Systeme*. [1]

*Beispiel?*

### 3.2.6. Consumption Mode

*nicht benötigen und nicht prüfen*

Dieser Aspekt der Kommunikation trifft nur auf die Empfänger zu und beschreibt, wie die Empfänger die Daten erhalten.

#### 3.2.6.1. Push

In diesem Modus verteilt das *Messaging System* selber die Nachrichten an die Empfänger. Dieser Modus hat bessere Echtzeit-Performance, aber benötigt ein Flow-Control Mechanismus. Ein Problem bei diesen Modus ist, dass das *Messaging System* den Empfänger überlasten könnte, wenn in kurzer Zeit sehr viele Nachrichten überträgt. [1, 6]

#### 3.2.6.2. Pull

Im Pull-Modus muss der Nachrichtenempfänger immer beim *Messaging System* nach neuen Nachrichten fragen. Auf diese Weise kann der Empfänger die Verarbeitung von Nachrichten in Übereinstimmung mit seiner eigenen Kapazität steuern. Jedoch muss der Empfänger in einem geeigneten Zeitintervall diese Abfragen durchführen um eine hohe Latenz zu vermeiden. Wenn aber dieses Intervall zu niedrig ist werden die *Message Broker* durch diese vielen Anfragen sehr stark belastet. Darüber hinaus ermöglichen diese Abfragen die Implementierung einer unkomplizierten Form von Lastenausgleich (*Load Balancing*), da jeder Empfänger eine Nachricht nur dann entgegennimmt, wenn er über die entsprechende Kapazität verfügt. [1, 6]

### 3.2.7. Quality-of-Service Garantien

*nicht erwähnt*

Als Middleware System muss das *Messaging System* bestimmte Garantien zusichern, damit eine zuverlässige Operation eines verteilten Systems ermöglicht wird. [6]

#### 3.2.7.1. Zustellungsgarantien

Damit die verlässliche Übertragung zwischen *Consumer* und *Producer* gewährleistet ist.

?

- **At-Most-Once** - Eine *Message* niemals wiederholt übertragen werden, jedoch bei dem Übertragungsvorgang verloren gehen.
- **At-Least-Once**: Eine Nachricht wird niemals nicht vollständig übertragen werden, kann aber wiederholt übertragen werden.
- **Exactly-Once**: Jede Nachricht wird exakt einmal übertragen.

*nix dazwischen*

Die meisten *Messaging Systems* bieten die *At-Most-Once* oder *At-Least-Once* Garantie. Die *Exactly-Once* wird durch ihre Komplexität nur von manchen *Message Systeme* angeboten. [6] *Satz*

#### 3.2.7.2. Ordnungsgarantien

*Relevant für Arbeit?*

Ein weiter wichtige Aspekt ist die Ordnung der Nachrichten, jedoch hat diese Garantie nur bei *Cluster Systemen* Relevanz. Denn in einem *Cluster* ist die Synchronisierung der Ordnung über mehrere *Broker* sehr ressourcenintensiv.

- **No-Ordering**: Es gibt keine Ordnung. Das ist ideal für eine hohe Performance.
- **Partition-Ordering**: Es gibt eine Ordnung für die Nachrichten in einem *Broker*, aber nicht über mehrere *Broker* hinweg.
- **Global-Ordering**: Die Nachrichten sind über den ganzen *Cluster* geordnet, jedoch hat dies eine große negative Auswirkung auf die Performance.

[6]

### 3.2.8. Protokoll

*rl. vor ESB da relevant dafür*

Das verwendete Protokoll, welches die Grundlage der Informationsaustausches bildet, hat eine sehr wichtige Bedeutung, da es die Funktionen des *Messaging Services* definiert und eingrenzt. Die Kommunikationsstandards kann man in zwei Gruppen einteilen in die *open-source* Protokolle und die *closed-source* Protokolle. Folgende weit verbreitete Protokolle werden zu den *open-source* Protokollen gezählt: *AMQP*, *XMPPTCP*, *REST* und *STOMP*. Die *open-source* *Messaging Protocols* sind den *closed-source* Protokollen in dem Aspekt Kopplung überlegen, da sie durch die offene Standardisierung die Abhängigkeit zu einen bestimmten *Messaging System* aufheben. *Closed-source protocols* sind oft Produkte von Veränderungen an existierenden *open-source* Protokollen oder völlig neue geschaffene Kommunikationsdefinitionen, welche nur für bestimmte Systeme gedacht sind. [6]

### 3.2.9. Latenz und Durchsatz

Der Begriff Latenz beschreibt die Zeit, die es benötigt Nachrichten zwischen zwei Endpunkten über *Messaging System* zu übertragen. Die folgenden Punkte haben größten Einfluss auf die Übertragungszeit:

- Die Dauer für die *Metadaten-Verarbeitung* eines Packets, wie z.B. die Validierung oder das *routing*.
- Die Zeit, welche in Anspruch genommen wird für die *Replikation* eines Packets. Dieser Aspekt trifft insbesondere auf *Messaging Systeme*, welche in einer *Cluster-Architektur* arbeiten zu.
- Die *Speicherzugriffsverzögerung*, welche durch die Methode und Ort des Zugriffs bestimmt ist. Zum Beispiel der Zugriff auf *RAM*-gespeicherte Daten sind weniger Zeit intensiv als ein Datenzugriff auf eine Festplatte.
- Weitere Verzögerungen werden durch die Einhaltung der *Quality-of-Service* Garantien erzwungen, wie beispielsweise die Ordungsgarantien, welche einen besonderen Effekt haben.

*Low Latency Messaging Services* minimieren diese Aspekte und können dadurch eine annähernd Echt-Zeit Übertragung ermöglichen. Der Durchsatz steht für die Menge an Daten, welche pro Zeiteinheit über das *Messaging System* übermittelt werden können. Damit die höchsten Durchsatzraten entstehen können müssen die oben angeführten Punkte natürlich minimiert werden, aber es gibt noch einen weiteren Lösungsansatz: *batch processing*. In dieser Methode werden mehrere Nachrichten gesammelt und auf einen Schlag übertragen, denn das ermöglicht den *Overhead* zu reduzieren. Jedoch ist diese Technik bekannt für den Kompromiss zwischen Durchsatz und Latenz. Denn um mehrere Nachrichten auf einmal übertragen zu können, muss gewartet werden bis entsprechend viele Nachrichten angekommen sind. Deswegen kann man bei manchen *Messaging Systemen*, wie Kafka [5], je nach Anwendungsfall bestimmen, wie viele Nachrichten akkumuliert werden. [6]

### 3.3. Vergleich von populären Enterprise Messaging Services

#### Text Einleitung wenn über Kafka und ROLLING

Kafka ist ein hoch skalierbares und verteiltes *Messaging System*, welches bekannt für die Fähigkeit große Datendurchsatzraten erreichen zu können ist. Dieses System wird in der Scala Programmiersprache geschrieben und hat eine sehr aktive Community, die das *open-source* Projekt ständig um Funktionen erweitert. Apache Kafka wird auch als *streaming platform* bezeichnet, da es den benötigten Durchsatzrate erreichen kann und eine Reihe an *stream processing* Funktionalitäten besitzt, wie zum Beispiel *Kafka Streams* oder *KSQL*. *Stream processing* ist die Datenverarbeitung an einen Fluss von Daten, also eine unendliche Serie an **Daten**. Außerdem verfügt Apache Kafka über *Connectors*, die zur Einbindung von externen Datenquellen oder Datensenken gedacht sind. Beispielsweise könnte man Daten über Kafka in einer Datenbank speichern oder über den *MirrorMaker* zwei Kafka *Cluster* verbinden. [7]

Folgende Auflistung zählt die wichtigsten Vorteile von Apache Kafka auf:

- hohe Skalierbarkeit
- hohe Verfügbarkeit
- hohe Durchsatzraten
- Aufbewahrung der Nachrichten auf der Festplatte

warum? wie? zu kurz

Die *disk-based Retention*, also die Speicherung der Nachrichten auf der Festplatte bringt den großen Vorteil mit sich, dass Nachrichten über einen längeren Zeitraum im *Messaging System* aufbewahrt werden können. Da die *Messages* auf der Festplatte persistiert werden und nicht nur im RAM, kann kein Datenverlust auftreten. Es kann zum Beispiel während Wartungsarbeiten an den Empfängern trotzdem sicher gestellt werden, dass keine Nachrichten verloren gehen. [5]

sofort  
daher  
bei  
der  
Garantie  
erhalten  
werden

##### 3.3.1.1. Architektur von Kafka

Grundsätzlich basiert die Architektur von Kafka auf das Messaging Model *Publish-Subscribe*. Die Nachrichten werden in verschiedene *Topics* unterteilt, und jedes *Topic* wird in mehreren Partition aufgeteilt. Diese Partitionen werden auf mehrere *Message Broker* verteilt und repliziert, da Apache Kafka normalerweise als *Cluster* arbeitet. Partitionen werden verwendet um die Nachrichten gleichmäßig auf die *Message Broker* aufzuteilen und den Empfängern gleichzeitiges Auslesen eines *Topics* zu ermöglichen. Wie es in der Abbildung 3.9 zu sehen ist, verwendet Kafka die *Peer-to-Peer-Replikationsmethode* und Empfänger erhalten Nachrichten über den *Consumption Mode Pull*. Die *Consumer* können sich zu Gruppen zusammenschließen (*consumer group*) und können dadurch gemeinsam ein *Topic* verarbeiten. *Consumer Groups* können parallel mehrere Partitionen auslesen (siehe Abbildung 3.8), aber die *Messages* werden auf die Empfänger aufgeteilt, somit bieten sich *consumer groups* perfekt für *Load Balancing* Anwendungen an. *Zookeeper* wird zum Koordinieren der *Consumer* und *Broker* verwendet, damit man in Kafka internes *Load Balancing* erreicht.

Apache Kafka unterstützt die Zustellungsgarantie *at-least-once* standardmäßig und die *at-most-once* Garantie kann beim *Producer* eingestellt werden. Dass die *exactly-once* Zusicherung eingehalten werden kann, müssen aber Veränderungen am Ziel-Storage System bewerkstelligt

nicht erwähnt  
Was ist über?

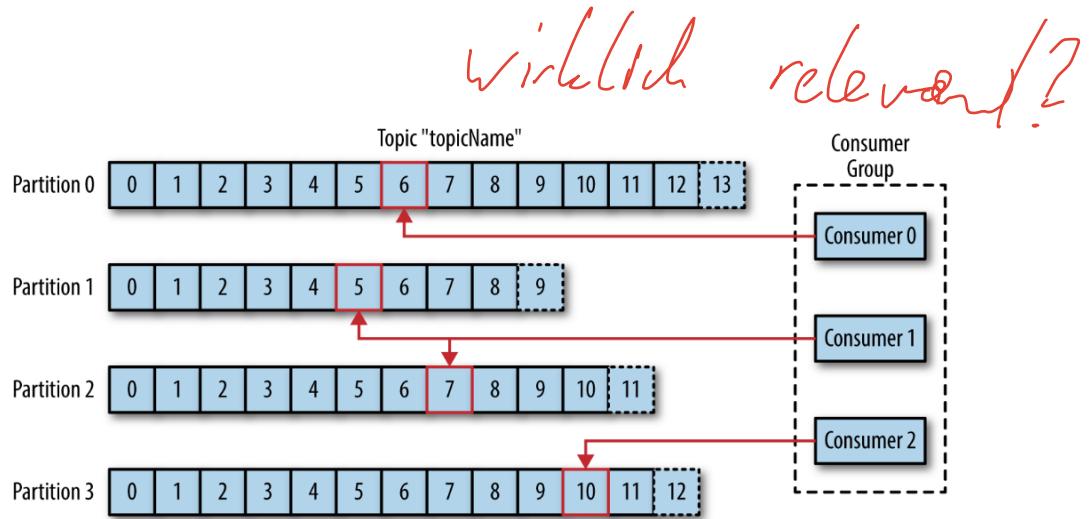


Abbildung 3.8.: Eine *consumer group* verarbeitet gemeinsam ein Topic. [5]

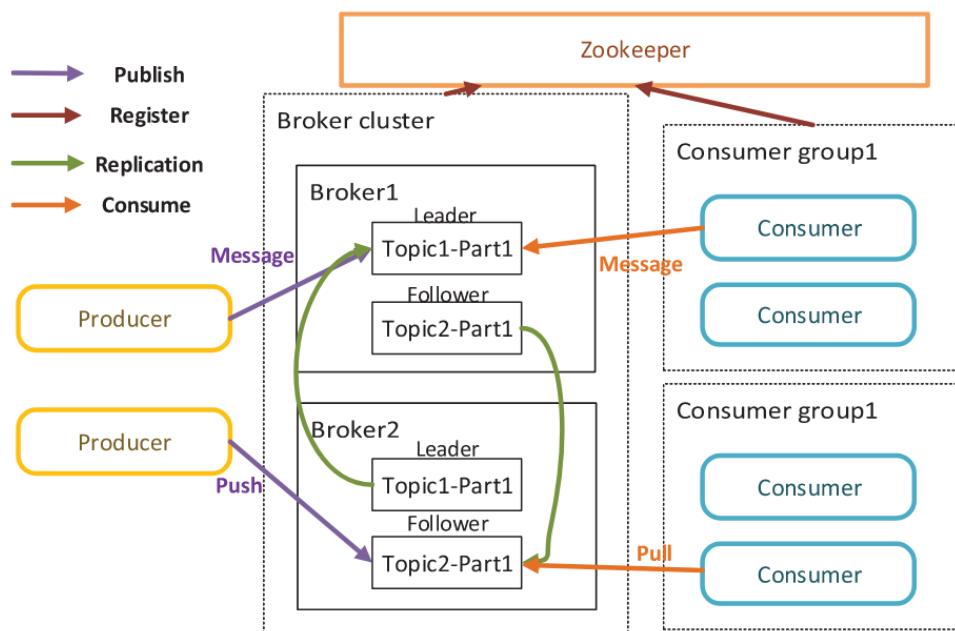


Abbildung 3.9.: Die Architektur des Apache Kafka Systems [6]

werden. Damit Kafka eine hohe Durchsatzrate erreicht werden kann, werden drei Methoden verwendet, die den *Overhead* minimieren:

- Die Nutzung von *Zero-Copy*-Methoden um überflüssiges Kopieren von Daten zu vermeiden.  
*nie erklärt*
- Kafka überträgt Nachrichten in *Batches*. (*batch processing*)
- Nachrichten werden komprimiert um eine effiziente Datenübertragung zu ermöglichen.

