



Abschlussprüfung Sommer 2021

Fachinformatiker für Anwendungsentwicklung  
Dokumentation zur betrieblichen Projektarbeit

# Entwicklung von ‘Routemap‘

Webbasiertes Tool zur Visualisierung von Routingregeln

Abgabetermin: Rostock, den 26.05.2021

**Prüfungsbewerber:**

Henry Knopsmeier  
Herderstraße 37  
18311 Ribnitz-Damgarten



**Ausbildungsbetrieb:**

Boreus GmbH  
Zur Schwedenschanze 2  
18435 Stralsund



Industrie- und Handelskammer  
zu Rostock

**Antrag für die betriebliche Projektarbeit**  
**- Abschlussprüfung Sommer/Winter 20 21**

Berufsbezeichnung/ Fachrichtung/ Einsatzgebiet:

Fachinformatiker FR Anwendungsentwicklung

Antragsteller:

Henry Knopsmeier  
Nikolaus-Dierling-Str.20  
18317 Saal OT Bartelshagen II  
Tel.: +49 152 09239638  
Mail: henry@knopsmeier.de

Ausbildungsbetrieb:

Boreus GmbH  
Zur Schwedenschanze 2  
18435 Stralsund  
Tel.: +49 3831 3676400  
Mail: info@boreus.de

Thema der Projektarbeit:

Entwicklung eines Programms zum Importieren von Routingregeln in eine Datenbank sowie die graphische Darstellung dieser Daten innerhalb eines gerichteten Graphen.

Projektbeschreibung:

Zur Minimalisierung des Analyseaufwandes bei einer Störung der Erreichbarkeit interner Systeme ist ein Programm zu entwickeln, welches auf einem zentralen Verwaltungssystem vorhandene Konfigurationsdateien parst. Anschließend werden die erfassten Daten in ein, für das vorhandene Datenbankmodell sowie die assoziierte Normalform, adäquates Format gefiltert. Weiterhin soll das Programm diese Datensätze in einer Datenbank neu anlegen oder bestehende Datensätze aktualisieren. Schlussendlich sollen die Daten in einem gerichteten Graphen visualisiert und über einen Webserver erreichbar gemacht werden.

Bei nicht ausreichendem Platz zusätzliches Blatt einfügen.

Projektphasen mit Zeitplanung in Stunden:

- Analyse: 5
- Entwurf: 10
- Implementierung: 30
- Abnahme und Deployment: 5
- Dokumentation: 20
- Gesamt: 70

Geplante Dokumentation:

- Konzept / Ablaufplan
- Dokumentation des Programms
- Quellcodeausschnitte
- Screenshots

Geplante Präsentationsmittel (bitte zutreffendes ankreuzen):

Laptop  
☒

Beamer  
☒

Tageslichtprojektor  
☐

andere (sind vom Prüfungsteilnehmer funktionsfähig mitzubringen)  
☐

Durchführungszeitraum:

von: 24.03.2021

bis: 16.04.2021

Projektverantwortlicher im Ausbildungs-/Praktikumsbetrieb:

Name:

Vorname:

Telefon Nr.:

Gäbler

René

03831 3676436

Suhl, 18.03.2021

Ort, Datum

*U. Griesner*

Unterschrift des Prüfungsteilnehmers

Einverständniserklärung des Ausbildungs-/Praktikumbetriebes zur Durchführung des Projektes:

Bad Gandorf, 23.02.2021

Ort, Datum

**boreus**  
*[Signature]*  
Seminar GmbH  
In Schwedenschanze 2, 18435 Stralsund  
Tel: +49 3831 387 6400  
Fax: +49 3831 387 6621

Der Antrag für die Projektarbeit wurde

angenommen



abgelehnt



Auflagen:

10.3.21

Ort, Datum:

*[Signature]*

Unterschrift des Prüfungsausschusses der  
IHK zu Rostock

Der Abgabetermin der betrieblichen Projektarbeit wird durch die IHK zu Rostock festgelegt:

26.5.21

26. MAI 2021

<b>IHK</b> Abschlussprüfung – Sommer 2021	Vor- und Familienname: Henry Knopsmeier
	Prüfungsnummer: (184) 00052
<b>IT- Berufe</b> <b>Betriebliche Projektarbeit (Teil A)</b> <b>Persönliche Erklärung</b>	Ausbildungsberuf:  Fachinformatiker/-in Anwendungsentwicklung

Ich versichere durch meine Unterschrift, dass ich das betriebliche Projekt in der vorgegebenen Zeit durchgeführt und die dazugehörige Dokumentation selbstständig ohne fremde Hilfe erstellt habe. Alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, wurden von mir als solche kenntlich gemacht.

Die eingereichte betriebliche Projektarbeit hat in dieser Form weder bei der prüfenden noch oder einer anderen IHK vorgelegen.

\_\_\_\_\_  
 Ort: Datum: Unterschrift Antragsteller/-in (Prüfling)

Ich habe die obige persönliche Erklärung zur Kenntnis genommen und bestätige, dass die betriebliche Projektarbeit und die dazugehörige Dokumentation in der vorgegebenen Zeit in unserem Betrieb durch den Prüfling selbstständig bearbeitet und erstellt wurden. Die einzureichende Dokumentation liegt im Unternehmen vor.

\_\_\_\_\_  
 Ort: Datum: Stempel/Unterschrift Verantwortliche/r für die Projektarbeit

## Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>IV</b>
<b>Tabellenverzeichnis</b>	<b>V</b>
<b>Listings</b>	<b>VI</b>
<b>Abkürzungsverzeichnis</b>	<b>VII</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Projektumfeld . . . . .	1
1.2 Projektbeschreibung . . . . .	1
1.3 Projektziel . . . . .	2
1.4 Projektbegründung . . . . .	2
1.5 Projektschnittstellen . . . . .	3
1.6 Projektabgrenzung . . . . .	3
<b>2 Projektplanung</b>	<b>3</b>
2.1 Projektphasen . . . . .	3
2.2 Abweichungen vom Projektantrag . . . . .	4
2.3 Ressourcenplanung . . . . .	4
2.4 Entwicklungsprozess . . . . .	4
<b>3 Analysephase</b>	<b>5</b>
3.1 Ist-Analyse . . . . .	5
3.2 Wirtschaftlichkeitsanalyse . . . . .	5
3.2.1 „Make or Buy“-Entscheidung . . . . .	5
3.2.2 Projektkosten . . . . .	6
3.2.3 Amortisationsdauer . . . . .	6
3.3 Nutzwertanalyse . . . . .	8
3.4 Anwendungsfälle . . . . .	8
3.5 Qualitätsanforderungen . . . . .	8
3.6 Lastenheft/Fachkonzept . . . . .	9
<b>4 Entwurfsphase</b>	<b>9</b>
4.1 Zielplattform . . . . .	9
4.2 Architekturdesign . . . . .	9
4.3 Entwurf der Benutzeroberfläche . . . . .	10
4.4 Datenmodell . . . . .	10
4.5 Planung der Geschäftslogik . . . . .	10
4.6 Maßnahmen zur Qualitätssicherung . . . . .	11

4.7	Pflichtenheft/Datenverarbeitungskonzept . . . . .	11
<b>5</b>	<b>Implementierungsphase</b>	<b>12</b>
5.1	Iterationsplanung . . . . .	12
5.2	Implementierung der Datenstrukturen . . . . .	12
5.3	Implementierung der Geschäftslogik . . . . .	12
5.4	Implementierung der Benutzeroberfläche . . . . .	14
<b>6</b>	<b>Abnahmephase</b>	<b>15</b>
6.1	Abnahme durch den Fachbereich . . . . .	15
6.2	Deployment . . . . .	15
<b>7</b>	<b>Dokumentation</b>	<b>16</b>
7.1	Projektdokumentation . . . . .	16
7.2	Anwenderdokumentation . . . . .	16
7.3	Entwicklerdokumentation . . . . .	16
7.3.1	API-Dokumentation . . . . .	16
7.3.2	Code-Dokumentation . . . . .	16
<b>8</b>	<b>Fazit</b>	<b>17</b>
8.1	Soll-/Ist-Vergleich . . . . .	17
8.2	Lessons Learned . . . . .	17
8.3	Ausblick . . . . .	17
<b>9</b>	<b>Literaturverzeichnis</b>	<b>18</b>
<b>A</b>	<b>Anhang</b>	<b>i</b>
A.1	Detaillierte Zeitplanung . . . . .	i
A.2	Verwendete Ressourcen . . . . .	ii
A.3	Amortisationsdauer . . . . .	iv
A.4	Lastenheft (Auszug) . . . . .	v
A.5	Use Case-Diagramm . . . . .	vi
A.6	Pflichtenheft (Auszug) . . . . .	vii
A.7	Datenmodell Diagramme . . . . .	x
A.8	Ereignisgesteuerte Prozesskette . . . . .	xii
A.9	Sequenzdiagramm . . . . .	xiii
A.10	Oberflächenentwurf . . . . .	xiv
A.11	Screenshot der Verzeichnisstruktur . . . . .	xv
A.12	Screenshot der Apache2-Konfiguration (Ausschnitt) . . . . .	xvi
A.13	Screenshot der Cron-Konfiguration (Ausschnitt) . . . . .	xvii
A.14	Screenshot der Continuous Integration (Mittel zum automatisierten Testen, Kompilieren etc.) (CI)-Pipeline . . . . .	xviii