### 3.3.2. RabbitMQ

RabbitMQ ist ein *open-source Messaging Service*, welcher um das Protokoll *AMQP* entwickelt wurde. Das System ist mit Erlang programmiert, da Erlang für das Entwickeln von verteilten Systemen geeignet ist, braucht es keine koordinierende Komponente wie *Zookeeper*. In der Abbildung 3.10 ist zu sehen, dass RabbitMQ auch als *Cluster* arbeiten kann, dafür werden die *Queues* auf mehrere *Broker* repliziert. Es gibt je *Queue* eine *Master-Queue* und eine oder mehrere *Backup-Queues*. Alle Operationen starten bei dem *Master* und werden zu den *Backup-Queues* weitergeleitet. Aber wenn die *Master-Queue* ausfällt wird sofort eine *Backup-Queue* zur *Master-Queue* aufsteigen und somit ist eine hohe Ausfallsicherheit gegeben. Jedoch hat RabbitMQ kein gutes Design für Skalierbarkeit, weil die komplette Replikation der *Queues*, ist für hohe Auslastungen nicht gut geeignet. *Exchanges* sind für das *Routen* der Nachrichten benötigt, sie können komplexe *Routing-Anforderungen* umsetzen, wie zum Beispiel die Verteilung der Nachrichten auf mehrere *Queues*. Das RabbitMQ System ist ein *general-purpose Message Broker*, da es sehr viele Funktionalitäten implementiert, darunter befinden sich:

- *Push / Pull Consumption Mode*
- verschiedene standardisierte Protokolle wie *AMQP*
- *Point-to-Point* und *Publish-Subscribe Messaging Models*
- *Disk Nodes* oder *Memory Nodes* für *RabbitMQ Cluster*
- *At-least-once* oder *at-most-once* Zustellungsgarantien

*nice!*

RabbitMQ glänzt in den vielseitigen Funktionalitäten, die es anbietet, jedoch kann es mit der hohen Skalierbarkeit und Performance, insbesondere dem Datendurchsatz von Apache Kafka nicht mithalten. [6, 3]

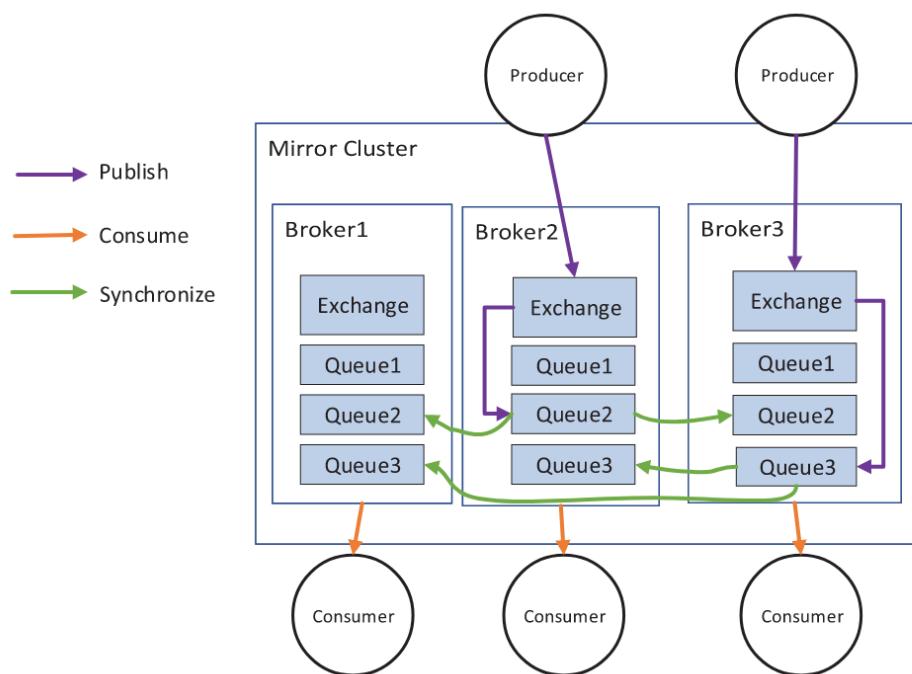


Abbildung 3.10.: Die Architektur des RabbitMQ Systems [6]

## 4. Visualisierung

*Kleine gehe Einleitung / Relevanz*

Die Repräsentation von Objekten aus der realen Welt wird mittels Abstraktion und Filterung der essenziellen Merkmale deutlich sparsamer in Bezug auf Datenspeicherung. Die Visualisierung dieser abstrahierten Daten ist definiert als eine ansprechende Darstellung, um eine benutzerfreundliche Interaktion zu ermöglichen. Im Kontext der Effizienz ist die Aufgabe der Visualisierung eine nahtlose Verbindung zwischen der Datenspeicherung und dem Endbenutzer mit einer hohen Performance herzustellen.

*Visualisierung ver Datei oder UI?*

*Was kommt in f.?*

### 4.1. Benutzerfreundlichkeit und Performance

Die Wirksamkeit einer Anwendung hängt nicht nur von deren Benutzerfreundlichkeit, sondern auch von deren Performance ab. Benutzerfreundlichkeit (auch Usability) ist die Leichtigkeit, mit den Daten interagieren zu können, und orientiert sich im Idealfall an den Denkweisen des Benutzers [8]. Die Effektivität einer Applikation wird demnach stark von ihrer intuitiven Benutzerbarkeit beeinflusst [9].

*Über (pi)ug zu 4.1.1 +*

*4.1.2*

#### 4.1.1. Intuitivität

Benutzer manifestieren die Erwartung, Applikationen ohne Konsultation von Gebrauchsanweisungen und Handbüchern bedienen zu können [10]. Diese Eigenschaft einer Anwendung nennt sich Intuitivität und ist ein Grundsatz moderner Programmsysteme. Intuitivität minimiert die Lernmenge initiativ erheblich.

Beispielsweise weist eine intuitive Navigation eine klare und deutliche Aufteilung der verschiedenen Bereiche auf. Somit können Benutzer schnell zu den gewünschten Bereichen gelangen, was wiederum den logischen Fluss der Applikation verbessert.

*c i  
Über (pi)ug  
check* [ ] *Quellen* [ ]

Konsistente Designmuster fördern die Vertrautheit des Benutzers mit dem Programm. Die Benutzererfahrung (UX) wird demnach wesentlich gesteigert. Im Laufe der Zeit haben sich aufgrund der schieren Menge an Webseiten die bewährtesten Designmuster herauskristallisiert. Im Bezug auf Visualisierungsmethoden zählen dazu neben Achsenbeschriftungen, Legenden bei Diagrammen und Symbolen für Sortierungsmöglichkeiten bei Tabellen auch normierte Beschriftungen der Daten in Graphen und auf den ersten Blick erkennbare Zeilen und Spalten von Matrizen. Durch die konsequente Anwendung solcher Muster werden Ästhetik und Benutzerfreundlichkeit verbessert.

Ein weiterer wichtiger Bestandteil einer barrierefreien Benutzeroberfläche sind Icons, da diese Informationen visuell vermitteln und nicht an eine bestimmte Sprache gebunden sind. Somit müssen Benutzer die Systemsprache nicht zwingen verstehen, um eine Anwendung zu bedienen. Dies ist besonders bei Anwendungen für ein internationales Publikum von großer Bedeutung. Eine hohe Priorität hat deswegen die überlegte Auswahl eines simplen Iconsets.

Ein weit verbreiteter Mythos des Navigationsdesigns ist die „Three-Click Rule“. Die Grundrichtlinie dieser Regel beschränkt die Anzahl an Klicks oder Eingaben jeglicher Art auf drei, um zu jedem Bereich im Programmsystem zu gelangen. Jedoch stellte sich bei detaillierteren Untersuchungen heraus, dass die Anzahl von Klicks alleine weder Einfluss auf die Benutzerzufriedenheit noch die Erfolgsrate der Applikation hat [11, 12]. Kritik äußert sich

demnach aufgrund der Vernachlässigung der kontextuellen Komplexität und individuellen Benutzerziele.

Eine ganzheitliche Bewertung der Benutzererfahrung erfordert eine Berücksichtigung verschiedener Faktoren, wie Effizienz, Effektivität, Konsistent und Zufriedenheit der Benutzer. Schwerpunkt jedes Designs sollte das Schaffen einer Applikation sein, welche aufgrund der Benutzerorientierung das Arbeiten mit abstrahierten Informationen der realen Welt verschnellert und erleichtert.

„je beschrieben“

#### 4.1.2. Interaktivität

Interaktivität beschreibt eine fortlaufende Kommunikation von Informationen zwischen dem Benutzer und der Anwendung, welche wiederum ständig mit dem Backend kommuniziert. Interaktive Elemente können von einfachen Buttons, Slidern oder Togglen bis hin zu Einstellungen (Filterung, Sortierung, Editierung) von Tabellen, Skalieren von Diagrammen und Herumschieben der Knoten eines Graphen reichen. Interaktivität ist demnach eine entscheidene Eigenschaft jeglicher Anwendungen, um eine positive Benutzererfahrung zu verstärken.

«Interaktivität ist das Potenzial eines technischen Einzelmediums oder einer Kommunikationssituation, das interaktive Kommunikation begünstigt, also den Prozess der Interaktion.» - *Christoph Neuberger* [13]

Die Integration effektiver Filter- und Suchfunktionen in Datenvisualisierungssystemen spielt eine Schlüsselrolle bei der Optimierung der Benutzererfahrung und der gezielten Extraktion relevanter Informationen. Untersuchungen haben deutlich gemacht, dass die gezielte Implementierung von sorgfältig gestalteten Filteroptionen in Datenvisualisierungssystemen die Nutzerbefähigung zur gezielten Suche nach spezifischen Datenpunkten oder Mustern in hohem Maße verbessert. Dies resultiert wiederum in einer signifikanten Steigerung der Effizienz innerhalb der Analyseprozesse [14]. ✓ vs Pkt vs Randbedingungen

Eine wichtige Komponente ist die Anpassungsfähigkeit der Filter, um unterschiedlichen Anforderungen gerecht zu werden. Die Möglichkeit, Filterkriterien zu kombinieren oder anpassbare Parameter festzulegen, ermöglicht eine feinere Steuerung und eine präzisere Datenauswahl [15]. Dies trägt dazu bei, dass Benutzer ihre Anfragen flexibel an die Komplexität des Datensatzes anpassen können. ✅ wirklich so?

Forschungen von Schneiderman et al. betonen die Bedeutung von „Dynamic Queries“, einer Technik, bei der Benutzer unmittelbares Feedback zu ihren Filteranfragen erhalten [16]. Diese Echtzeit-Rückmeldung erleichtert Benutzern eine explorative Analyse. Des Weiteren haben Studien gezeigt, dass die Integration von Vorschlägen während der Eingabe (Autovervollständigung) und die Verwendung von Synonymen die Benutzerfreundlichkeit der Suchfunktionen verbessern [14]. Diese Erkenntnisse verdeutlichen die Relevanz einer kontinuierlichen Forschung und Weiterentwicklung von Filter- und Suchmechanismen, um den sich ständig ändernden Anforderungen der Benutzer gerecht zu werden.

#### 4.1.3. Performance

Die Effizienz von Webanwendungen spielt eine entscheidende Rolle in der Gestaltung einer ansprechenden Benutzererfahrung. Zahlreiche Studien haben sich mit der Leistungsoptimierung von Webseiten befasst und zeigen, dass eine schnellere Ladezeit einen unmittelbaren Einfluss auf die Zufriedenheit und Interaktion der Nutzer hat. In diesem Zusammenhang präsentiert eine umfassende Untersuchung, wie die Minimierung von HTTP-Anfragen, das effiziente Caching von Ressourcen und die Reduzierung von DNS-Lookups als kritische Elemente für die Optimierung der Webseitenleistung [17].

Besonders relevant für die Verbesserung der Benutzerfreundlichkeit sind Optimierungstechniken, wie „Lazy Loading“ und „Indexing“. Lazy Loading ermöglicht es, Ressourcen nur dann zu laden, wenn sie vom Nutzer angefordert werden, was die Interaktivität beschleunigen kann [18, 19]. In Bezug darauf zeigen Forschungen auf, dass eine effiziente Indexierung von Inhalten die Navigation und den Zugriff auf relevante Informationen für die Nutzer erheblich verbessert [20, 21].

*meinst du die oben beschriebenen?*

Diese Erkenntnisse aus aktuellen Studien betonen die Bedeutung von Leistungsoptimierungs-techniken in der Webentwicklung und verdeutlichen, wie diese direkte Auswirkungen auf die Benutzerfreundlichkeit haben können. Die ganzheitliche Berücksichtigung von Aspekten wie Minimierung von HTTP-Anfragen, effizientes Ressourcen-Caching, Lazy Loading und Indexing kann somit als Schlüssel zur Schaffung einer optimalen Nutzererfahrung dienen.

### 4.2. Arten von Datendarstellungen

Die Visualisierung von hochvernetzten Daten ist heutzutage von zentraler Bedeutung, um die Herausforderungen bei der umfassenden Analyse und Deutung dieser komplexen Datenstrukturen zu bewältigen. Datengefüge solcher Art, die sich beispielsweise in sozialen Netzwerken, biologischen Systemen, Nahrungsketten, Dateisystemen oder sogar in der Struktur der Sprache, wie dem Alphabet [22], manifestieren, erfordern spezialisierte Visualisierungstechniken zur Extraktion ihrer inharenten Muster und Strukturen.

Ein weiterer entscheidender Aspekt ist die effiziente Analyse von Daten nach ihrer Wichtigkeit. In Zeiten der ständig wachsenden Datenflut ist es für Datenanalysten von besonderer Relevanz, spezifische Merkmale schnell zu identifizieren. Die Sortierung von Daten nach ihrer Bedeutung ermöglicht nicht nur eine präzise Fokussierung auf relevante Informationen, sondern unterstützt auch die Bewältigung der ständig verändernden Datenlandschaft. Diese gezielte Herangehensweise erleichtert es Datenanalysten, inmitten der Datenflut diejenigen Informationen hervorzuheben, die für die Analyse und Interpretation von höchster Relevanz sind.

*Beschreibung was in dem Kapitel kommt +*

#### 4.2.1. Zweidimensionale (2D) und Dreidimensionale (3D) Darstellungen

In der Vergangenheit beschränkte sich die Möglichkeit der visuellen Darstellung auf Papier auf zweidimensionale Visualisierungen. Heutzutage hingegen ermöglichen Computerbildschirme und moderne Programme die einfache Illustration von dreidimensionalen Darstellungen (beispielsweise Spline [23]). Zukünftige Fortschritte könnten die Verwendung von Hologrammen

*Wie passt diese Kapitel zu ?*

*Einsände in Arbeit wenn so angeführt*

ermöglichen, um Visualisierungen noch anschaulicher und logischer darzustellen und somit eine verbesserte Verständlichkeit von Daten für den Menschen zu fördern. *Quelle?*

Gegenwärtig fertigen zahlreiche Designer trotz technologischer Fortschritte nach wie vor Grafiken in zweidimensionaler Form an. Dies röhrt daher, dass sie über bewährte Kompetenzen in der Erstellung von Diagrammen und ähnlichen Elementen verfügen und dabei effizient agieren können. Gleichzeitig ist zu beobachten, dass aufgrund der kontinuierlich steigenden Rechenleistung von relativ kostengünstigen Grafikkarten vermehrt virtuelle Grafiken in dreidimensionaler Form erstellt werden. Diese Zunahme erfolgt jedoch nicht immer aufgrund zielgerichteter Überlegungen.

«Because it is so inexpensive to display data in an interactive 3D virtual space, people are doing it - often for the wrong reasons.» - *Colin Ware* [24]

In Fällen, in denen dreidimensionale Visualisierungen demnach keine substantielle Verbesserung für die effiziente Übermittlung relevanter Informationen an den Nutzer bieten, wird davon tendenziell abgeraten. Dies begründet sich unter anderem damit, dass Nutzer bereits an die vorherrschende Verwendung zweidimensionaler Elemente im Webdesign gewöhnt sind. Aus diesem Grund beschränkt sich der weitere Verlauf dieser Arbeit ausschließlich auf die Analyse und Erörterung zweidimensionaler Visualisierungsmethoden, da diese in den etablierten Konventionen des Webdesigns verankert sind und als Grundlage für eine vertraute Benutzererfahrung dienen. *Quelle?*

#### 4.2.2. Tabellen

Die Darstellung von Daten in tabellarischer Form stellt eine essenzielle Methode dar, um komplexe Informationen übersichtlich zu präsentieren. Tabellen ermöglichen die strukturierte Anordnung von Daten in Zeilen und Spalten, wodurch eine klare und leicht verständliche Organisation entsteht. Jede Zelle innerhalb der Tabelle kann dabei eine spezifische Eigenschaft eines Dateneintrages beziehungsweise eine Relation zu anderen Datensätzen repräsentieren. Besonderheit einer Tabelle ist, dass jeder Eintrag die gleichen Eigenschaften aufweisen muss, andernfalls hätte die Tabelle viele leere Felder. Die effektive und effiziente Darstellung dieser Daten erfordert eine intuitive Filterung und logische Anordnung der Daten.

«Tables are an arrangement of words, numbers, or signs, in parallel columns, used to depict data or relationships.» - *Sandeep B Bavdekar* [25]

Nutzername	E-Mail	Rollen
MaxMustermann	max.mustermann@email.com	Administrator, Benutzer
PeterPan	peter.pan@email.com	Gast
LauraDeveloper	laura.developer@email.com	Entwickler, Benutzer
AlexHacker	alex.hacker@email.com	IT-Spezialist, Benutzer
SophiaDesigner	sophia.designer@email.com	Designer, Benutzer
MiaCoder	mia.coder@email.com	Entwickler, Benutzer

Abbildung 4.1.: einfache Tabelle

*eher Grid als Date darstellung?*

#### 4.2.2.1. Sortierung

Um Datenzusammenhänge verständlicher zu machen, werden die Datenpunkte in Tabellen nach identifizierenden Spalten sortiert. Das bedeutet, dass beispielsweise die Eigenschaft „Name“ ausschlaggebend für die alphabetische Sortierung aller Einträge ist. Bei Statistiken ist häufig eine Sortierung nach Ergebnissen üblich, um eine absteigende oder aufsteigende Repräsentation der Prozentsätze oder dergleichen zu erhalten.

Eine statische und intuitive Sortierung fördert die Benutzerfreundlichkeit, da die Vertrautheit mit der Tabelle der Applikation erhöht wird. Deutlich vorteilhafter sind dynamische und interaktive Sortierungen, welche die Interaktivität und somit die Effizienz der Applikation verbessern. Dynamische Sortierungen überlassen dem Benutzer die Entscheidung, nach welcher Spalte die Tabelle und ob dies auf- oder absteigend sortiert werden soll [26]. Diese erweiterte Funktionalität der Sortierung ermöglicht eine effiziente Analyse der Zusammenhänge aller Dateneinträge.

Nutzername ^	E-Mail	Rollen
AlexHacker	alex.hacker@email.com	IT-Spezialist, Benutzer
LauraDeveloper	laura.developer@email.com	Entwickler, Benutzer
MaxMustermann	max.mustermann@email.com	Administrator, Benutzer
MiaCoder	mia.coder@email.com	Entwickler, Benutzer
PeterPan	peter.pan@email.com	Gast
SophiaDesigner	sophia.designer@email.com	Designer, Benutzer

Abbildung 4.2.: nach Nutzernamen aufsteigend sortierte Tabelle

Viel  
Platz  
wenig  
Ausgabekraft

Nutzername ^	E-Mail	Rollen
SophiaDesigner	sophia.designer@email.com	Designer, Benutzer
PeterPan	peter.pan@email.com	Gast
MiaCoder	mia.coder@email.com	Entwickler, Benutzer
MaxMustermann	max.mustermann@email.com	Administrator, Benutzer
LauraDeveloper	laura.developer@email.com	Entwickler, Benutzer
AlexHacker	alex.hacker@email.com	IT-Spezialist, Benutzer

Abbildung 4.3.: nach Nutzernamen absteigend sortierte Tabelle

#### 4.2.2.2. Filterung

Das schnelle Ausfindigmachen von Datensätzen ist eine essenzielle Funktion zur Steigerung der Effizienz [27]. Filtermöglichkeiten wie zum Beispiel das Suchen nach Begriffen innerhalb der Tabelle oder die Einschränkung der Dateneinträge auf bestimmte kategoriale Eigenschaften haben aufgrund der häufigen Vorkommnisse in Anwendungen bereits vorgefertigte UI-Elemente in fast allen Frameworks. Gemeinsam mit Sortierungsmöglichkeiten bieten diese Funktionalitäten eine äußerst effiziente Arbeitsumgebung, um selbst die komplexesten Daten verständlich darzustellen.

Filterfunktionalitäten variieren oft in der Häufigkeit der Aktualisierung der Daten. Besonders effiziente Systeme und Client-seitige Filtersysteme können sich eine sofortige, unverzögerte

Aktualisierung der Daten bereits beim Eintippen des Suchbegriffes erlauben. Jedoch setzen viele Applikationen auf eine etwas zeitverzögerte Filterung, um die Anzahl der Serveranfragen zu vermeiden. Die dritte Möglichkeit ist das Aktualisieren beim Klicken auf eine Taste, entweder in der Weboberfläche oder der Tastatur.

Entwickler		
Nutzername	E-Mail	Rollen
LauraDeveloper	laura.developer@email.com	Entwickler, Benutzer
MiaCoder	mia.coder@email.com	Entwickler, Benutzer

Abbildung 4.4.: Tabelle mit Filterfunktionalität (unverzögert)

Entwickler		
Nutzername	E-Mail	Rollen
LauraDeveloper	laura.developer@email.com	Entwickler, Benutzer
MiaCoder	mia.coder@email.com	Entwickler, Benutzer

Abbildung 4.5.: Tabelle mit Filterfunktionalität (Tasteneingabe)

#### 4.2.2.3. Paginierung

*Performance Verzögerung?*

Die Interaktion mit Tabellen wird erschwert, wenn die Tabelle viele Datensätze enthält und der Nutzer lange Zeit scrollen muss. Um die Anzahl an Datensätzen zu limitieren und gleichzeitig keine Funktionalitäten der Tabelle zu verlieren, gibt es die Paginierung. Sie unterteilt die schiere Datenmenge in geordnete Seiten, welche jeweils eine maximale Anzahl an Elementen beinhalten. Diese Seiten können anschließend mittels Nutzerinteraktion der Reihe nach aufgerufen werden. Als Vergleich kann man sich ein Buch mit Seiten vorstellen, welche horizontale durchblättert werden. Ein bekanntes Anwendungsbeispiel ist die Google Suche, welche ebenfalls nur eine limitierte Anzahl an Suchergebnissen anzeigt und weitere Ergebnisse auf den nachfolgenden Seiten der Paginierung versteckt. Erst nachdem der Nutzer das Laden der zweiten Seite mittels Interaktion angefordert hat, werden diese Datensätze angezeigt. Aufgrund der häufigen Implementierung der Paginierung bei vielen UI-Libraries, ist diese eine bewährte Methode zur Steigerung der Benutzerfreundlichkeit. [28]

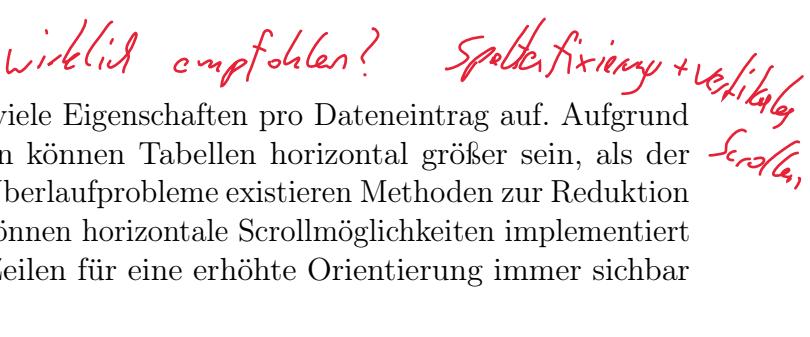
Die Vorteile der Paginierung beschränken sich nicht nur auf die verringerte Zeit, welche beim vertikalen Scrollen benötigt wird. Studien bezüglich Benutzerverhalten bei Internetsuchen haben gezeigt, dass horizontales Scrollen (Paginierung) den Nutzer dazu bewegt, weniger Zeit auf der Suchergebnisseite und mehr Zeit auf den tatsächlichen Webseiten verbringt. Ein weiterer, noch viel bemerkenswerter Vorteil ist die Schnelligkeit beim Finden der gesuchten Information unter Verwendung von Paginierung [29]. Auch auf mobilen Geräten ist die Suchgenauigkeit höher, wenn Paginierung verwendet wird. Außerdem wird der Inhalt, welcher sich erst nach dem Scrollen zeigt, besser vom Benutzer aufgenommen, wenn horizontales Scrolling verwendet wird.

*hier können viel mehr  
Fälle nachher  
gezeigt werden  
siehe  
Mobiles  
Telefon*

Bei unvorstellbar großen Datenmengen und dementsprechend steigender Seitenanzahl kann auch eine logarithmische Paginierung verwendet werden. Hierbei werden die Interaktionsmöglichkeiten eben nicht auf lineare Seitensprünge limitiert, sondern erlauben ein schnelles Navigieren zu hohen Seitenzahlen. [30]

#### 4.2.2.4. horizontaler Platzmangel

Komplexe Daten weisen überlicherweise viele Eigenschaften pro Dateneintrag auf. Aufgrund des begrenzten Platzes auf Bildschirmen können Tabellen horizontal größer sein, als der verfügbare Platz. Zur Vermeidung dieser Überlaufprobleme existieren Methoden zur Reduktion der Größe einer Tabelle [25]. Außerdem können horizontale Scrollmöglichkeiten implementiert werden, wobei bestimmte Spalten und Zeilen für eine erhöhte Orientierung immer sichtbar sein sollten.



Nutzername	E-Mail
MaxMustermann	max.mustermann@email.com
PeterPan	peter.pan@email.com
LauraDeveloper	laura.developer@email.com
AlexHacker	alex.hacker@email.com
SophiaDesigner	sophia.designer@email.com
MiaCoder	mia.coder@email.com

Nutzername	Rollen
MaxMustermann	Administrator, Benutzer
PeterPan	Gast
LauraDeveloper	Entwickler, Benutzer
AlexHacker	IT-Spezialist, Benutzer
SophiaDesigner	Designer, Benutzer
MiaCoder	Entwickler, Benutzer

Abbildung 4.6.: aufgeteilte Tabelle aufgrund Platzmangels

#### 4.2.2.5. Anwendungsfälle bei stark vernetzten Daten

Tabellen eignen sich besonders gut, um komplexe, stark vernetzte Daten darzustellen, wenn diese in eine zweidimensionale Struktur eingebettet werden können. Jeder Eintrag sollte im Idealfall die gleichen Eigenschaften (Spalten) aufweisen, um leere Zellen zu vermeiden. Die Verwendung dieser visuellen Darstellungsformen ermöglicht eine übersichtliche und anschauliche Repräsentation von Informationen, was die Interpretation und Extraktion von Mustern erleichtert. Weil Tabellen meistens zweidimensional verwendet werden, können sowohl numerische als auch kategoriale Eigenschaften optimal verglichen werden.

*Tabelle → Prototypen für Date darstellung*

### 4.2.3. Diagramme

Die Präsentation von Daten durch Diagramme stellt eine weitere bedeutende Methode dar, um komplexe Informationen visuell aufzubereiten. Im Gegensatz zu Tabellen verwenden Diagramme grafische Elemente wie Linien, Balken oder Kreise, um Beziehungen, Muster und

Trends zwischen Datenpunkten zu verdeutlichen. Diagramme haben den Vorteil, dass sie es ermöglichen, Zusammenhänge auf einen Blick zu erfassen und die visuelle Wahrnehmung des Betrachters zu nutzen. Sie eignen sich besonders gut zur Illustration von Veränderungen über die Zeit oder zum Vergleich von Anteilen innerhalb eines Gesamtkontexts.