A.15	Screenshot der Anwenderdokumentation (Ausschnitt) . . . . .	xix
A.16	Screenshot der Application Programming Interface (API)-Dokumentation (Ausschnitt) . . . . .	xx
A.17	Screenshots der Anwendung . . . . .	xxi
A.18	Iterationsplan . . . . .	xxii
A.19	Listings . . . . .	xxiii
A.19.1	Listing des Datenbank-Moduls . . . . .	xxiii
A.19.2	Listing der CTE . . . . .	xxiv
A.19.3	Listing der SQL-Funktion . . . . .	xxv
A.20	Soll-/Ist-Vergleich . . . . .	xxv

## Abbildungsverzeichnis

1	Use Case-Diagramm . . . . .	vi
2	Vereinfachtes ER-Modell . . . . .	x
3	Tabellenschema der Datenbank . . . . .	xi
4	Ereignisgesteuerte Prozesskette (EPK) des Prozesses der Störungsbear- beitung . . . . .	xii
5	Sequenzmodell der Anwendung der verschiedenen Komponenten . . . . .	xiii
6	Mock-Up der Benutzeroberfläche . . . . .	xiv
7	Jegliche Verzeichnisse und Dateien des Projektes . . . . .	xv
8	Virtualhost-Konfiguration des Apache2-Webservers . . . . .	xvi
9	Cronjob-Konfiguration . . . . .	xvii
10	Erfolgreich durchgeführte Pipeline . . . . .	xviii
11	Installationsanleitung der Anwenderdokumentation . . . . .	xix
12	Ausschnitt der API-Dokumentation . . . . .	xx
13	Anzeige des Graphen sowie der das Netzwerk routenden Regeln . . . . .	xxi



## Tabellenverzeichnis

1	Zeitplanung . . . . .	4
2	Kostenaufstellung . . . . .	6
3	Zeiteinsparung . . . . .	7

## Listings

1	Auszug des Datenbank-Modul-Quelltextes . . . . .	xxiii
2	Sequential Query Language (SQL)-Befehl zur Definition der Common Table Expression (temporär benannte Resultatssätze zur Wiederverwen- dung in weiteren Abfragen) (CTE) . . . . .	xxiv
3	SQL-Befehl zur Definition der benötigten Funktion . . . . .	xxv

## Abkürzungsverzeichnis

<b>JSON</b>	Javascript Object Notation
<b>DNS</b>	Domain Name System
<b>Boreus</b>	Boreus GmbH
<b>IP</b>	Internet Protocol
<b>API</b>	Application Programming Interface
<b>URL</b>	Uniform Resource Locator
<b>SLA</b>	Service Level Agreement (Vertrag mit dem Kunden, welcher z. B. Reaktionszeiten festlegt)
<b>ERM</b>	Entity-Relationship-Modell
<b>HTML</b>	HyperText Markup Language
<b>CI</b>	Continuous Integration (Mittel zum automatisierten Testen, Kompilieren etc.)
<b>FCGI</b>	Fast Common Gateway Interface
<b>EPK</b>	Ereignisgesteuerte Prozesskette
<b>CTE</b>	Common Table Expression (temporär benannte Resultatssätze zur Wiederverwendung in weiteren Abfragen)
<b>SSL</b>	Secure Socket Layer
<b>CA</b>	Common Authority
<b>SQL</b>	Sequential Query Language

## 1 Einleitung

Die folgende Projektdokumentation beschreibt den Entwicklungsprozess des IHK-Abschlussprojekts, welches der Autor im Rahmen seiner Ausbildung zum Fachinformatiker, Fachrichtung Anwendungsentwicklung, durchgeführt hat. Der zuständige Ausbildungsbetrieb ist, als Mitglied der Release42-Group die Boreus GmbH ([Boreus](#)), ein Rechenzentrum sowie Cloud-Anbieter mit Standort in Stralsund.

### 1.1 Projektumfeld

Der Auftraggeber des Projekts ist die für die Infrastruktur des Betriebs verantwortliche Fachabteilung von [Boreus](#). Zu den Hauptaufgaben der Infrastruktur bei [Boreus](#) zählen unter anderem die Administration von Produktionssystemen, Storage, Domain Name System (DNS), Datenbanken, Virtualisierungsplattformen sowie Netzwerktechnik.

Unter den essentiellen Services der Infrastruktur ist der Bereich der Netzwerktechnik jedoch besonders wichtig und stellt den Kern der Infrastruktur dar. Zu den Systemen, welche in diesen Bereich fallen zählen z.B. Loadbalancer, Firewalls, Router, Proxys sowie Switches. Diese Systeme werden von den Mitarbeitern der Infrastruktur genutzt, um den Zugriff auf interne Systeme und den allgemeinen Datenfluss zu kontrollieren. Aufgrund der Komplexität des innerbetrieblichen Netzwerks ist zur Administration des Letzteren für die Mitarbeiter der Infrastruktur ein breit gefächertes Know-How essentiell. Jedoch müssen ebenfalls jegliche anderen Abteilungen einen guten Überblick über die Topologie des Netzwerks behalten. Vor Allem beim Auftreten von Störungen gilt es, schnell an Informationen zu gelangen, um die Störungsursache schnellstmöglich zu beseitigen. Daher sind adäquate Mittel zur Kommunikation, Analyse sowie zum Informationsaustausch zwischen der Infrastruktur und weiteren Abteilungen zwingend erforderlich.

### 1.2 Projektbeschreibung

Die Infrastrukturabteilung bei [Boreus](#) erteilt u.a. Freigaben für den Zugriff von Kundensystemen zu z. B. anderen Systemen des Kunden, in das Internet oder auch via Tunnel in verschiedene Netzwerke. Hierfür ist es nötig, auf verschiedenen Infrastrukturkomponenten wie Firewalls, Switchen und Proxies Freigaben zu konfigurieren. Grundsätzlich wird hier eine Whitelist genutzt. Das bedeutet, dass ein System nur über speziell konfigurierte Freigaben in andere Netze oder mit anderen Systemen kommunizieren kann.

Die erste Flusskontrolle des Kommunikationsweges geschieht beim Gateway des Kundennetzes. Bei [Boreus](#) agiert für gewöhnlich eine Firewall als Router und somit ebenfalls

als Gateway für das Netzwerk des Kunden. Auf den Firewalls sind so genannte Routingregeln konfiguriert, welche statisch, das bedeutet manuell durch einen Mitarbeiter, oder dynamisch, also durch einen Automatismus gepflegt werden. Diese Regeln bestimmen den Weg, welches ein Paket, anhand des Netzwerks aus dem ein Paket stammt, verfolgt. Dies kann in andere Netze derselben Firewall oder auch über weitere Firewalls geschehen, welche wiederum Regeln zum Routen des Datenpakets enthalten. Das Nachvollziehen des Weges, welchen ein Datenpaket eines Systems verfolgen soll, gestaltet sich, aufgrund der Komplexität der Daten, oft schwierig und liegt nicht im Aufgabenbereich anderer Abteilungen, weshalb regelmäßig, mithilfe der Infrastrukturabteilung, Problemlösungen erarbeitet werden müssen. Da dieser Prozess einen hohen Zeitaufwand mit sich bringt und die Entstörung solcher Probleme schnellstmöglich geschehen muss, ist eine Lösung zur Vereinfachung der Analyse des Problems zu schaffen.