Diagramme haben im Gegensatz zu Tabellen den Nachteil, numerische Daten nicht exakt anzuzeigen, während Tabellen die intuitive Visualisierung vernachlässigen. Aus diesem Grund eignen sich Diagramme, um einen schnellen Überblick über die aktuellen Daten zu erhalten. Tabellen ermöglichen das genaue Analysieren der Datensätze. Moderne Libraries stellen jedoch bereits Diagramme zur Verfügung, welche genaue Datenanalysen mittels Interaktivität des Diagramms ermöglichen. Beispielsweise können exakte Datenwerte mittels Mausinteraktionen spezifisch angezeigt werden. Somit erhält man beide Vorteil, sowohl ein effizienter Einstieg als auch eine detaillierte Analyse, in Diagrammen.

#### 4.2.3.1. Mengen in linearen Diagrammen

Bestehende Forschungen haben bereits eine optimale Visualisierung von Mengen mittels linearen Diagrammen erforscht und dabei herausgefunden, dass grafische Elemente, wie zum Beispiel Farbe und Größe der Linien, einen entscheidenden Einfluss auf die Effektivität der Lesbarkeit des Diagramms hat. Außerdem wurde in den Studien festgestellt, dass eine minimale Anzahl an Liniensegmenten, Hilfslinien zwischen den überschneidenden Bereichen und die Dicke der Linien jeweils unterschiedlich ausgeprägte Einflüsse auf Bearbeitungszeit und Fehlerquote haben. [31]

*Bilder / Beispiele      Mehr Inhalt*

#### 4.2.3.2. UML Klassendiagramme

*→ wie passt das darin?*

Um die Effizienz beim Arbeiten mit Klassendiagrammen zu erhöhen haben Gutowenger et al. die Software GoVisual programmiert, dessen Hauptziele die Minimalisierung von Überschneidungen und Biegungen von Linien und Kanten, einheitliche Richtung innerhalb jeder Klassenhierarchie und eine gute Kantenbeschriftung sind. Dabei wurde ein Algorithmus entwickelt, welcher die gemischt-aufwärts planarisierte Darstellung und das orthogonale Layout berechnet, um diese Optimierungsmaßnahmen und darausfolgend die Effizienz des UML Diagramms sicherzustellen. [32]

#### 4.2.4. Graphen

*Gitterordnung      Diagramme zu Graphen nicht passend  
Übergang fehlt*

Graphen sind abstrakte mathematische Strukturen, die aus einer Menge von Knoten und den dazwischen verlaufenden Kanten bestehen. Jede Kante in einem Graphen repräsentiert eine Verbindung zwischen genau zwei Knoten. Die wissenschaftliche Disziplin, die sich mit der Untersuchung und Analyse solcher Graphen befasst, ist als Graphentheorie bekannt. In der Graphentheorie werden unterschiedliche Eigenschaften und Charakteristika von Graphen erforscht, wodurch sie als mächtiges Werkzeug in verschiedenen wissenschaftlichen, informatischen und ingenieurwissenschaftlichen Anwendungen dient. [33]

## 4.2.4.1. Ausrichtung

*Teil überprüfen nicht zusammenhängend speziell ab 4.2.4.8*

Die Besonderheit eines gerichteten Graphen ist die eindeutige Vorgabe der Kantenrichtung. Innerhalb dieses Graphen verbindet jede Kante präzise einen Anfangs- mit einem Endknoten und wird durch Pfeile repräsentiert, die eindeutig die Richtung der Beziehung zwischen diesen Knoten anzeigen. In der realen Welt könnten derartige gerichtete Verbindungen exemplarisch in einem sozialen Netzwerk und seinem „Following“-System auftreten, da die Möglichkeit besteht, dass eine Person A einer Person B folgt, jedoch nicht zwangsläufig umgekehrt. [33]

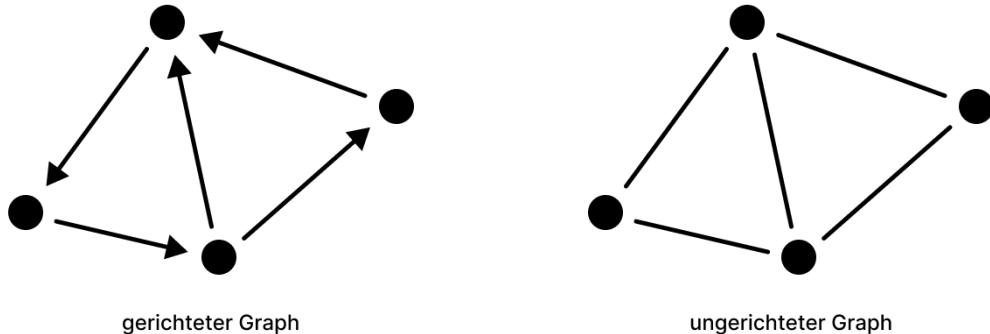


Abbildung 4.7.: Ausrichtung

## 4.2.4.2. Konnektivität

Zusammenhängende Graphen haben die Eigenschaft keine Knoten aufzuweisen, welche vom Rest des Graphen isoliert sind. Diese Charakteristik impliziert das Fehlen isolierter Teilgraphen innerhalb des Gesamtgefüges. Die Gewährleistung der Zusammenhangseigenschaft bedeutet, dass sämtliche Knoten durch Pfade miteinander verbunden sind, wodurch der Graph als kohärente und nicht in isolierte Teile zerlegbare Einheit betrachtet wird. Nicht zusammenhängende Graphen können an ihren isolierten Knoten erkannt werden. [34]

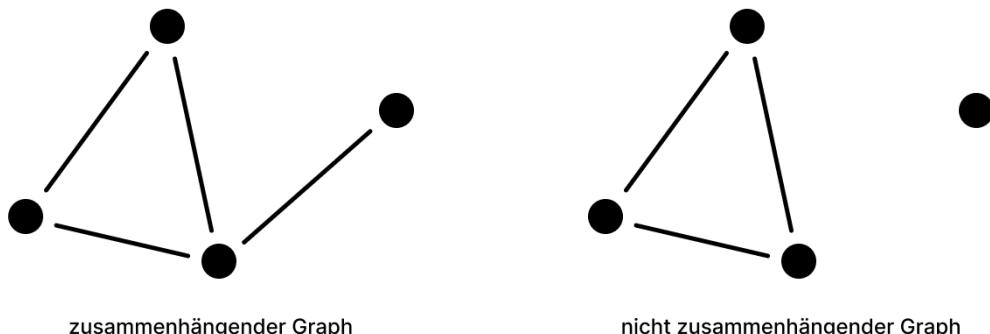


Abbildung 4.8.: Konnektivität

#### 4.2.4.3. Zyklen

Ein zyklischer Graph manifestiert sich durch die Existenz mindestens eines geschlossenen Pfades innerhalb seiner Struktur. Ein Pfad in diesem Sinne definiert sich als Abfolge von Kanten, dessen Anfangsknoten gleich dem Endknoten ist. Azyklisch wird ein Graph genannt, wenn seine Eigenschaften dem Gegenteil eines zyklischen Graphen entsprechen.

Die Ausführung von Algorithmen auf zyklischen Graphen erfordert besondere Achtsamkeit, da Zyklen potenziell zu komplexen, aufeinander abhängigen Situationen führen können. Das Ignorieren dieser Zyklen birgt das Risiko, dass solche Abhängigkeiten nicht aufgelöst werden. Daher ist eine präzise Analyse und Integration der zyklischen Strukturen in den algorithmischen Prozess unerlässlich, um die korrekte Verarbeitung von Daten und Abhängigkeiten zu gewährleisten. [33]

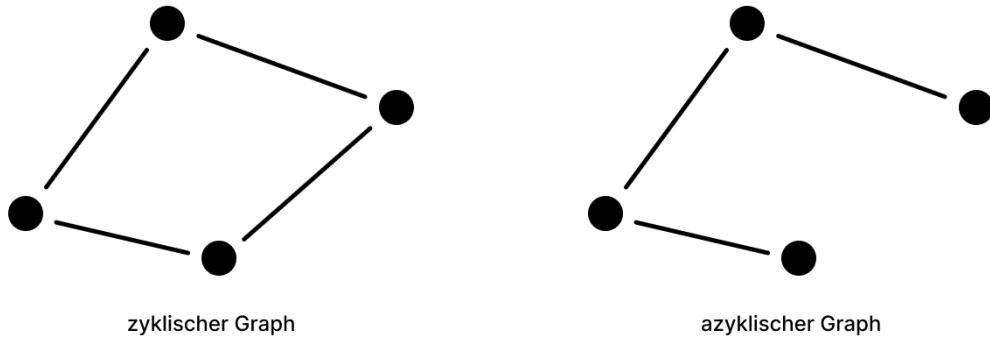


Abbildung 4.9.: Zyklen

#### 4.2.4.4. Gewicht

In einem gewichteten Graphen werden den Kanten zusätzliche numerische Werte zugeordnet, die als Gewichte bezeichnet werden. Diese Gewichte dienen dazu, verschiedene Arten von Beziehungen oder Kosten zwischen den Knoten zu repräsentieren. Die Gewichtung ermöglicht eine präzise Modellierung von unterschiedlichen Einflussstärken oder Ressourcenverbrauch entlang der graphentheoretischen Struktur, wie zum Beispiel Dauer und Länge einer Verbindungsstrecke zweier Haltestellen bei einem öffentlichen Verkehrsnetzwerk. [33]

#### 4.2.4.5. Vollständigkeit

Ein vollständiger Graph ist durch die Eigenschaft gekennzeichnet, dass jeder Knoten mit jedem anderen Knoten durch eine Kante verbunden ist. Diese charakteristische Vollständigkeit der Verbindungen impliziert eine maximale Interaktion zwischen den einzelnen Knoten des Graphen, wodurch sämtliche möglichen Kantenrelationen realisiert sind. Diese Einschränkung der Flexibilität von den Eigenschaften eines Graphen kommen aus diesem Grund seltener vor, weswegen die meisten Graphen unvollständig sind. [33]

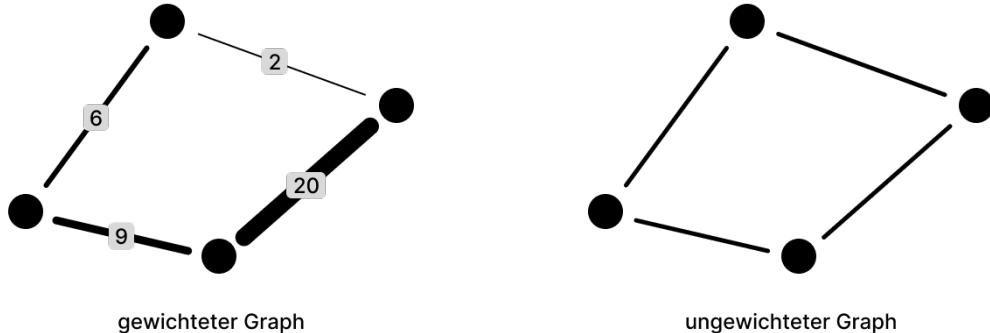


Abbildung 4.10.: Gewicht

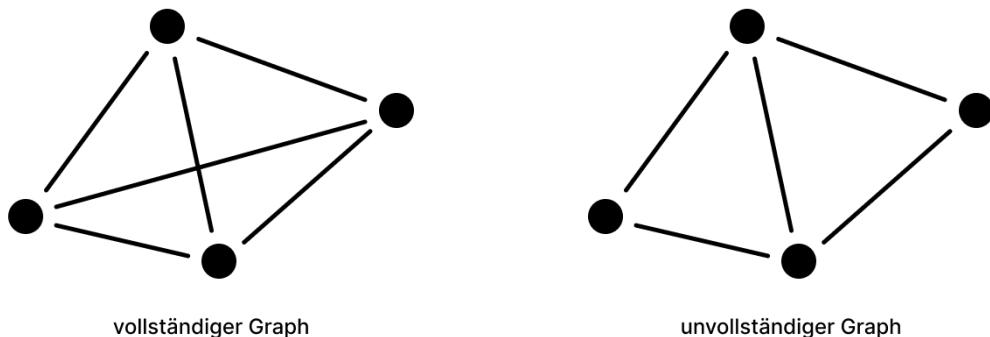


Abbildung 4.11.: Vollständigkeit

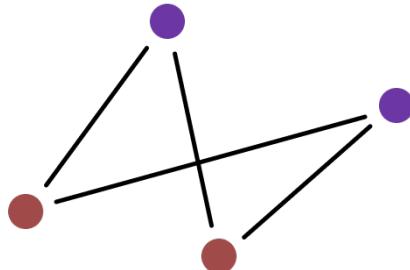
#### 4.2.4.6. Bipartition

Die Besonderheit eines bipartiten Graphen ist die Unterteilung aller Knoten in zwei disjunkte Mengen, wobei innerhalb beider Mengen keine Kanten existieren. Knoten dürfen dementsprechend nur miteinander verbunden sein, falls sich diese nicht in derselben Menge befinden. Allen nicht bipartiten Graphen fehlt es an solch einer Eigenschaft. [33]

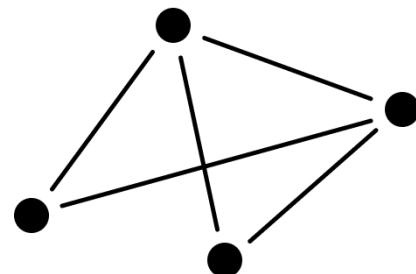
#### 4.2.4.7. Planarität

Ein planarer Graph kann auf einer Ebene ohne Kantenkreuzungen dargestellt werden, was eine klare zweidimensionale Visualisierung ermöglicht. Im Gegensatz dazu erfordert ein nicht planarer Graph Kantenkreuzungen bei der Darstellung auf einer Ebene, was die visuelle Erfassung erschwert. Die Planarität beeinflusst somit nicht nur die Struktur, sondern auch die graphische Repräsentation und Analyse des Graphen [35, 36].