### 1.3 Projektziel

Ziel des Projektes ist die Erfassung jeglicher Routingregeln und halten dieser Daten innerhalb einer Datenbank zur visuellen Veranschaulichung der Regeln innerhalb eines gerichteten Kraft-Knoten-Graphen<sup>1</sup>. Der Graph soll über eine Webanwendung abrufbar sein. Hierbei soll es eine Möglichkeit zum Filtern der Daten nach Netzwerkadressen oder einer einzelnen Internet Protocol (IP) geben. Je gefundener Firewall, welche das angegebene Netz routet, sollen sich auch die einzelnen, sich auf das Netzwerk oder die IP beziehenden Routingregeln anzeigen lassen können. Die leichte Erreichbarkeit der Webanwendung sowie die Möglichkeit zum Filtern der Daten soll den Arbeitsablauf zur Störungsanalyse beschleunigen und vereinfachen.

### 1.4 Projektbegründung

Der Zeitaufwand und die mangelnden Mittel zum schnellen, einfachen und verständlichen vermitteln von Informationen bezüglich des Routings sowie der dadurch regelmäßig unterbrochene Arbeitsfluss sind die größten Hürden bei der Entstörung eines Problems. Aufgrund des mit dem jeweiligen Kunden vereinbarten Service Level Agreement (Vertrag mit dem Kunden, welcher z.B. Reaktionszeiten festlegt) (SLA) gilt es hier, schnellstmöglich zu reagieren und zu handeln. Die Kommunikation mit der für den jeweiligen Kunden verantwortlichen Abteilung gestaltet sich hier meist schwierig, da die Interpretation der Routingregeln nicht immer eindeutig ist. Aufgrund dieser Probleme hat sich Boreus dazu entschieden diesen Prozess mittels der Entwicklung der Routemap zu vereinfachen.

---

<sup>1</sup>Aus Knoten und Kanten bestehender Graph, dessen Anordnung sich aus errechneten Kräften ergibt

## 1.5 Projektschnittstellen

Damit die Webanwendung, welche den Graphen zeichnet, die Datensätze, welche zum Generieren des Graphens nötig sind, ermitteln kann, benötigt es eine Schnittstelle zur Kommunikation mit der Datenbank. Diese soll ein selbst entwickeltes [API](#) darstellen, welches als einzelner Endpunkt anhand des Querystrings<sup>2</sup> der aufgerufenen [URL](#) Daten aus der Datenbank entnehmen und zurückliefern soll. Sowohl die Webanwendung als auch das [API](#) soll durch einen bereits installierten Apache Webserver<sup>3</sup> mit dem bereits genutzten Modul `mod_fcgid`<sup>4</sup> ausgeliefert werden. Als Datenbank soll die relationale Datenbank MariaDB<sup>5</sup> genutzt werden, da es bei [Boreus](#) bereits eine bestehende Datenbankinstanz gibt, in der Daten anderer Netzwerkkomponenten vorhanden sind. Zur Herstellung der Verbindung zwischen dem [API](#) und der Datenbank wird die offizielle C-Programmbibliothek MariaDB-Connector<sup>6</sup> genutzt.

## 1.6 Projektabgrenzung

Da sowohl die Mariadb Datenbank als auch der Apache2 Webserver inklusive dem Modul `mod_fcgid` bereits besteht, wird die Installation dieser sowie die Konfiguration der Datenbank nicht Teil dieses Abschlussprojekts sein. Lediglich das Schema soll hier erarbeitet sowie implementiert und die Apache2 Konfiguration angefertigt werden.

# 2 Projektplanung

## 2.1 Projektphasen

Vor Beginn des Abschlussprojekts wurde die zur Verfügung stehende Zeit in verschiedene Phasen eingeteilt, welche sich auf 70 Stunden belaufen. Diese Einteilung ist der Tabelle 1: [Zeitplanung](#) zu entnehmen. Es wird jeweils die Phase zusammen mit der geplanten Zeit dargestellt. Die Hauptphasen lassen sich zusätzlich in kleinere Unterphasen, welche zum Beispiel einen einzelnen Arbeitsschritt oder Prozess darstellen können, unterteilen.

---

<sup>2</sup>Teil der Uniform Resource Locator ([URL](#)), welcher Schlüssel-Wert Paare enthält

<sup>3</sup>Vgl. <https://httpd.apache.org>

<sup>4</sup>Vgl. [https://httpd.apache.org/mod\\_fcgid/](https://httpd.apache.org/mod_fcgid/)

<sup>5</sup>Vgl. <https://mariadb.org/about/>

<sup>6</sup>Vgl. <https://mariadb.com/kb/en/mariadb-connector-c-3112-release-notes/>

Projektphase	Geplante Zeit
Planungs- und Analysephase	5 h
Entwurfsphase	10 h
Implementierungsphase	30 h
Abnahmetest der Fachabteilung und Deployment	5 h
Erstellen der Dokumentation	20 h
<b>Gesamt</b>	<b>70 h</b>

Tabelle 1: Zeitplanung

## 2.2 Abweichungen vom Projektantrag

Abweichend vom Projektantrag werden Datensätze in der Datenbank nicht aktualisiert, sondern partiell je Firewall gelöscht und neu angelegt. Der Grund für diese Änderung ist, dass sich die inkrementelle Pflege der Datensätze als schwierig gestaltet, da regelmäßig die Datensätze auf alte Daten geprüft und adäquat angepasst werden müssten. Daher wurde sich dafür entschieden, wöchentlich die Tabellen der Datenbank zu trunkieren<sup>7</sup> und jegliche Datensätze neu anzulegen, indem die Daten jeder Firewall neu eingelesen werden. Eine eventuelle Anpassung des Programms zur inkrementellen Datenthaltung könnte jedoch zu einem späteren Zeitpunkt vorgenommen werden.

## 2.3 Ressourcenplanung

In der Übersicht, welche sich im Anhang [A.2: Verwendete Ressourcen](#) auf Seite [ii](#) befindet, sind alle Ressourcen aufgelistet, die für das Projekt eingesetzt wurden. Damit sind sowohl Hard- und Softwareressourcen als auch das Personal gemeint. Es wurde besonders Wert darauf gelegt, bei genutzten Third-Party Programmbibliotheken auf eine Open-Source Lizenz zu achten, um die anfallenden Projektkosten auf ein Minimum zu begrenzen.

## 2.4 Entwicklungsprozess

Vor der Realisierung des Projektes musste vom Autor ein passender Entwicklungsprozess gewählt werden. Aufgrund der vielen Teilergebnisse, welche sich durch dieses Projekt bilden werden, fiel die Entscheidung auf einen agilen Entwicklungsprozess. Ein iteratives Durchlaufen der Projektphasen sowie die kontinuierliche Rücksprache mit den Stakeholdern<sup>8</sup> wird durch die agile Herangehensweise ermöglicht und fördert die zeitliche Präsentation von Resultaten. Aufgrund dieser Tatsache wurde verhältnismäßig

---

<sup>7</sup>Das Trunkieren bezeichnet das Leeren der Tabellen

<sup>8</sup>Sämtliche am Projekt beteiligte Personen und Institutionen

wenig Zeit in die Entwurfsphase<sup>9</sup> der Projektplanung eingeplant, da sich Teile dieser Phase erst bei vorranschreitender Entwicklung der Software bilden. Ebenfalls stellen sich die kurzen Reaktionszeiten auf spontane Änderungen oder Wünsche als Vorteil dieses Entwicklungsprozesses dar.

## 3 Analysephase

### 3.1 Ist-Analyse

Wie bereits im Abschnitt 1.2: [Projektbeschreibung](#) beschrieben, ist die Infrastrukturabteilung für die Administration des innerbetrieblichen Netzwerks und den dazugehörigen Systemen zuständig. Somit sind die Mitarbeiter dieser Abteilung der erste Ansprechpartner für Probleme oder sonstige Fragen bezüglich des angewandten Routings.

Da es bisher keine Übersicht der Regeln<sup>10</sup> gibt, werden verschiedene Befehlszeilen-Programme zur Analyse genutzt oder sich direkt mit dem routenden System verbunden und die Regeln manuell abgefragt. Dies ist ein aufwändiger Prozess, welcher nur von der Infrastrukturabteilung durchgeführt werden kann. Ferner sind die erfassten Daten für Mitarbeiter anderer Abteilungen, aufgrund der komplexen Netzwerktopologie, meist nur schwer nachvollziehbar und der Arbeitsfluss der Mitarbeiter wird häufig unterbrochen. Auch nicht mehr zielführende Routen sind somit nur schwer bis gar nicht auffindbar. Zur Nachvollziehbarkeit des gesamten Prozesses der Störungsanalyse wurde eine EPK dargestellt. Diese Darstellung lässt sich unter [Abbildung 4: EPK des Prozesses der Störungsbearbeitung](#) finden.

### 3.2 Wirtschaftlichkeitsanalyse

Die Umsetzung des Projektes ist aufgrund der in den Abschnitten 1.4: [Projektbegründung](#) sowie 3.1: [Ist-Analyse](#) Punkte dringend erforderlich. Die Wirtschaftlichkeit der Realisierung soll in den folgenden Abschnitten geklärt werden.

#### 3.2.1 „Make or Buy“-Entscheidung

Zwar gibt es auf dem Markt andere Produkte, welche die Anforderung dieses Projektes erfüllen, jedoch keine, welche ohne Mehraufwand mit der bestehenden Infrastruktur kompatibel ist oder vollautomatisiert Daten erheben und wiederverwendbar zur Verfügung stellen kann. Des Weiteren sind einige Produkte aufgrund ihrer Lizenzen oder

---

<sup>9</sup>Vgl. Tabelle 1: [Zeitplanung](#)

<sup>10</sup>Regeln bezieht sich hier auf die Routingregeln der verschiedenen Firewalls, bzw. Router



den damit verbundenen Kosten nicht mit unserer Firmenstrategie zur Nutzung fremder Software vereinbar. Auch die Möglichkeit, die erhobenen Daten sowie die Software zukünftig selbst zu erweitern, um eine Ankopplung an bestehende oder zu entstehende Services zu ermöglichen, war für die Entscheidung relevant. Keine fremde Softwarelösung konnte diese Anforderungen erfüllen, daher wurde sich dazu entschieden, eine betriebseigene Lösung zu entwickeln, damit eine maßgeschneiderte und frei erweiterbare Lösung besteht.

### 3.2.2 Projektkosten

Im Folgenden sollen die Projektkosten, welche durch die Entwicklung dieses Projektes entstehen, kalkuliert werden. Berücksichtigt werden müssen hier anfallende Personalkosten sowie jegliche Aufwendungen für verwendete Ressourcen<sup>11</sup>.

Da die genauen Personalkosten nicht herausgegeben werden dürfen, werden in der folgenden Kalkulation imaginäre Stundensätze genutzt, welche sich aus der Kombination des Bruttolohns sowie der Sozialabgaben ergeben. Für eine Stunde eines Auszubildenden wurden 15€ und für die eines Mitarbeiters 25€ angenommen. Ein pauschaler Stundensatz von 15€ wurde für die Nutzung der genannten Ressourcen veranschlagt. Die Kosten, die für die einzelnen Phasen des Projektes anfallen, sowie die gesamten Projektkosten lassen sich der Tabelle 2: [Kostenaufstellung](#) entnehmen und belaufen sich auf 2.460,00 €.

Vorgang	Mitarbeiter	Zeit	Personal	Ressourcen	Gesamt
Entwicklungskosten	1 x Auszubildender	70 h	1.050,00 €	1.050,00 €	2.100,00 €
Fachgespräch	2 x Mitarbeiter	3 h	150,00 €	90,00 €	240,00 €
Code-Review	1 x Mitarbeiter	2 h	50,00 €	30,00 €	80,00 €
Abnahme	1 x Mitarbeiter	1 h	25,00 €	15,00 €	40,00 €
					<b>2.460,00 €</b>

Tabelle 2: Kostenaufstellung

### 3.2.3 Amortisationsdauer

Im folgenden Schritt soll ermittelt werden, wann sich die Entwicklung dieses Projektes amortisiert hat. Der ermittelte Wert gibt Aufschluss darüber, ob die Umsetzung des Projektes wirtschaftlich sinnvoll ist und ob sich auf Dauer Kostenersparnisse erzielen lassen.

<sup>11</sup>Vgl. Anhang [A.2: Verwendete Ressourcen](#) auf Seite ii

Durch die Minimalisierung des Zeitaufwandes bei der Analyse einer Störungsursache bezüglich Routingproblemen sowie der notwendigen Informationssuche, welche sich durch die Anwendung dieses Projektes ergibt, ließen sich die Personalkosten reduzieren. Das konkrete Maß an eingesparter Zeit lässt sich der Tabelle 3: Zeiteinsparung entnehmen, wobei sich die tägliche Zeiteinsparung aus der folgenden Formel ergibt:

$$\text{Anzahl je Tag} \cdot (\text{Zeit(alt) je Vorgang} - \text{Zeit(neu) je Vorgang}) \quad (1)$$

Die für die Prozesse angegebenen Zeiten wurden durch Mitarbeiter des Fachbereiches (bezüglich des ursprünglichen Zeitaufwandes) sowie durch den Autor (bezüglich der resultierenden Zeitaufwandes) geschätzt.

Vorgang	Anzahl je Tag	Zeit (alt) je Vorgang	Zeit (neu) je Vorgang	Einsparung je Tag
Informationssuche & Analyse	1	10 min	1 min	9 min
<b>Zeiteinsparung je Arbeitsjahr (220 Arbeitstage)</b>				<b>1.980 min</b>

Tabelle 3: Zeiteinsparung

$$\frac{1.980 \text{ Minuten}}{60 \text{ Minuten/Stunde}} = 33 \text{ Stunden Einsparung je Arbeitsjahr} \quad (2)$$

Durch die gewonnene Zeiteinsparung können die jährlichen Kosten um

$$33 \text{ Stunden} \times \frac{(25 \text{ €} + 15 \text{ €})}{\text{Stunde}} = 1.320 \text{ €} \quad (3)$$

gesenkt werden. Daher kann die Amortisationsdauer wie folgt berechnet werden:

$$\frac{2.460,00 \text{ €}}{1.320,00 \frac{\text{€}}{\text{Jahr}}} = 1,9 \text{ Jahre} \approx 1 \text{ Jahr } 47 \text{ Wochen} \quad (4)$$

Zur bildlichen Veranschaulichung wurde die Amortisation zusätzlich graphisch dargestellt. In der Grafik im Anhang A.3: Amortisationsdauer auf Seite iv stehen sowohl die variablen Kosten je Jahr der alten sowie der neuen Lösung zum Vergleich. Die durch die Entwicklung entstandenen Kosten sind ebenfalls mit in die Veranschaulichung eingeflossen.

Anhand des Resultats der Berechnung sowie der Grafik lässt sich ermitteln, dass rein aus wirtschaftlichen Gründen, die Umsetzung des Projektes sinnvoll ist. Die Voraussetzung hierfür ist, dass die Anwendung mindestens über die errechnete Zeit von ca. 1,9 Jahren genutzt wird, damit sich Anschaffungskosten und Kosteneinsparung ausgleichen.

### 3.3 Nutzwertanalyse

Zwar ist die Umsetzung dieses Projektes allein aus dem wirtschaftlichem Aspekt sinnvoll, aber auch bei längerer Amortisationsdauer wäre die Entwicklung dieses Programms profitabel, da bei [Boreus](#) der Support des Kunden eine höhere Priorität besitzt als die Vermeidung eventuell anfallender Kosten zur Verbesserung des Ersteren. Da jeder Kunde ein [SLA](#) besitzt und Reaktionszeiten von einem Maximum von 15 Minuten üblich sind, ist es hier von höchster Priorität, Möglichkeiten zur Verkürzung der Entstörungszeit zu finden. Da die Umsetzung dieses Projektes die besagte Zeitersparnis mit sich bringt und darüber hinaus Kommunikationswege verkürzt und vereinfacht, wäre die Umsetzung auch bei schlechteren wirtschaftlichen Gegebenheiten sinnvoll gewesen. Desweiteren bietet es sich an, die bei der Installation errichtete Datenbank mitsamt Datensätzen für weitere Projekte wieder zu verwenden oder die Daten in Kombination mit weiteren Daten zu nutzen.

### 3.4 Anwendungsfälle

Im Laufe der Analysephase wurde ein Use-Case-Diagramm erarbeitet, welches veranschaulichen soll, welche Anwendungsfälle das umzusetzende Programm abdecken soll. Dieses Diagramm befindet sich im Anhang [A.5: Use Case-Diagramm](#) auf Seite [vi](#).

### 3.5 Qualitätsanforderungen

Um die Aufwände zur nachträglichen Wartung und Pflege der Anwendung zu minimalisieren und die Nutzung so einfach wie möglich zu gestalten, fiel die Wahl der Art des Programms auf eine Webanwendung. Die daraus resultierenden Vorteile sind z. B. wenige bis gar keine Kompatibilitätsprobleme, da die Anwendung in den gängigsten Browsern funktionieren können und keine Installation notwendig sein wird. Für das Backend<sup>12</sup> liegt der Fokus auf Ressourcen- sowie Laufzeiteffizienz. Die Menge an zu bearbeiten Daten ist nicht unerheblich und kann eine lange Laufzeit nach sich ziehen. Zudem ist das Zielsystem eine Kernkomponente der innerbetrieblichen Infrastruktur und darf in seiner sonstigen Funktion unter keinen Umständen durch das Programm beeinträchtigt werden. Damit keine Kompatibilitätsprobleme bei der Installation der Software auf dem Produktionssystem auftreten, wird der Quelltext bei jeder Änderung automatisiert kompiliert und eine kompilierte Programmdatei verfügbar gemacht.