Der Graph in Abbildung 4.13 ist nicht planar gezeichnet, jedoch kann er planar gemacht werden und ist deswegen ein planarer Graph. [33]

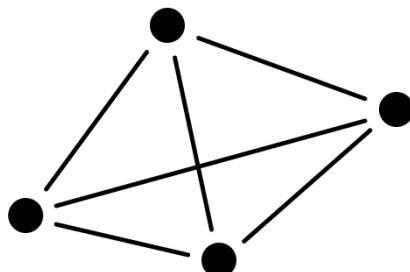


bipartiter Graph

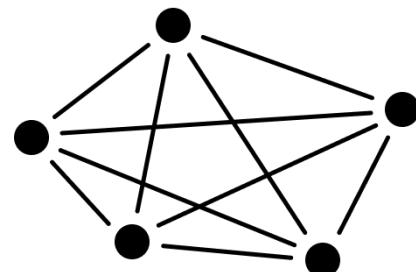


nicht bipartiter Graph

Abbildung 4.12.: Bipartition



planarer Graph



nicht planarer Graph

Abbildung 4.13.: Planarität

#### 4.2.4.8. Eulerscher Graph

Ein eulerscher Graph zeichnet sich durch die Existenz eines geschlossenen Pfades aus, der alle Kanten des Graphen genau einmal durchläuft. Diese charakteristische Eigenschaft ist als Eulerkreis bekannt und verleiht dem Graphen eine herausragende Struktur, da er eine systematische Durchquerung aller Verbindungen ermöglicht, ohne eine Kante zu wiederholen [37, 38].

#### 4.2.4.9. Hamiltonscher Graph

Ein hamiltonscher Graph zeichnet sich durch die inhärente Eigenschaft aus, dass er die Existenz eines geschlossenen Pfades aufweist, welcher alle Knoten des Graphen exakt einmal durchläuft. Dieser geschlossene Pfad, bekannt als Hamiltonkreis, verankert die charakteristische Struktur des Graphen und ermöglicht eine vollständige, einmalige Durchquerung aller Knoten [37, 33].

Das Problem Hamilton beschreibt das Finden eines solchen Pfades (auch Kreis). Dieses Problem hat die Eigenschaft keinen effizienten Algorithmus zu besitzen, jedoch kann eine potenzielle Lösung effizient auf ihre Richtigkeit überprüft werden. Aufgrund dieser Eigenschaft gehört Hamilton zur Klasse NP. Spezifischer ist Hamilton sogar ein NP-vollständiges Problem, eine Klasse mit den schwierigsten Problemen der Mathematik.

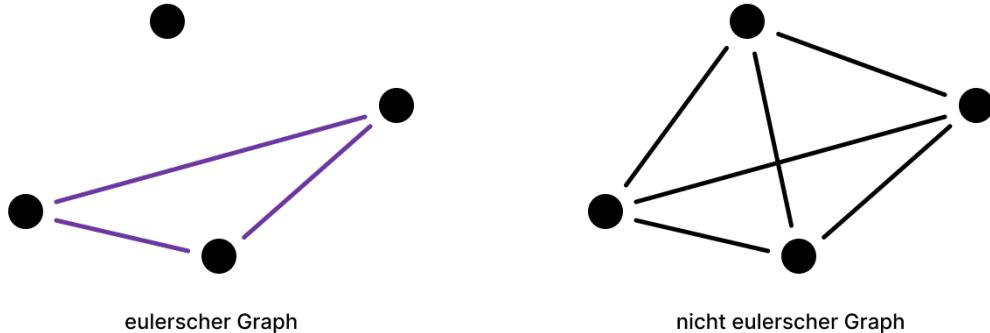


Abbildung 4.14.: Eulerscher Graph

«Das Problem, einen Hamiltonschen Kreis in einem Graphen zu finden, bezeichnet man mit HAMILTON. Auch für dieses Problem ist kein effizienter Algorithmus bekannt.» - Steffen Reith [39]

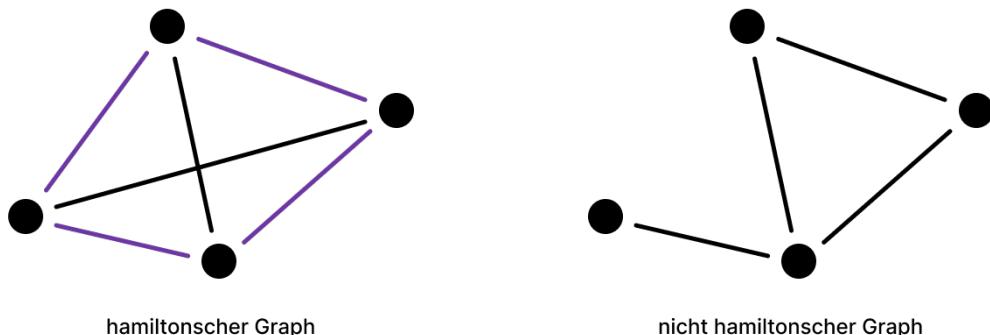


Abbildung 4.15.: Hamiltonscher Graph

#### 4.2.4.10. Ramsey Number

Aufgrund jüngsten Entwicklung im Bereich der Berechnung der Unter- und Obergrenze der diagonalen Ramsey Nummer, möchte ich hier kurz diese fünfjährige Forschung erwähnen. Die Autoren dieser exponenziellen Verbesserungen beschreiben die Ramsey Nummer folgendermaßen:

«The Ramsey number  $R(k)$  is the minimum  $n \in N$  such that every red-blue colouring of the edges of the complete graph on  $n$  vertices contains a monochromatic clique on  $k$  vertices.» - Marcelo Campos [40]

Die Ramsey-Zahl  $R(k)$  ist demnach definiert als die kleinste natürliche Zahl  $n$ , so dass jede Rot-Blau-Färbung der Kanten des vollständigen Graphen mit  $n$  Knoten eine monochromatische Clique mit  $k$  Knoten enthält. Mit anderen Worten, die Ramsey-Zahl gibt an, wie groß der vollständige Graph sein muss, damit jede mögliche Kantenfärbung eine einfarbige (monochromatische) Clique der Größe  $k$  enthält.

#### 4.3. Libraries zur visuellen Darstellung

*Übertragung von Darstellung*

Der Prozess der Transformierung der Programmiersprache in die Darstellung der einzelnen Pixel auf dem Display ist lang. Applikationen basieren auf Frameworks<sup>1</sup>, welche diesen Prozess bereits für Programmierer übernehmen. Es gibt tausende JavaScript bzw. TypeScript Frameworks, die bekanntesten darunter React, Angular, Vue, Solid, Svelte und Node [41].

«Don't call us, we'll call you.» - Pat Wentz [25]

Mittels Libraries können die Funktionalitäten des jeweiligen Frameworks erweitert werden. Diese Funktionalitäten können jegliche Anforderungen betreffen, seien es UI-Kits, Diagramme, 3D-Elemente oder Graphen. In den meisten Fällen werden diese Libraries von der Community entwickelt und stetig verbessert. Die richtige Wahl einer Library zur Erfüllung der Anforderungen ist besonders bei stark vernetzten Daten von großer Bedeutung, da die Performance bei großen Datenmengen schnell schwinden kann.

Eine grundlegende, quelloffene JS Library namens „D3“[42] bietet aufgrund ihrer Flexibilität einige Möglichkeiten zur Darstellung verschiedenster Grafiken. Die Bibliothek basiert auf Webstandards, ist sozusagen „low-level“, um eine Grundstruktur für moderne Visualisierungsbibliotheken zu schaffen. Daher auch die Flexibilität. Darauf aufbauend eignet sich die nach langer Suche gefundene „Cytoscape“-Bibliothek[43], welche vereinfacht gesagt eine Liste mit Knoten und eine Liste mit Kanten annimmt, um den Graphen zu erstellen. Vorteilhaft erscheinen die Vielfalt der Einstellungsmöglichkeiten bezüglich Darstellungsmerkmalen, wie Farben, Dicken, Verhalten, usw., die hohe Performance und Integration von Community Plugins. Besonders im biomolekular Bereich scheint diese Bibliothek auch einige Anwendungsfälle gefunden zu haben [44].

<sup>1</sup>In diesem Dokument werden Begriffe, wie „Framework“ und „Library“, trotz ihres englischen Ursprungs verwendet. Diese Entscheidung erfolgt aufgrund der gängigen Praxis im technischen Bereich, um eine konsistente Terminologie beizubehalten.

## 5. Architektur des Prototypen

### 5.1. Backend

### 5.2. Frontend

## 6. Aufsetzen eines Kafka-Systems

### 6.1. Herausforderungen einer GraphQL API mit relationalen Datenbanken

## 7. Webapplikation

### 7.1. Dashboard mit Kennzahlen des Netzmodells

### 7.2. Tabellen mit Filter- und Suchfunktionen

### 7.3. Graph des Stromnetzmodells

#### 7.3.1. Einfache Integration von Cytoscape in Angular

Die Umsetzung der Visualisierung eines Graphen in Angular ist mittels Cytoscape [43] geschehen. Diese Bibliothek, welche normalerweise für biologische Darstellungen verwendet wird, bietet außerordentlich viele Möglichkeiten, Knoten und Kanten im Graphen zu erstellen, hinzuzufügen, zu löschen und ist mit einigen anderen JavaScript-Bibliotheken kompatibel.

Die Integration in Angular ist dank dem Package Manager npm recht einfach. Zuerst muss das Paket cytoscape installiert werden.

```
1   npm install cytoscape
```

Quellcode 7.1: Cytoscape installieren

Anschließend müssen die in Cytoscape definierten Typen in den @types Ordner gespeichert werden, was mit diesem Befehl erzielt werden kann:

```
1   npm i —save-dev @types/cytoscape
```

Quellcode 7.2: Cytoscape Typen speichern

Somit kann man bei jeder Komponente in Angular die Bibliothek importieren:

```
1   import cytoscape from 'cytoscape';
```

Quellcode 7.3: Cytoscape importieren

Cytoscape bietet nun Möglichkeiten, um einen Graphen zu erstellen und in ein bestimmtes DOM-Element zu rendern. Ein einfacher Graph könnte auf Code-Ebene beispielsweise so aussehen:

```

1   var cy = cytoscape({
2       container: document.getElementById('cy'), // container to render
3       in
4       elements: [
5           // list of graph elements to start with
6           {
7               // node a
8               data: { id: 'a' },
9           },
10          {
11              // node b

```

```

12         data: { id: 'b' },
13     },
14     {
15       // edge ab
16       data: { id: 'ab', source: 'a', target: 'b' },
17     },
18   ],
19
20   style: [
21     // the stylesheet for the graph
22     {
23       selector: 'node',
24       style: {
25         'background-color': '#666',
26         label: 'data(id)',
27       },
28     },
29   ],
30   {
31     selector: 'edge',
32     style: {
33       width: 3,
34       'line-color': '#ccc',
35       'target-arrow-color': '#ccc',
36       'target-arrow-shape': 'triangle',
37       'curve-style': 'bezier',
38     },
39   },
40 ],
41
42   layout: {
43     name: 'grid',
44     rows: 1
45   },
46 });

```

Quellcode 7.4: einfachen Graph initialisieren

Hierbei muss man bei Angular darauf achten, dass der Graph erst initialisiert werden darf, wenn das DOM-Element gerendert wurde. Um sicherzustellen, dass dieser Rendervorgang bereits geschehen ist, gibt es die Funktion `ngAfterViewInit`:

```

1  export class GraphComponent {
2    ngAfterViewInit(): void {
3      var cy = cytoscape({ ... });
4    }
5  }

```

Quellcode 7.5: Funktion `ngAfterViewInit`

Mit diesen Konfigurationen rendert Angular mittels Cytoscape einen Graphen, welcher in etwa so aussieht:

Cytoscape bietet verschiedene Layouts, sprich Anordnungen von Knoten und Kanten im Container, Animationen und stilistische Anpassungen und Adaptionen für den Graphen. Aktuelle API-Definitionen sind unter <https://js.cytoscape.org/> dokumentiert.



Abbildung 7.1.: einfacher Cytoscape Graph

## 8. Bewertung

# I. Literaturverzeichnis

- [1] Curry, Edward: *Message-Oriented Middleware*. In: Mahmoud, Qusay H. (Herausgeber): *Middleware for Communications*, Seiten 1–28. Wiley, 1. Auflage, Juni 2004, ISBN 978-0-470-86206-3 978-0-470-86208-7.
- [2] Papazoglou, Mike P. und Willem Jan Van Den Heuvel: *Service Oriented Architectures: Approaches, Technologies and Research Issues*. The VLDB Journal, 16(3):389–415, Juli 2007, ISSN 1066-8888, 0949-877X.
- [3] Toshev, Martin: *Learning RabbitMQ: Build and Optimize Efficient Messaging Applications with Ease*. Packt Publishing, Birmingham, 2016, ISBN 978-1-78398-456-5.
- [4] Menge, Falko: *Enterprise Service Bus*. In: *Free and Open Source Software Conference*, Band 2, Seiten 1–6, 2007.
- [5] Narkhede, Neha: *Kafka: The Definitive Guide*. O'Reilly Media, 2017.
- [6] Fu, Guo, Yanfeng Zhang und Ge Yu: *A Fair Comparison of Message Queuing Systems*. IEEE Access, 9:421–432, 2021, ISSN 2169-3536.
- [7] Stopford, Ben: *Designing Event-Driven Systems*. 2018.
- [8] Richter, Michael: *Kriterien der Benutzerfreundlichkeit*. Psychologisches Institut der Universität Zürich, November 1997. Onlinequelle: [https://cognit.ch/michaelrichter/literat\\_97.pdf](https://cognit.ch/michaelrichter/literat_97.pdf).
- [9] Krug, Steve: *Don't make me think!: Web & Mobile Usability: Das intuitive Web*. MITP-Verlags GmbH & Co. KG, 2018. Onlinequelle: [https://books.google.at/books?id=e-VIDwAAQBAJ&printsec=frontcover&hl=de&source=gbs\\_ge\\_summary\\_r&cad=0#v=onepage&q&f=false](https://books.google.at/books?id=e-VIDwAAQBAJ&printsec=frontcover&hl=de&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false).
- [10] Mardita, Rizki: *Designing Intuitive User Interface*. Medium, Mai 2017. Onlinequelle: <https://uxplanet.org/create-visual-ui-design-for-better-products-14f91e557638>.
- [11] Laubheimer, Page: *The 3-Click Rule for Navigation Is False*. Nielsen Norman Group, 2019. Onlinequelle: <https://www.nngroup.com/articles/3-click-rule/>.
- [12] Wright, Chris: *Web Optimization: The Myth of the 3 Click Rule*. CMS-Wire, 2010. Onlinequelle: <https://www.cmswire.com/cms/web-engagement/web-optimization-the-myth-of-the-3-click-rule-009018.php>.
- [13] Neuberger, Christoph: *Interaktivität, Interaktion, Internet*. Publizistik, 52(1):33–50, 2007.
- [14] Morville, Peter und Jeffery Callender: *Search patterns: design for discovery*. O'Reilly Media, Inc., 2010. Onlinequelle: [https://books.google.at/books?id=LzgNHuKKGJIC&printsec=frontcover&hl=de&source=gbs\\_ge\\_summary\\_r&cad=0#v=onepage&q&f=false](https://books.google.at/books?id=LzgNHuKKGJIC&printsec=frontcover&hl=de&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false).
- [15] Shneiderman, Ben: *The eyes have it: A task by data type taxonomy for information visualizations*. In: *Proceedings 1996 IEEE symposium on visual languages*, Seiten 336–343. IEEE, 1996.
- [16] Ahlberg, Christopher, Christopher Williamson und Ben Shneiderman: *Dynamic queries for information exploration: An implementation and evaluation*. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, Seiten 619–626, 1992.