---

<sup>12</sup>Backend bezieht sich hier auf das Programm zum parsen der Routingregeln und hinterlegen der Daten in die Datenbank

### 3.6 Lastenheft/Fachkonzept

Das Lastenheft beinhaltet alle Anforderungen des Auftraggebers an die zu entwickelnde Anwendung. Dieses wurde am Ende der Analysephase zusammen mit der Infrastruktur-Abteilung erstellt und befindet sich auszugsweise im Anhang [A.4: Lastenheft \(Auszug\)](#) auf Seite [v](#).

## 4 Entwurfsphase

### 4.1 Zielplattform

Aufgrund der bereits im Abschnitt [3.5: Qualitätsanforderungen](#) sowie im Anhang [A.4: Lastenheft \(Auszug\)](#) auf Seite [v](#) genannten Punkte wurde sich für Javascript als Programmiersprache für die Webanwendung sowie C++ für das Programm zum Parsen der Routingregeln und dem Anlegen der Datensätze in der Datenbank entschieden. Javascript ist flexibel anwendbar und auf jedem bekannten Browser lauffähig. C++ ist sowohl sehr laufzeit- als auch ressourceneffizient und der Compiler bereits auf dem Produktionssystem vorhanden, auf dem es installiert werden soll. Für das [API](#) fiel die Wahl auf Python, da es eine hervorragende Programmbibliothek gibt, um sich das Fast Common Gateway Interface ([FCGI](#))-Modul des Apache2 Webserver zu Nutze zu machen. Ferner ist Python eine der Standardsprachen bei [Boreus](#). Beim Webserver und der Datenbank wurde sich zu einem Apache2 Webserver sowie einer MariaDB-Datenbank entschieden, da diese bereits bestehen und sich zur Nutzung von dem zu entwickelnden Programm anbieten. Jegliche Anwendungen werden auf Linux-Derivaten deployed.

### 4.2 Architekturdesign

Bei der Architektur der Anwendung wurde sich für ein Model View Adapter Design entschieden. Die View ist die Webanwendung, welche die Daten visualisiert, die sie von dem [API](#) bekommt. Das [API](#) stellt hier den Adapter dar, da es die Daten aus dem Model extrahiert und an die View weiterreicht. Das Model ist hierbei die Datenbank. Der Vorteil dieses Architekturdesigns ist, dass die View und das Model nur den Adapter kennen, sich allerdings nicht gegenseitig und daher voneinander unabhängig sind. Das bedeutet, dass, solange die View oder das Model kompatibel mit dem Adapter bleiben, sie jeweils geändert werden können, ohne die jeweils andere Komponente zu beeinflussen.

### 4.3 Entwurf der Benutzeroberfläche

Da die Anwendung hauptsächlich in der Störungsanalyse genutzt werden wird, ist es essenziell, dass die Oberfläche simpel und intuitiv bedienbar ist. Daher wurde durch den Autor ein Mock-Up der geplanten Benutzeroberfläche mit der Hilfe des Online-Tools [draw.io](https://draw.io)<sup>13</sup> erstellt, welches in [Abbildung 6: Mock-Up der Benutzeroberfläche](#) zu finden ist. Nachdem der Benutzer beim Aufruf der Webanwendung nach der zu routenden Netzwerkadresse gefragt wurde, soll ein Knoten-Graph generiert und angezeigt werden. Zusätzlich soll über die Bewegung des Mauszeigers über einen Knoten ein Beschriftungsfeld mit dem Namen des Routers auftauchen. Ferner sollen sich die auf das Netzwerk passenden Routingregeln des ausgewählten Routers per Click auf den Knoten anzeigen lassen. Dies soll per Pop-Up<sup>14</sup> dargestellt werden.

### 4.4 Datenmodell

Um alle Parameter einer Routingregel in der Datenbank abbilden und Datenredundanz vermeiden zu können, wurde nach ergiebiger Analyse der zu verarbeitenden Daten zuerst ein vereinfachtes Entity-Relationship-Modell (ERM) erstellt, welches Entitäten, Relationen sowie die dazugehörigen Kardinalitäten abbildet und sich in [Abbildung 2: Vereinfachtes ER-Modell](#) finden lässt. In [Abbildung 3: Tabellenschema der Datenbank](#) wird zusätzlich ein Tabellenschema dargestellt, welches sich aus dem ERM ableiten ließ.

### 4.5 Planung der Geschäftslogik

Das Sequenzdiagramm in der [Abbildung 5: Sequenzmodell der Anwendung der verschiedenen Komponenten](#) kann zur Planung der Geschäftslogik genutzt werden. Anhand dieser Darstellung wird ersichtlich, dass bei jeder Änderung der Routingregeln automatisch durch die Firewalls, welche bei [Boreus](#) als Router agieren, eine Datei mit den jeweiligen Routingregeln als Inhalt angelegt werden muss. Diese Daten müssen auf einem zentralen Verwaltungssystem hinterlegt und versioniert<sup>15</sup> werden. Bei jedem dieser Vorgänge muss automatisch das Programm zum Einlesen dieser angelegten Dateien gestartet werden, welches den Inhalt dieser Dateien parsen und in der Datenbank speichern muss. Wurden die Datensätze aller zu lesenden Dateien in der Datenbank hinterlegt, müssen die Datensätze zur Auslieferung durch das [API](#) bereit stehen.

---

<sup>13</sup>Vgl. <https://draw.io/>

<sup>14</sup>Pop-Up bezeichnet einen Vorgang, in dem sich ein Fenster o.Ä. vor die Oberfläche bewegt

<sup>15</sup>Zur Versionsverwaltung soll die Software git genutzt werden

Schlussendlich sollen die Datensätze sowie die Zusammenhänge dieser über den generierten Kraft-Knoten-Graphen innerhalb der Webanwendung visualisiert werden. Daher muss nach dem Aufruf der Webanwendung vom Benutzer eine Netzwerkadresse angegeben werden, zu der über das [API](#) die dazugehörigen Datensätze aus der Datenbank ermittelt werden müssen. Hier wird es nötig sein, mit einer rekursiven [CTE](#) die Zusammenhänge der Router anhand der vorhandenen Datensätze zu ermitteln. Da für die Generierung des Graphen eine Javascript-Programmbibliothek<sup>16</sup> genutzt werden soll, muss das [API](#) zusätzlich die abgefragten Datensätze in die Javascript Object Notation ([JSON](#)) umwandeln, um eine reibungslose Einbindung der Daten in das Frontend zu gewährleisten.

## 4.6 Maßnahmen zur Qualitätssicherung

Zur Sicherstellung der Kompatibilität der Anwendung mit dem System muss eine [CI](#)-Pipeline in der Versionsverwaltung angelegt werden, damit bei jeder Änderung des Quellcodes dieser auf der aktuellen Betriebssystemversion sowie jeder neueren Version des Betriebssystems kompiliert wird. Hiermit muss sichergestellt werden, dass das Programm auf dem Endsystem lauffähig ist. Nach jedem Durchlauf dieser Pipeline muss das lauffähige Programm zur Vorbereitung auf das Deployment als Artefakt<sup>17</sup> bereitgestellt werden. Es müssen die Betriebssystemversionen Linux Debian<sup>18</sup> 9 Stretch sowie Linux Debian 10 Buster getestet werden. Weiterhin muss der Javascript-Quelltext minified<sup>19</sup> werden, um eine schnellstmögliche Auslieferung der Webseite zu gewährleisten.

## 4.7 Pflichtenheft/Datenverarbeitungskonzept

Zum Ende der Entwurfsphase wurde ein Pflichtenheft erstellt. Dieses beschreibt die Mittel und Pläne, mit der der Autor die Anforderungen an das Projekt umsetzen möchte. Ein Auszug des auf dem Lastenheft (siehe Kapitel 3.6: [Lastenheft/Fachkonzept](#)) aufbauenden Pflichtenhefts ist im Anhang [A.6: Pflichtenheft \(Auszug\)](#) auf Seite [vii](#) zu finden.

---

<sup>16</sup>Vgl. <https://github.com/vasturiano/force-graph>

<sup>17</sup>Nach jedem Durchlauf der Pipeline wird die Testumgebung für gewöhnlich gelöscht. Ein Artefakt bleibt jedoch bestehen und kann heruntergeladen werden.

<sup>18</sup>Vgl. <https://www.debian.org>

<sup>19</sup>Die Minification bezeichnet das Komprimieren des Quelltextes

## 5 Implementierungsphase

### 5.1 Iterationsplanung

Wie bereits im Abschnitt 2.4: [Entwicklungsprozess](#) erwähnt, wurde sich für einen agilen Entwicklungsprozess bei der Projektdurchführung entschieden. Zur Unterstützung des iterativen Durchlaufens der Projektphasen wurde vor Beginn der Implementierungsphase ein Iterationsplan erstellt. Dieser lässt sich im Anhang A.18: [Iterationsplan](#) auf Seite xxii finden und zeigt in chronologischer Reihenfolge, welche Aufgaben zu erledigen sind.

### 5.2 Implementierung der Datenstrukturen

Die Implementierung des im Abschnitt 4.4: [Datenmodell](#) erwähnten Tabellenschemas für die zu erstellende Anwendung wurde laut Planung durchgeführt. Es wurde initial von einem Datenbankadministrator ein neues Schema sowie ein Benutzer mit den Zugriffsrechten auf dieses Schema erstellt. Der Autor implementierte darauffolgend die Tabellen sowie zwei zusätzliche Views<sup>20</sup> zur Vereinfachung der Handhabung im späteren Verlauf. Ferner wurde eine Funktion zur Nutzung in der CTE hinterlegt, mit welcher geprüft werden kann, ob sich eine IP innerhalb eines angegebenen Netzwerkes befindet. Im Angang wurde sowohl ein Listing der CTE (Vgl. Anhang A.19.2: [Listing der CTE](#) auf Seite xxiv) als auch ein Listing der Funktion (Vgl. Anhang A.19.3: [Listing der SQL-Funktion](#) auf Seite xxv) hinterlegt.

### 5.3 Implementierung der Geschäftslogik

Da durch die Implementierung der Geschäftslogik der meiste Aufwand bei der Umsetzung des Projektes verursacht wurde, soll an dieser Stelle der unterliegende Prozess genauer erläutert werden. Zur effizienten Umsetzung der Implementierung dieser ist es nötig, eine adäquate Entwicklungsumgebung zu wählen. Für die Programmertätigkeiten hat sich der Autor jedoch für zwei Texteditoren entschieden. Im Allgemeinen wurde hier der Texteditor Vim<sup>21</sup>, bei der Implementierung komplexerer Strukturen hingegen Visual Studio Code<sup>22</sup> genutzt.

Als erster Schritt der Implementation wurde das ‘Grundgerüst’ des Programms geschaffen. Es wurde ein GitLab Projekt erstellt, in dem neben der README-Datei<sup>23</sup>

---

<sup>20</sup>Als View können SELECT-Befehle mit einem Namen hinterlegt werden, damit diese Befehle leichter wiederaufrufbar sind.

<sup>21</sup>Vgl. <https://www.vim.org>

<sup>22</sup>Vgl. <https://github.com/microsoft/vscode>

<sup>23</sup>Datei, in der eine grobe Zusammenfassung der relevantesten Informationen zu finden ist.



und der Lizenz auch die Konfigurationsdatei für die CI-Pipeline sowie Dockerfiles<sup>24</sup> zum Erstellen von Containern für die Testumgebung angelegt wurden. Ein Screenshot der lauffähigen Pipeline ist in der [Abbildung 10: Erfolgreich durchgeführte Pipeline](#) zu sehen. Nachgehend wurde die Verzeichnisstruktur erstellt, welche vorerst mit Platzhalterdateien befüllt wurde. Ein Screenshot der Verzeichnisstruktur befindet sich in der [Abbildung 7: Jegliche Verzeichnisse und Dateien des Projektes](#). Nach Fertigstellung des Grundgerüsts konnte mit der tatsächlichen Implementation begonnen werden.

Zunächst wurden das Utility-, Logging- sowie Config-Modul entwickelt und manuell auf ihre Funktionalität getestet. Das Utility-Modul handhabt unter anderem die Prüfung, ob sich eine angegebene IP-Adresse in einem Netzwerk befindet. Letztere Funktionalität wird gebraucht, da es Fälle geben kann, in denen es keine Scope-Link-Route<sup>25</sup> zu einer Netzwerkadresse gibt. Sofern eine IP gefunden wird, zu der keine Scope-Link-Route gefunden werden kann, wird mit dieser Funktion ermittelt, welchem Netzwerk sie zugehörig ist. Das Logging-Modul besitzt eine Klassenmethode zur Ausgabe von Informationen sowie Fehlerausgaben. Damit die beim Start des Programms angegebene Verboseität<sup>26</sup>, welche ein Attribut der Klasse ist, in jedem Scope erhalten bleibt, wurde sich hier für die Anwendung der Singleton-Pattern<sup>27</sup> entschieden. Das Config-Modul liest beim Programmstart die Konfigurationsdatei ein und gibt jegliche Key-Value-Paare der Konfiguration als `std::unordered_map` zurück. Die erhaltenen Werte werden über den Programmlauf als formale Parameter an die jeweiligen Funktionen oder Methoden weitergegeben.

Als nächster Schritt des Iterationsplans wurde das Datenbank-Modul implementiert. Hier wurde ebenfalls das Singleton-Pattern angewandt, damit sichergestellt werden kann, dass unter keinen Umständen mehr als eine Verbindung zum Datenbankserver etabliert wird. Die Klasse bietet Methoden zum Verbinden zu der Datenbank und dem damit verbundenen Starten einer Transaktion sowie dem Schließen der Verbindung. Ferner enthält die Klasse eine polymorphe Methode zum Absenden von Queries an die Datenbank. Sofern eine `std::vector<std::vector<std::string>>`-Objektreferenz als formaler Methodenparameter übergeben wird, wird zusätzlich zum Absenden der Query das Output des SQL-Befehls in dem besagten Objekt gespeichert. Ein Auszug des Datenbankmoduls befindet sich im Anhang [Anhang A.19.1: Listing des Datenbank-Moduls](#) auf Seite [xxiii](#).

Als letzter Schritt der Implementierung des Parser-Programms wurden die Funktionen des Parser-Moduls programmiert. Dieses Modul enthält ebenfalls eine polymorphe

---

<sup>24</sup>Dockerfiles definieren, wie ein Docker-Container gebaut werden soll (z. B. Welches Betriebssystem)

<sup>25</sup>Route, welche sich auf ein am jeweiligen System anliegendes Netzwerk bezieht

<sup>26</sup>Bezeichnet die Menge der Ausgabe des Programms.

<sup>27</sup>Vgl. [https://de.wikipedia.org/wiki/Singleton\\_\(Entwurfsmuster\)](https://de.wikipedia.org/wiki/Singleton_(Entwurfsmuster))



Funktion, welche die Pfade der Dateien ermittelt, in denen sich die Routingregeln befinden und diese, sofern noch Unterverzeichnisse mit weiteren Dateien bestehen, sich selbst aufrufen. Wurde der absolute Pfad zu einer Datei gefunden, wird diese eingelesen und je Zeile der Datei die letzte Funktion des Moduls aufgerufen. Diese parst die jeweilige Routingregel, prüft diese auf Vollständigkeit und schickt ca. sieben bis zehn Queries an die Datenbank, um die verschiedenen Datensätze anzulegen.

Die nächste Aufgabe ist das [API](#) bereitzustellen. Letzteres besteht aus einem einzelnen Endpunkt, der anhand der im Querystring angegebenen Parameter entscheidet, wie Daten aus der Datenbank extrahiert und verarbeitet werden müssen, bevor sie dem Anfragenden zurückgegeben werden. Es werden die Parameter `user`, `pass`, `network` sowie der optionale Parameter `router` erwartet. Ist `router` gegeben, werden die Routingregeln zurückgegeben, welche auf dem Router konfiguriert wurden und sich auf das ebenfalls angegebene Netz beziehen. Andernfalls wird ein [JSON](#)-Objekt zurückgegeben, welches die Konstellation der Router zueinander beschreibt und vom Javascript visualisiert werden soll.

Da das Parser-Programm nach Deployment des Projektes wöchentlich gestartet werden soll, wurde eine Cron-Konfiguration vorbereitet. Ein Ausschnitt dieser befindet sich in der [Abbildung 9: Cronjob-Konfiguration](#). Weiterhin wurde die Konfiguration des Apache2-Webservers angepasst, damit das [API](#) durch diesen ausgeliefert werden kann. Ein Ausschnitt dieser Konfiguration ist in der [Abbildung 8: Virtualhost-Konfiguration des Apache2-Webservers](#) abgebildet.