- [17] Souders, Steve: *High-performance web sites*. Communications of the ACM, 51(12):36–41, 2008. Onlinequelle: <https://dl.acm.org/doi/fullHtml/10.1145/1409360.1409374>.
- [18] Sharma, Sushil und Pietro Murano: *A usability evaluation of Web user interface scrolling types*. First Monday, 2020. Onlinequelle: <https://firstmonday.org/ojs/index.php/fm/article/view/10309/9400>.
- [19] Hogan, Lara Callender: *Designing for Performance: Weighing Aesthetics and Speed*. O'Reilly Media, Inc., 2014. Onlinequelle: [https://books.google.at/books?id=QPixBQAAQBAJ&printsec=frontcover&hl=de&source=gbs\\_ge\\_summary\\_r&cad=0#v=onepage&q&f=false](https://books.google.at/books?id=QPixBQAAQBAJ&printsec=frontcover&hl=de&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false).
- [20] Jorgensen, Corinne und Elizabeth D Liddy: *Information access or information anxiety?—An exploratory evaluation of book index*. Indexer, 1(20):64–68, 1996. Onlinequelle: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=44fb27a9e274e4975a6bea77af8c0542c3c9600b>.
- [21] Barnum, Carol, Earvin Henderson, Al Hood und Rodney Jordan: *Index versus full-text search: a usability study of user preference and performance*. Technical Communication, 51(2):185–206, 2004.
- [22] Paralogical: *I removed most of the syllables from english and it's 30% faster now*. YouTube, 2023. Onlinequelle: <https://www.youtube.com/watch?v=sRbcw2sGkJw>.
- [23] *Spline*. Onlinequelle: <https://spline.design/>.
- [24] Ware, Colin: *Information Visualization: Perception for Design*. Elsevier Inc., 2021. Onlinequelle: [https://books.google.at/books?id=3-HFDwAAQBAJ&printsec=frontcover&hl=de&source=gbs\\_ge\\_summary\\_r&cad=0#v=onepage&q&f=false](https://books.google.at/books?id=3-HFDwAAQBAJ&printsec=frontcover&hl=de&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false).
- [25] Bavdekar, Sandeep B: *Using tables and graphs for reporting data*. J Assoc Physicians India, 63(10):59–63, 2015. In dieser Arbeit werden die Begriffe 'Graph' und 'Diagramm' synonym verwendet, um auf grafische Darstellungen von Daten oder Beziehungen hinzuweisen, ohne dabei auf etwaige fachliche Unterschiede einzugehen.
- [26] Jacobsen, Jens: *Daten, Diagramme, Dashboards gute UX*. Usabilityblog, 2017.
- [27] Liu, Jia Xin: *Challenges of increasing usability on table filter for huge amounts of data*, 2022.
- [28] *Pagination Examples that Work – We Analyzed the Most Effective Strategies*, 2023.
- [29] Kim, Jaewon, Paul Thomas, Ramesh Sankaranarayana, Tom Gedeon und Hwan Jin Yoon: *Pagination versus scrolling in mobile web search*. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, Seiten 751–760, 2016.
- [30] Doin: *Logarithmic Page Navigation*, 2013.
- [31] Rodgers, Peter, Gem Stapleton und Peter Chapman: *Visualizing sets with linear diagrams*. ACM Transactions on Computer-Human Interaction (TOCHI), 22(6):1–39, 2015.
- [32] Gutwenger, Carsten, Michael Jünger, Karsten Klein, Joachim Kupke, Sebastian Leipert und Petra Mutzel: *A new approach for visualizing UML class diagrams*. In: *Proceedings of the 2003 ACM symposium on Software visualization*, Seiten 179–188, 2003.
- [33] Ohlbach, Hans Jürgen: *Graphen*. Ludwig-Maximilians-Universität München, 2018.

- [34] Klein, Felix: *Zusammenhang von Graphen*. Mathepedia.
- [35] Läuchli, Peter und Peter Läuchli: *Planarität*. Algorithmische Graphentheorie, Seiten 83–98, 1991.
- [36] Schmit, Anne Marie: *Der Satz von Kuratowski*. 2018. Onlinequelle: <https://eplus.uni-salzburg.at/obvusbhs/content/titleinfo/4981548/full.pdf>.
- [37] Brandstädt, Andreas und Andreas Brandstädt: *Eulerkreise und Hamiltonkreise*. Graphen und Algorithmen, Seiten 40–60, 1994.
- [38] Liebling, Thomas M und Thomas M Liebling: *Euler-Graphen,-Zyklen und-Kreise*. Graphentheorie in Planungs-und Tourenproblemen: am Beispiel des städtischen Straßendienstes, Seiten 24–31, 1970.
- [39] Reith, Steffen und Heribert Vollmer: *Ist P = NP? Einführung in die Theorie der NP-Vollständigkeit*. Technischer Bericht, Technical Report 269, Institut für Informatik, Universität Würzburg, 2001 . . . , 2001. Onlinequelle: <https://www.thi.uni-hannover.de/fileadmin/thi/publikationen/re-vo01.pdf>.
- [40] Campos, Marcelo, Simon Griffiths, Robert Morris und Julian Sahasrabudhe: *An exponential improvement for diagonal Ramsey*. arXiv preprint arXiv:2303.09521, 2023. Onlinequelle: <https://arxiv.org/pdf/2303.09521.pdf>.
- [41] Moraru, Andrei Lucian: *The Problem With Too Many JavaScript Frameworks*. Medium, 2021. Onlinequelle: <https://betterprogramming.pub/the-problem-with-too-many-js-frameworks-11531ac8b896>.
- [42] *D3 JavaScript Library*. Onlinequelle: <https://d3js.org/>.
- [43] *Cytoscape JavaScript Library*. Oxford Bioinformatics (2016, 2023). Onlinequelle: <https://js.cytoscape.org/>.
- [44] Shannon, Paul, Andrew Markiel, Owen Ozier, Nitin S Baliga, Jonathan T Wang, Daniel Ramage, Nada Amin, Benno Schwikowski und Trey Ideker: *Cytoscape: a software environment for integrated models of biomolecular interaction networks*. Genome research, 13(11):2498–2504, 2003.
- [45] Freeman, Linton C.: *Visualizing Social Networks*. University of California, Irvine, 2000. Onlinequelle: <https://bebr.ufl.edu/sites/default/files/Freeman%20-%202000%20-%20Visualizing%20social%20networks.pdf>.
- [46] Jünger, Michael, Petra Mutzel, Stephen Kobourov, Sue Whitesides, Andreas Keren, Holger Eichelberger, Martin Harrigan, Patrick Healy und Michael Belling: *Graph Drawing*. In: *Proc. Dagstuhl Seminar [online]*, Band 5191. Citeseer. Onlinequelle: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=cdecf712eb5bd499ed2c2fd52bfc57d0b22e24b4>.
- [47] Mutzel, Michael Jünger und Sebastian Leipert: *Graph Drawing: 9th International Symposium, GD 2001 Vienna, Austria, September 23–26, 2001, Revised Papers*, Band 2265. Springer, 2003. Onlinequelle: [https://books.google.at/books?id=i0NsCQAAQBAJ&printsec=frontcover&hl=de&source=gbs\\_ge\\_summary\\_r&cad=0#v=onepage&q&f=false](https://books.google.at/books?id=i0NsCQAAQBAJ&printsec=frontcover&hl=de&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false).
- [48] Jünger, Michael und Petra Mutzel: *Graph drawing software*. Springer Science & Business Media, 2012.

- [49] Fry, Ben: *Visualizing data*. O'Reilly Media, Inc., 2008. Onlinequelle: [https://books.google.at/books?id=RRswXg4pJhcC&printsec=frontcover&hl=de&source=gbs\\_ge\\_summary\\_r&cad=0#v=onepage&q&f=false](https://books.google.at/books?id=RRswXg4pJhcC&printsec=frontcover&hl=de&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false).
- [50] Kepner, Jeremy, David Bader, Aydin Buluç, John Gilbert, Timothy Mattson und Henning Meyerhenke: *Graphs, matrices, and the GraphBLAS: Seven good reasons*. Procedia Computer Science, 51:2453–2462, 2015.

## II. Abbildungsverzeichnis

3.1.	Ein Beispiel für ein verteiltes System mit direkter Kommunikation [1] . . . . .	13
3.2.	System mit einer Datenbank als zentrale Kommunikationskomponente [3] . . . . .	14
3.3.	Ein Beispiel für ein verteiltes System mit einem zentralen <i>Messaging System</i> [1]	15
3.4.	Ein einfacher <i>Enterprise Service Bus</i> [4] . . . . .	16
3.5.	Eine <i>Queue</i> [1] . . . . .	17
3.6.	Das <i>Point-to-Point Messaging Model</i> [1] . . . . .	18
3.7.	Das <i>Publish-Subscribe Messaging Model</i> [1] . . . . .	19
3.8.	Eine <i>consumer group</i> verarbeitet gemeinsam ein Topic. [5] . . . . .	23
3.9.	Die Architektur des Apache Kafka Systems [6] . . . . .	23
3.10.	Die Architektur des RabbitMQ Systems [6] . . . . .	25
4.1.	einfache Tabelle . . . . .	29
4.2.	nach Nutzernamen aufsteigend sortierte Tabelle . . . . .	30
4.3.	nach Nutzernamen absteigend sortierte Tabelle . . . . .	30
4.4.	Tabelle mit Filterfunktionalität (unverzögert) . . . . .	31
4.5.	Tabelle mit Filterfunktionalität (Tasteneingabe) . . . . .	31
4.6.	aufgeteilte Tabelle aufgrund Platzmangels . . . . .	32
4.7.	Ausrichtung . . . . .	34
4.8.	Konnektivität . . . . .	34
4.9.	Zyklen . . . . .	35
4.10.	Gewicht . . . . .	36
4.11.	Vollständigkeit . . . . .	36
4.12.	Bipartition . . . . .	37
4.13.	Planarität . . . . .	37
4.14.	Eulerscher Graph . . . . .	38
4.15.	Hamiltonscher Graph . . . . .	38
7.1.	einfacher Cytoscape Graph . . . . .	44
B.1.	Jira Plan . . . . .	77

### III. Tabellenverzeichnis

B.1. Kapitelverzeichnis . . . . .	56
B.2. Arbeitstagebuch Schlipfinger . . . . .	78
B.3. Arbeitstagebuch Schneider . . . . .	80

## IV. Quellcodeverzeichnis

7.1. Cytoscape installieren . . . . .	42
7.2. Cytoscape Typen speichern . . . . .	42
7.3. Cytoscape importieren . . . . .	42
7.4. einfachen Graph initialisieren . . . . .	42
7.5. Funktion ngAfterViewInit . . . . .	43

<b>htlkrems</b>	<b>HÖHERE TECHNISCHE BUNDES - LEHRANSTALT Krems</b>
Abteilung:	<b>Informationstechnologie</b>

## V. Abkürzungsverzeichnis

**API** Application Programming Interface

**CRUD** Create - Read/Retrieve - Update - Delete/Destroy

**DI** Dependency Injection

**DNS** Domain Name System

**HCI** Human Computer Interaction

**HTTP** Hypertext Transfer Protocol

**SOA** Service-Oriented Architecture

**UI** User Interface

**UX** User Experience

<b>htlkrems</b>	<b>HÖHERE TECHNISCHE BUNDES - LEHRANSTALT Krems</b>
Abteilung:	<b>Informationstechnologie</b>

## VI. Übersetzungsverzeichnis

**Dashboard** Grafische Übersicht, Schnelle Einblicke; Kontext: Stromnetzmodell

**Framework** Rahmenwerk - Entwicklungsgerüst, vorgefertigte Struktur, abstrahierte Umgebung

**Library** Bibliothek - Sammlung von Funktionen, wiederverwendbare Code-Module

## A. Zusammenfassung und Ausblick

### A.1. Zusammenfassung

Zusammenfassend war diese Diplomarbeit ein sehr lehrreiches Projekt, bei dem wir viele neue Erfahrungen gemacht haben. ...

### A.2. Ausblick

## B. Anhang

### B.1. Kapitelverzeichnis

Kapitel	Autor
1. Präambel	Schneider Felix
2. Einleitung	Schneider Felix
3. Message Propagation	Schlipfinger Clemens
4. Visualisierung	Schneider Felix
5.1 Backend	Schlipfinger Clemens
5.2 Frontend	Schneider Felix
6. Kafka	Schlipfinger Clemens
7.1 API	Schneider Felix
8. Webapplikation	Schneider Felix

Tabelle B.1.: Kapitelverzeichnis

## B.2. Besprechungsprotokolle

### B.2.1. Begleitprotokoll 1



# Begleitprotokoll 1

Meeting Minutes

—

## Project

Id 202324-CF-Networkanalysis

Name Visualisierung der Ergebnisse der Netzdatenmodellanalyse

## Team

Jürgen Katzenschlager Project manager

Clemens Schlipfinger Backend programmer

Felix Schneider Frontend programmer

## Version History

Version	Datum	AutorIn	Änderungen
0.1	28.10.2023	Felix Schneider & Clemens Schlipfinger	Erstellung des Dokuments

## Inhaltsverzeichnis

[Table of Contents]

<b>Inhaltsverzeichnis.....</b>	<b>1</b>
<b>Metadaten.....</b>	<b>2</b>
<b>Ergebnisse.....</b>	<b>2</b>
<b>Inhalte der Besprechung.....</b>	<b>2</b>
Themen.....	2
Next Steps.....	2

<b>htlkrems</b>	<b>HÖHERE TECHNISCHE BUNDES - LEHRANSTALT Krems</b>
Abteilung:	Informationstechnologie

2



## Metadaten

- Datum: 28.10.2023
- Zeit: 18:00 - 19:02 Uhr
- Ort: Home Office
- Anwesenden Personen
  - Jürgen Katzenschlager
  - Clemens Schlipfinger
  - Felix Schneider

## Ergebnisse

- Pflichtenheft erstellt und Siemens geschickt
- Mockup halb fertig

## Inhalte der Besprechung

### Themen

- Pflichtenheft
  - Datum bei Suche?
  - Deployment/Abgabe als Docker-Container?
  - Mockup:
    - Dashboard mehr Content
    - Accounts müssen weg!
    - Filter: Sortierung
  - Zeichnung der Systemarchitektur mit Farben und Icons erweitern.

### Next Steps

- Jira Plans
  - Arbeitspakete definieren
- Mockup und Zeichnung der Systemarchitektur verfeinern
- Art der Abgabe herausfinden (Siemens fragen)

## B.2.2. Begleitprotokoll 2



# Begleitprotokoll 2

Meeting Minutes

---

## Project

Id 202324-CF-Networkanalysis

Name Visualisierung der Ergebnisse der Netzdatenmodellanalyse

## Team

Jürgen Katzenschlager Project manager

Clemens Schlipfinger Backend programmer

Felix Schneider Frontend programmer

## Version History

Version	Datum	AutorIn	Änderungen
0.1	13.12.2023	Felix Schneider & Clemens Schlipfinger	Erstellung des Dokuments

## Inhaltsverzeichnis

[Table of Contents]

<b>Inhaltsverzeichnis.....</b>	<b>1</b>
<b>Metadaten.....</b>	<b>2</b>
<b>Ergebnisse.....</b>	<b>2</b>
<b>Inhalte der Besprechung.....</b>	<b>2</b>
Themen.....	2
Next Steps.....	2

<b>htlkrems</b>	<b>HÖHERE TECHNISCHE BUNDES - LEHRANSTALT Krems</b>
Abteilung:	Informationstechnologie

2



## Metadaten

- Datum: 13.12.2023
- Zeit: 11:13 - 11:46 Uhr
- Ort: 5AHIT
- Anwesenden Personen
  - Jürgen Katzenschlager
  - Clemens Schlipfinger
  - Felix Schneider

## Ergebnisse

- Datenbankmodell fertig

## Inhalte der Besprechung

### Themen

- Datenbankmodell
  - Equipments (Connected vs Connecting)
  - Single Table machen?
  - Node\_has\_equipment entfernen
  - Findings Types
- Strimzi: Kafka Topics Config

### Next Steps

- Jira Plan exportieren als Bild
- Jira Issues aufräumen und Spring planen
  - Research und Theoretische Grundlagen
  - Definition der API
  - Kafka Konfigurieren (Topics, ...)
  - Database Schema fertig stellen
- Postman Collections anschauen, damit Frontend nicht abhängig

### B.2.3. Begleitprotokoll 3



# Begleitprotokoll 3

Meeting Minutes

---

## Project

Id 202324-CF-Networkanalysis

Name Visualisierung der Ergebnisse der Netzdatenmodellanalyse

## Team

Jürgen Katzenschlager Project manager

Clemens Schlipfinger Backend programmer

Felix Schneider Frontend programmer

## Version History

Version	Datum	AutorIn	Änderungen
0.1	07.01.2023	Felix Schneider & Clemens Schlipfinger	Erstellung des Dokuments

## Inhaltsverzeichnis

[Table of Contents]

<b>Inhaltsverzeichnis.....</b>	<b>1</b>
<b>Metadaten.....</b>	<b>2</b>
<b>Ergebnisse.....</b>	<b>2</b>
<b>Inhalte der Besprechung.....</b>	<b>2</b>
Themen.....	2
Next Steps.....	2



## Metadaten

- Datum: 07.01.2023
- Zeit: 18:03 - 18:27 Uhr
- Ort: Home Office
- Anwesenden Personen
  - Jürgen Katzenschlager
  - Clemens Schlipfinger
  - Felix Schneider

## Ergebnisse

- Datenbankmodell und API definiert
- Kafka mit Kubernetes rennt, Topics aufgesetzt
- Angular Dark/Light Mode integrieren
- Spring Framework informieren, PostgreSQL
- Theoretischer Teil mit Bildern von Felix fertig, Clemens angefangen

## Inhalte der Besprechung

### Themen

- Theoretischen Teil
- Jira Plan anschauen
  - Nicht erledigte Issues (mitten im Sprint hinzugefügt) zurück in Backlog
- Empirischen Teil
  - Zuerst Architektur beschreiben, dort kommen dann neue Themen, zB GraphQL, welche kurz beschrieben werden können
  - Interessante / schwierige Themen, welche während der Programmierung aufgetreten sind beschreiben (Notizen machen)

### Next Steps

- Jira Sprint planen (Januar)
  - Prototypen (Felix und Clemens)
  - Theoretischen Teil fertig schreiben (Clemens)
- Abbildungen besser zum Thema / Überschrift beschreiben

Meeting Minutes / Begleitprotokoll 3

<b>htlkrems</b>	<b>HÖHERE TECHNISCHE BUNDES - LEHRANSTALT Krems</b>
Abteilung:	<b>Informationstechnologie</b>

width=!,height=!,pages=1,scale=.85,pagecommand=

#### B.2.4. Begleitprotokoll 4

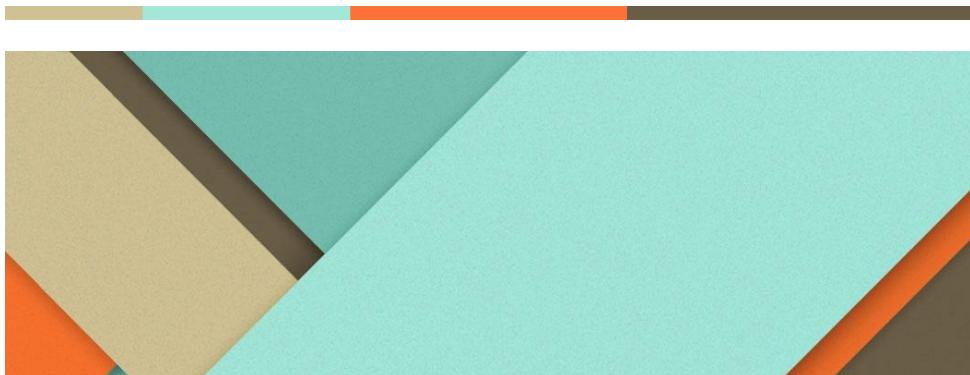
width=!,height=!,pages=2-,scale=.85,pagecommand=

width=!,height=!,pages=1,scale=.85,pagecommand=

#### B.2.5. Begleitprotokoll 5

width=!,height=!,pages=2-,scale=.85,pagecommand=

## B.3. Pflichtenheft



# Pflichtenheft

Requirements Specification

—

## Project

Id 202324-CF-Networkanalysis

Name Visualisierung der Ergebnisse der Netzdatenmodellanalyse

## Team

Jürgen Katzenschlager Project manager

Clemens Schlipfinger Backend programmer

Felix Schneider Frontend programmer

**Version History**

<b>Version</b>	<b>Datum</b>	<b>AutorIn</b>	<b>Änderungen</b>
0.1	17.08.2023	Felix Schneider	Erstellung des Dokuments
0.2	22.08.2023	Felix Schneider	Hinzufügen einiger Ziele
0.3	23.08.2023	Felix Schneider	Ziele schreiben [überarbeitungswürdig]
0.4	10.10.2023	Felix Schneider	Funktionale Anforderungen Grafik
0.5	11.10.2023	Clemens Schlipfinger	Systemarchitektur Grafik hinzufügen
0.5	25.10.2023	Felix Schneider	Mockup hinzufügen
0.6	25.10.2023	Felix Schneider	Nicht Funktionale Anforderungen anfangen
0.7	26.10.2023	Felix Schneider	Überarbeitung Ziele Frontend, technische Anforderungen
0.8	27.10.2023	Clemens Schlipfinger	Hinzufügen Backend Ziele und Anforderungen
0.9	30.10.2023	Felix & Clemens	Überarbeitung mit Siemens

## Inhaltsverzeichnis

[Table of Contents]

<b>Inhaltsverzeichnis.....</b>	<b>2</b>
<b>Ziele.....</b>	<b>3</b>
MUSS-Ziele.....	3
KANN-Ziele.....	4
NICHT-Ziele.....	4
<b>Funktionale Anforderungen.....</b>	<b>5</b>
Anforderungen an das Backend.....	5
Optionale Ziele.....	5
Anforderungen des Frontends.....	6
<b>Nicht funktionale Anforderungen.....</b>	<b>7</b>
<b>Technische Anforderungen.....</b>	<b>8</b>
Übersicht.....	8
Backend.....	8
Frontend.....	8
<b>Mockup.....</b>	<b>9</b>



## Ziele

[Goals]

### MUSS-Ziele

Die Muss-Ziele definieren klar, welche Anforderungen unbedingt erfüllt sein müssen. Alle Ziele sind bis **12.04.2024** zu erfüllen.

- Backend
  - Der Backend-Server soll die Daten (Findings) persistieren und für die Webanwendung in einer sinnvollen Art zur Verfügung stellen.
  - Der Backend-Server soll in der Lage sein, gleichzeitig Daten von verschiedenen Netzmodellanalysen zu verarbeiten.
  - Der Backend-Server soll sich leicht in das bestehende System integrieren und als optionale Erweiterung des Systems gelten.
  - Der Backend-Server soll asynchron funktionieren, welches eine gleichzeitige Verarbeitung und Bereitstellung der Daten ermöglicht.
- Frontend
  - Die Webanwendung bietet umfangreiche Suchmöglichkeiten in tabellarischer Form, nach den Kriterien Schweregrad, Kategorie, Spannungsebene, ID des Findings und der technischen Adresse des Findings..
  - Die Webanwendung visualisiert Relationen im Stromnetzwerk mittels Graphen.
  - Die Webanwendung stellt Informationen übersichtlich in einem Dashboard dar.



## KANN-Ziele

Die Kann-Ziele sind optional.

- Backend
  - Das Backend soll auch den Client über neue Daten informieren.
  - Die API verfügt über eine Authentifizierung mittels Siemens Integration.
- Frontend
  - Die Webanwendung stellt Zusammenhänge in Diagrammen dar.

## NICHT-Ziele

Die Nicht-Ziele sind explizit nicht zu erfüllen. Um Verneinungen zu vermeiden, sind diese trotzdem so formuliert, als würden sie erfüllt werden müssen.

- Die Fehler des Netzmodells sollen erkannt und in die Findings (Java Objekt) gespeichert werden.
- Es wird im bestehenden Repository von Siemens gearbeitet.
- Die Applikation wird auch für mobile Geräte entwickelt.



## Funktionale Anforderungen

[Functional Requirements]

### Anforderungen an das Backend

Das Backend soll die Daten der Netzmodellfehleranalyse verarbeiten. Die Daten sind die Findings, welche die Ergebnisse der Netzmodellfehleranalyse sind.

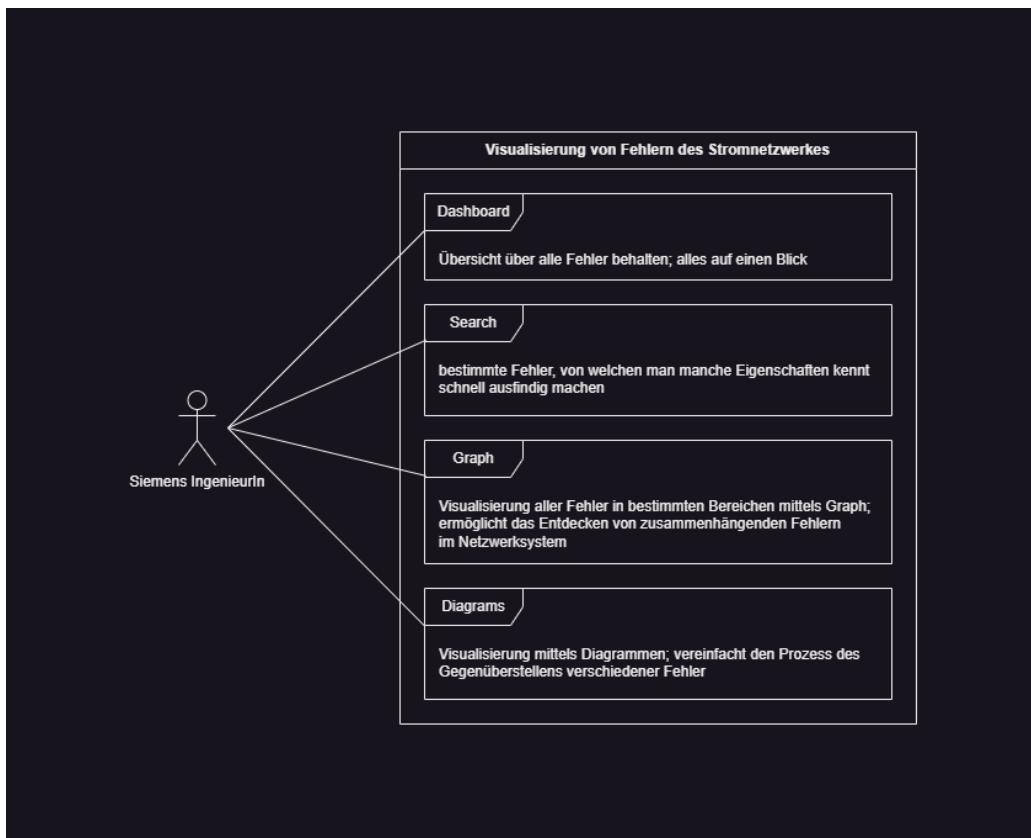
- Es müssen die Daten mit Hilfe einer Datenbank **persistiert** werden und bis zur nächsten Netzmodellfehleranalyse gehalten werden.
- Die Findings müssen durch eine **Schnittstelle (GraphQL) dem Frontend** angeboten werden.
- Der Backend soll sich als **optionale Erweiterung** leicht in das bisherige System integrieren lassen. Zusätzlich soll es nur kleine Veränderungen am Haupt-System erzwingen.
  - Für den Entwicklungsprozess soll eine **Simulationsssoftware** verwendet werden.
  - Es kann davon ausgegangen werden, dass ein Kafka-System und eine Datenbank schon bestehen. Das Programm, welches die Daten in das Kafka Topic schreibt, wird von Siemens bereitgestellt.
- Das Backend soll **verschiedene Netzmodellfehleranalyse** verarbeiten können, das heißt, es muss Findings von mehreren Exportmodulen gleichzeitig entgegennehmen können.