## 5.4 Implementierung der Benutzeroberfläche

Als letzte Aufgabe der Implementierungsphase ist das Javascript-Frontend zu implementieren. Das Script fragt beim Start den Benutzer, für welche Netzwerkadresse die das Netzwerk routenden Router angezeigt werden sollen, worauf die Eingabe zuerst validiert wird. Ist die Benutzereingabe in Ordnung, wird ein `XMLHttpRequest`-Objekt<sup>28</sup> erstellt. Per Aufruf des [API](#) wird desweiteren ein [JSON](#)-Objekt erstellt, welches über den Methodenaufruf `.graphData()` an das Graph-Objekt der genutzten Force-Graph-Programmbibliothek übergeben wird. Damit können die Daten im Knoten-Graphen visualisiert werden. Beim Klick auf einen Knoten werden die anzuzeigenden Daten ebenfalls per Aufruf des [API](#) abgefragt. Neben der Datei, in der sich der Javascript-Quelltext befindet, wurde ebenfalls eine HyperText Markup Language ([HTML](#))-Datei erstellt, in der das gesamte Script eingebunden und beim Aufruf der Webseite die `init()`-Funktion aufgerufen wird.

---

<sup>28</sup>Vgl. <https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest>

Mit der Implementierung der Benutzeroberfläche konnte die Implementierungsphase erfolgreich abgeschlossen werden. Ein Screenshots der Anwendung in der Entwicklungsphase befindet sich im Anhang [A.17: Screenshots der Anwendung](#) auf Seite [xxi](#).

## 6 Abnahmephase

### 6.1 Abnahme durch den Fachbereich

Nachdem die Implementation der Anwendung abgeschlossen wurde, konnte diese dem Fachbereich zur Abnahme vorgelegt werden. Da für dieses Projekt die agile Softwareentwicklungsmethode gewählt wurde, konnte dem Fachbereich bereits nach jeder Iteration eine aktuelle Version der Anwendung vorgeführt werden. Daher waren sie bereits vor dem Ende der Implementation mit der Anwendung vertraut, wodurch sich die Abnahme zeitlich kurz gestaltete. Ebenso konnten durch diese Herangehensweise Hinweise und oder Kritik bereits während dem Durchführen der Iterationsphasen geäußert und berücksichtigt werden, sodass dem nächsten Schritt, dem Deployment, nichts mehr im Wege stand. Es wurde jedoch zusätzlich durch einen weiteren Entwickler ein Code-Review durchgeführt.

### 6.2 Deployment

Da die Anwendung im Versionierungstool GitLab inkrementell gepflegt wurde, bei jeder Änderung des Quellcodes automatisch kompiliert wurde, somit als Artefakt zur Verfügung stand und nur auf zwei Produktionssystemen deployed werden musste, gestaltete sich das Deployment verhältnismäßig simpel. Es wurden ein Benutzer auf den Systemen für die Ausführung des Programms sowie alle nötigen Verzeichnisse und Dateien (z. B. die Konfigurationsdatei und das Secure Socket Layer (SSL)-Common Authority (CA)-Zertifikat) auf den Systemen angelegt. Hiernach konnte das vorkompilierte Programm für die jeweilige Betriebssystemversion auf das System vom GitLab-Server heruntergeladen und die Konfiguration für den Cronjob wie bereits im Abschnitt [5.3: Implementierung der Geschäftslogik](#) genannt angepasst werden. Wie ebenfalls im selben Abschnitt erwähnt, wurde hier auch die Konfiguration des bereits installierten Apache2-Webserver angepasst. Nach dieser Anpassung konnte das [API](#) unter dem in der Konfiguration genannten Verzeichnis abgelegt werden. Ferner musste das Frontend deployed werden. Hierfür wurde ein neuer virtueller Host für den Apache2-Webserver angelegt, welcher die Webanwendung ausliefert. Als letzter Schritt wurde ein bereits bestehendes Bash-Script angepasst, welches die Routingregeln in die Dateien schreibt, welche die Anwendung einliest, sodass dieses ebenfalls das Programm mit Angabe des jeweiligen Routers startet.

Da die Fachbereiche, wie im vorherigen Schritt erwähnt, bereits mit der Anwendung vertraut waren, waren keine zusätzlichen Benutzerschulungen erforderlich.

## 7 Dokumentation

Die Dokumentation dieses Projektes setzt sich aus drei Teilbereichen zusammen. Diese werden im Folgenden weiter erläutert.

### 7.1 Projektdokumentation

Die Projektdokumentation wurde während der gesamten Durchführungszeit des Projektes angefertigt und beschreibt die gesamte Planung und Umsetzung des Projektes.

### 7.2 Anwenderdokumentation

Die Anwenderdokumentation wurde in Form einer README.md-Datei im Projektverzeichnis der Versionsverwaltung hinterlegt. Sie beschreibt die Funktionsweise, die Installation sowie die Benutzung des Programms. Sie soll dem Benutzer dienen, eventuelle Fragen direkt prüfen zu können. Ein Ausschnitt dieser Dokumentation ist der [Abbildung 11: Installationsanleitung der Anwenderdokumentation](#) zu entnehmen.

### 7.3 Entwicklerdokumentation

#### 7.3.1 API-Dokumentation

Für das [API](#) wurde eine separate Dokumentation angefertigt, welche sich im Wiki-Abschnitt des GitLab-Projektes befindet. Sie beschreibt den Endpunkt sowie die benötigten Parameter. Diese lässt sich der [Abbildung 12: Ausschnitt der API-Dokumentation](#) entnehmen.

#### 7.3.2 Code-Dokumentation

Jegliche Beschreibungen der Funktionalität sowie die Nutzung der Funktionen und Methoden befinden sich in Form von Kommentaren innerhalb des Quellcodes. Der Übersichtlichkeit halber wurden Funktions- und Methodenbeschreibungen in den Header-Dateien hinterlegt. Weitere Dokumentation befindet sich dennoch innerhalb des Restes des Quellcodes. Ein Auszug dieser Dokumentation lässt sich im bereits aufgezeigten Anhang [A.19.1: Listing des Datenbank-Moduls](#) auf Seite [xxiii](#) einsehen.

## 8 Fazit

### 8.1 Soll-/Ist-Vergleich

Rückblickend kann, in Hinsicht auf die Umsetzung des Abschlussprojektes, gesagt werden, dass die Anforderungen des Pflichtenhefts erfüllt wurden. Die in Abschnitt 2.1: [Projektphasen](#) genannten Phasen wurden eingehalten, wenn auch für einige der Phasen mehr oder weniger Zeit in Anspruch genommen wurde. Die Soll- und Ist-Werte der in Anspruch genommenen Zeiten wurden in der Tabelle [A.20: Soll-/Ist-Vergleich](#) zum Vergleich gestellt. Der Tabelle ist zu entnehmen, dass nur geringfügig von den geschätzten Zeiten abgewichen wurde. Die Abweichungen ziehen jedoch keine Abweichung der gegebenen 70 Stunden an Durchführungszeit nach sich, wodurch der Rahmen des Abschlussprojekts eingehalten werden konnte.

### 8.2 Lessons Learned

Der Autor konnte durch die Durchführung dieses Projektes wertvolle Erfahrungen im Bereich der Planung sowie Durchführung von Projekten sammeln. Besonders die Bedeutung der vorzeitigen Planung eines Projektes und die Auswirkungen dieser auf die Umsetzung des jeweiligen Projektes wurden hierbei deutlich. Auch die Vorteile einer agilen Softwareentwicklung wurden erneut untermalt, da sich die wiederholten Rücksprachen auf den Abnahmetest sowie das Deployment positiv auswirkten. Abschließend ist festzuhalten, dass sich die Umsetzung dieses Projektes nicht nur für [Boreus](#), sondern ebenfalls besonders für den Autor als bereichernd herausgestellt hat.

### 8.3 Ausblick

Zur Zufriedenheit des Stakeholders konnten jegliche Anforderungen an dieses Projekt realisiert werden. Weiterhin bietet dieses Projekt eine solide Grundlage für zukünftige Erweiterungen. Es könnten z. B. zusätzliche Ansichten oder Visualisierungen der Routingdaten entwickelt werden, oder diese nutzbringend in Zusammenhang mit weiteren Daten gebracht werden.