### Optionale Ziele

- Es kann eine Authorization für das Einschränken des Zugriffs auf die Schnittstelle zum Frontend geben.
- Der Backend-Server soll über die Funktionalität verfügen, den Client mit **Server-Side Push Updates** über den Eingang von neuen Daten zu informieren.

## Anforderungen des Frontends

Dieses Use Case Diagramm veranschaulicht die Funktionalitäten des Frontends:



## Nicht funktionale Anforderungen

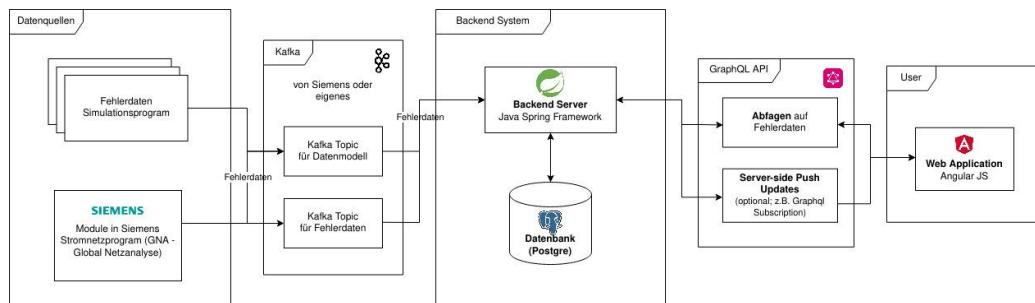
[Non-Functional Requirements]

Aspekt \ Projektergebnis	Webanwendung	Kafka-System
<b>Usability</b>	Intuitive Benutzbarkeit	-
<b>Design</b>	Nicht responsive für mobile Geräte	-
<b>Navigation</b>	Einfache Menüs, max. drei Klicks notwendig, um irgendwohin zu kommen	-
<b>Look &amp; Feel</b>	Anlehnung an Siemens Applikationen; Intuitiv	Integration Siemens Kafka System
<b>Barriere-freiheit</b>	Kein TTS, Zoom oder Augensteuerung	-
<b>Mehrsprachigkeit</b>	Prinzipiell alles English, Icons ermöglichen eingeschränkte multilinguale Nutzung; Möglichkeit auf Erweiterung der Sprachen mittels Konfigurationsfile	-
<b>Customisation</b>	Dark / Light Mode	-
<b>Zeitverhalten</b>	-	Schnelle Änderungen

## Technische Anforderungen

In diesem Kapitel wird der Technologie-Stack detaillierter beschrieben.

### Übersicht



### Backend

Die **Daten**, welche über das Kafka System übertragen werden können in **zwei Arten** unterteilt werden:

- Findings - Informationen über die gefundenen Fehler
- Datenmodell Objekte - Informationen über den Aufbau des Netzes

Diese zwei Arten von Daten werden über **zwei verschiedene Kafka Topics** erhalten.

Der Backend-Server, welcher mit dem **Java Spring Framework** implementiert ist, soll die Daten über die Event Streaming Plattform **Kafka** erhalten. Weiters muss es erhaltene Daten in einer **Datenbank** speichern und als **GraphQL API** den Frontend zur Verfügung stellen.

### Frontend

Die Webanwendung verwendet das AngularJS Framework. Genau wie React, Vue und viele andere JS Frameworks basiert dieses auf Komponenten.

Über eine GraphQL API können Abfragen an das Backend gestellt werden.

## Mockup

Das Mockup für die Web-Applikation wurde mit Figma erstellt. Hier ist ein Link zu der Datei: <https://www.figma.com/file/PpiE28AJCmFWBdhD66wYX5/Siemens?type=design&mode=design&t=pvAVxSDRdFICOXGr-1>

Folgende Bilder veranschaulichen grob das Design:



	<b>HÖHERE TECHNISCHE BUNDES - LEHRANSTALT</b> <b>Krems</b>	
Abteilung:	Informationstechnologie	

## B.4. Arbeitspakte

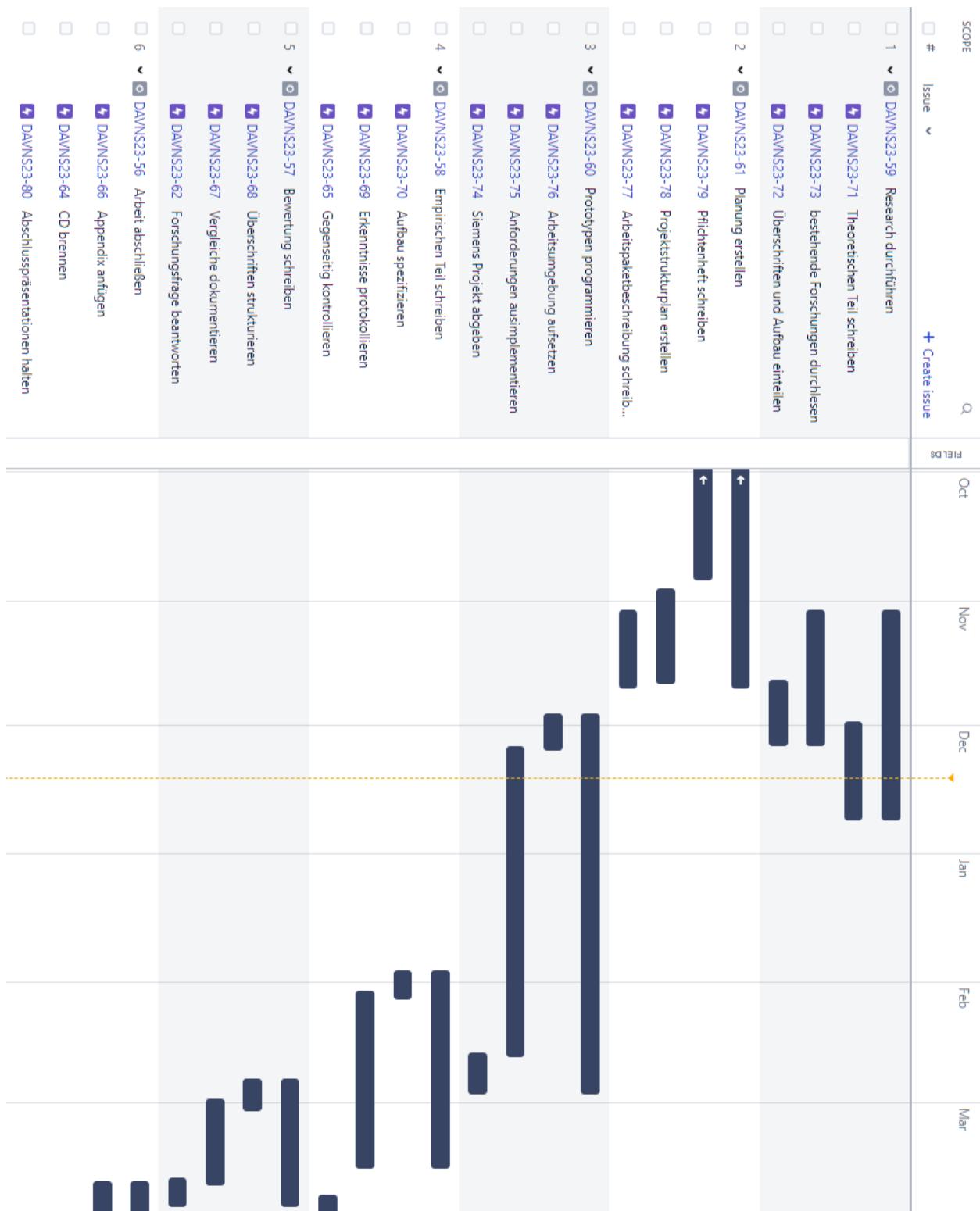


Abbildung B.1.: Jira Plan

## B.5. Projekttagebücher

### B.5.1. Projekttagebuch Clemens Schlipfinger

Tag	Zeit	kumulativ	Fortschritt
-----	------	-----------	-------------

Tabelle B.2.: Arbeitstagebuch Schlipfinger

## B.5.2. Projekttagebuch Felix Schneider

Tag	Zeit	kumulativ	Fortschritt
14 September	00:08:33	00:08:33	Teams Team organisieren
18 September	00:12:06	00:20:39	Diplomarbeit Datenbank Felder ausfüllen
18 September	00:25:01	00:45:40	Clemens EWA erklären
18 September	00:07:24	00:53:04	Diplomarbeit Datenbank Felder ausfüllen
18 September	00:06:03	00:59:14	Diplomarbeit Datenbank Felder ausfüllen
19 September	00:36:28	01:35:52	Diplomarbeit Datenbank Felder ausfüllen
20 September	02:35:23	04:11:15	Mockup Figma View erstellen
21 September	00:58:00	05:09:15	Mockup Figma View erstellen
21 September	00:32:00	05:41:15	Mockup Figma View erstellen
21 September	01:43:00	07:24:15	Mockup Figma View erstellen
23 September	02:30:00	09:54:15	Diplomarbeit Datenbank Felder ausfüllen
10 Oktober	00:18:00	10:12:15	UseCase Frontend für Pflichtenheft
13 Oktober	03:00:00	13:12:15	Research Graph Visualisation
19 Oktober	01:52:00	15:04:15	Mockup
19 Oktober	01:21:00	16:25:15	Mockup
19 Oktober	01:03:47	17:29:02	Mockup
23 Oktober	00:17:46	17:46:48	Mockup
23 Oktober	00:20:25	18:07:13	Mockup
24 Oktober	00:37:49	18:45:02	Mockup
24 Oktober	00:02:04	18:47:06	Mockup
25 Oktober	01:27:02	20:14:08	Mockup
25 Oktober	00:53:00	21:07:08	Mockup
25 Oktober	00:27:04	21:34:12	Pflichtenheft
25 Oktober	00:10:23	21:44:35	Mockup
25 Oktober	00:01:12	21:45:47	Pflichtenheft
26 Oktober	00:35:09	22:20:56	Pflichtenheft
26 Oktober	00:05:34	22:26:33	Mockup
27 Oktober	00:51:59	23:18:32	Pflichtenheft
28 Oktober	01:20:00	24:38:32	Besprechung
29 Oktober	00:30:00	25:08:32	Mockup
29 Oktober	00:23:14	25:31:46	Mockup
30 Oktober	06:00:00	31:31:46	Meeting bei Siemens; Überarbeitung Pflichtenheft; Kleinigkeiten klären
01 November	00:30:00	32:01:46	Jira Plan erstellen
01 November	00:30:00	32:31:46	Jira Plan erstellen
01 November	00:30:00	33:01:46	Confluence weiterarbeiten
03 November	00:45:00	33:46:46	Jira Plan und Confluence
03 November	00:21:42	34:08:28	Mockup
03 November	00:31:16	34:39:44	Mockup
03 November	02:06:28	36:46:12	Learn Angular
16 November	00:50:59	37:37:11	Jira Plan erstellen
16 November	00:21:11	37:58:22	Jira Plan erstellen
10 Dezember	00:20:38	38:19:00	Angular Table
10 Dezember	00:51:16	39:10:16	Grundstruktur aufbauen
13 Dezember	00:45:00	39:55:16	Meeting Minutes 2
13 Dezember	03:00:00	42:55:16	Datenbankstruktur Modellierung und Research Visualisierungsmethoden
14 Dezember	00:28:10	43:23:26	Research zitieren Benutzerfreundlichkeit Seite 80 von 81
16 Dezember	01:11:49	44:35:15	Research zitieren Benutzerfreundlichkeit

<b>htlkrems</b>	<b>HÖHERE TECHNISCHE BUNDES - LEHRANSTALT Krems</b>
Abteilung:	<b>Informationstechnologie</b>

## B.6. Datenträgerbeschreibung

Auf der beigelegten DVD befinden sich folgende Dateien:

- Die schriftliche Arbeit befindet sich unter **thesis/documentation/final.pdf**
- Die schriftliche Arbeit in L<sup>A</sup>T<sub>E</sub>X-Format befindet sich unter **thesis/latex/**
- Sämtliche Projektmanagement Dateien befinden sich unter **thesis/management/**

Das ausschließliche Recht zur Nutzung, Verwaltung und Verfügung über den erstellten Programmcode liegt in Übereinstimmung mit den einschlägigen Vertragsvereinbarungen und geistigen Eigentumsrechten bei Siemens. Aus diesem Grund dürfen rechtlich gesehen keine Programmdateien oder Quellcode auf die DVD gebrannt werden.