## 9 Literaturverzeichnis

ELECTRICMONK.NL - 2021

*Apache, FastCGI and Python*

[https://www.electricmonk.nl/docs/apache\\_fastcgi\\_python/apache\\_fastcgi\\_python.html](https://www.electricmonk.nl/docs/apache_fastcgi_python/apache_fastcgi_python.html)

MARIADB.COM - 2021

*About MariaDB Connector/C*

<https://mariadb.com/kb/en/about-mariadb-connector-c/>

SODOCUMENTATION.NET - 2021

*Singleton Design Pattern*

<https://sodocumentation.net/cplusplus/topic/2713/singleton-design-pattern>

GITLAB.COM - 2021

*GitLab CI/CD*

<https://docs.gitlab.com/ee/ci/>

DOCKER.COM - 2021

*Dockerfile Reference*

<https://docs.docker.com/engine/reference/builder/>

OVERLEAF.COM - 2021

*LaTeX Documentation*

<https://www.overleaf.com/learn>

## A Anhang

### A.1 Detaillierte Zeitplanung

<b>Planungs- und Analysephase</b>	<b>5 h</b>
1. Ist-Analyse (Fachgespräche, Use-Case-Diagramm etc.)	2 h
2. „Make or buy“-Entscheidung und Wirtschaftlichkeitsanalyse	1 h
3. Soll-Konzept	1 h
4. Erstellung des Lastenhefts	1 h
<b>Entwurfsphase</b>	<b>10 h</b>
1. Benutzeroberfläche entwerfen	1 h
2. Datenbankentwurf (Tabellen- und <a href="#">ERM</a> )	5 h
2.1. Erstellung des Tabellenschemas	4 h
2.2. Erstellung des <a href="#">ERM</a>	1 h
3. Planung der Geschäftslogik (Entscheidungstabelle)	2 h
4. Erstellung des Pflichtenhefts	2 h
<b>Implementierungsphase</b>	<b>30 h</b>
1. Anlegen der Datenbank	1 h
2. Erstellung der Benutzeroberfläche ( <a href="#">HTML</a> und Javascript)	4 h
3. Implementierung der Geschäftslogik	23 h
3.1. Implementierung der Hilfsmodule (Logging-, Config- und Utility-Modul)	3 h
3.2. Implementierung des Parser-Moduls	15 h
3.3. Implementierung des Datenbank-Moduls	2 h
3.4. Implementierung des <a href="#">API</a>	3 h
4. Konfiguration des Apache2-Webserver sowie des Crons	0,5 h
5. Konfiguration der Gitlab <a href="#">CI</a> -Pipeline	1,5 h
<b>Abnahmetest der Fachabteilung und Deployment</b>	<b>5 h</b>
1. Abnahmetest der Fachabteilung	3 h
2. Deployment der Webanwendung und des <a href="#">API</a>	1 h
3. Deployment des Programms auf den Produktionssystemen	1 h
<b>Erstellen der Dokumentation</b>	<b>20 h</b>
1. Erstellen der Projektdokumentation	15 h
2. Erstellen der Entwicklerdokumentation	3 h
3. Erstellen der Benutzerdokumentation	2 h
<b>Gesamt</b>	<b>70 h</b>

## A.2 Verwendete Ressourcen

### Hardware

- Büroarbeitsplatz
  - Laptop
  - Tastatur
  - Maus
  - Zweiter Monitor
  - Headset
- Homeofficearbeitsplatz
  - Desktop-Computer
  - Tastatur
  - Maus
  - Fünf Monitore
  - Kopfhörer
  - Mikrofon
- Root-Server

### Software

- Manjaro Linux Nibia mit Kernel 5.10.5 und Bash 5.1.0 - Betriebssystem
- Debian Linux Stretch sowie Buster - Betriebssystem des Servers und der Container
- Deepin-Terminal - Terminal-Emulator für die Bash
- Vivaldi - Browser
- Visual Studio Code - Texteditor
- VIM - Texteditor
- gcc - Compiler
- Python 3.8 - Python Interpreter

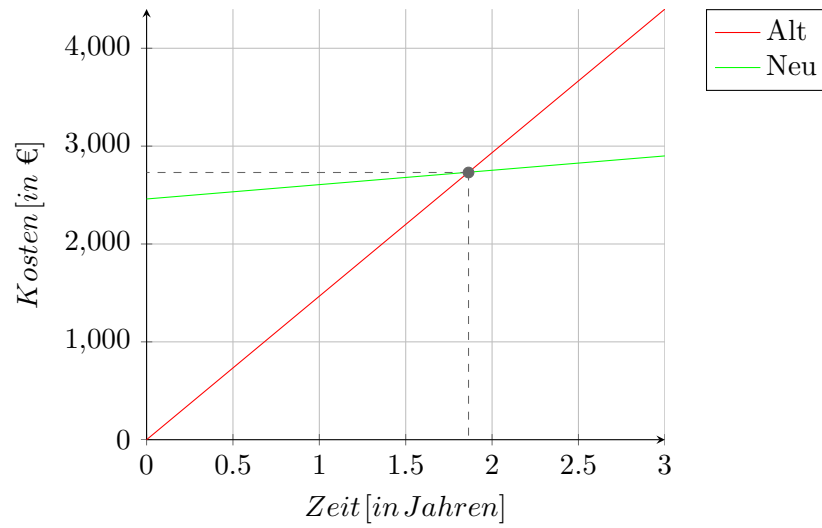
- git - Verteilte Versionsverwaltung
- GitLab - Git-Repository Manager
- Jira - Kanban-Software von Atlassian
- Docker - Containerisierungsplattform
- Kaniko - Docker Container Build-Tool
- SonarQube - Plattform zum Testen der Codequality
- MariaDB Programmbibliothek, Server und Client - Relationales Datenbanksystem
- Apache2 inkl. mod\_fcgid - Webserver inkl. [FCGI](#)-Modul
- Cron - Programm zur regelmäßigen Ausführung von Bash-Befehlen
- minify - Programm zum Minimieren und Obfuscieren von Javascript
- draw.io - Webanwendung zum zeichnen von UML-Diagrammen von [diagrams.net](#)
- overleaf.com - Online Entwicklungsumgebung zum Anfertigen von  $\text{\LaTeX}$ -Dokumenten

## Personal

- Projektleiter der Infrastruktur - Anforderungsanalyse und Projektabnahme
- Entwickler - Umsetzung des Projektes
- Anwendungsentwickler - Review des Codes und Test des Programms



### A.3 Amortisationsdauer



## A.4 Lastenheft (Auszug)

Es folgt ein Auszug aus dem Lastenheft mit Fokus auf die Anforderungen an die Anwendung:

Die Anwendung muss folgende Anforderungen erfüllen:

1. Verarbeitung der Daten
  - 1.1. Die Anwendung muss die von den Routern bereitgestellten Informationen bezüglich der Routingregeln verarbeiten.
  - 1.2. Die ermittelten Daten müssen auf Anomalien geprüft werden und es muss eine dementsprechende Ausgabe stattfinden.
  - 1.3. In einer bereitgestellten Datenbankinstanz müssen die ermittelten Daten zur Weiterverarbeitung hinterlegt werden.
  - 1.4. Die Anwendung muss auf Wunsch die Daten eines einzelnen Routers verarbeiten können.
2. Bereitstellung der Daten
  - 2.1. Es muss eine Schnittstelle ([API](#)) geschaffen werden, über die die Daten gefiltert ausgegeben werden können.
  - 2.2. Es muss nötig sein, sich zur Datenabfrage gegen die [API](#) zu authentifizieren.
3. Darstellung der Daten
  - 3.1. Die erfassten und über das [API](#) abfragbaren Daten müssen in einer Webanwendung dargestellt werden.
  - 3.2. Die Webanwendung muss die Daten mittels eines Kraft-Knoten-Graphen anzeigen können.
  - 3.3. Beim Aufruf der Webseite muss eine Abfrage stattfinden, durch die die Anzeige manipuliert werden kann.
  - 3.4. Hier muss nach Netzen gefiltert werden können.
4. Sonstige Anforderungen
  - 4.1. Die Nutzung der graphischen Darstellung der Daten darf keine Installation von zusätzlicher Software benötigen.
  - 4.2. Die Anwendung muss auf einem Linux System lauffähig sein.
  - 4.3. Sowohl die Webanwendung als auch das [API](#) muss mittels eines Apache2-Webservers ausgeliefert werden können.
  - 4.4. Die Anwendung muss jederzeit erreichbar sein.
  - 4.5. Das Backend muss mittels einer [CI-Pipeline](#)<sup>29</sup> vorkompiliert werden und als Atrefakt auf dem Produktionssystem deploybar sein.

---

<sup>29</sup>Eine Anreihung von sog. Jobs, zum Testen von Software

## A.5 Use Case-Diagramm

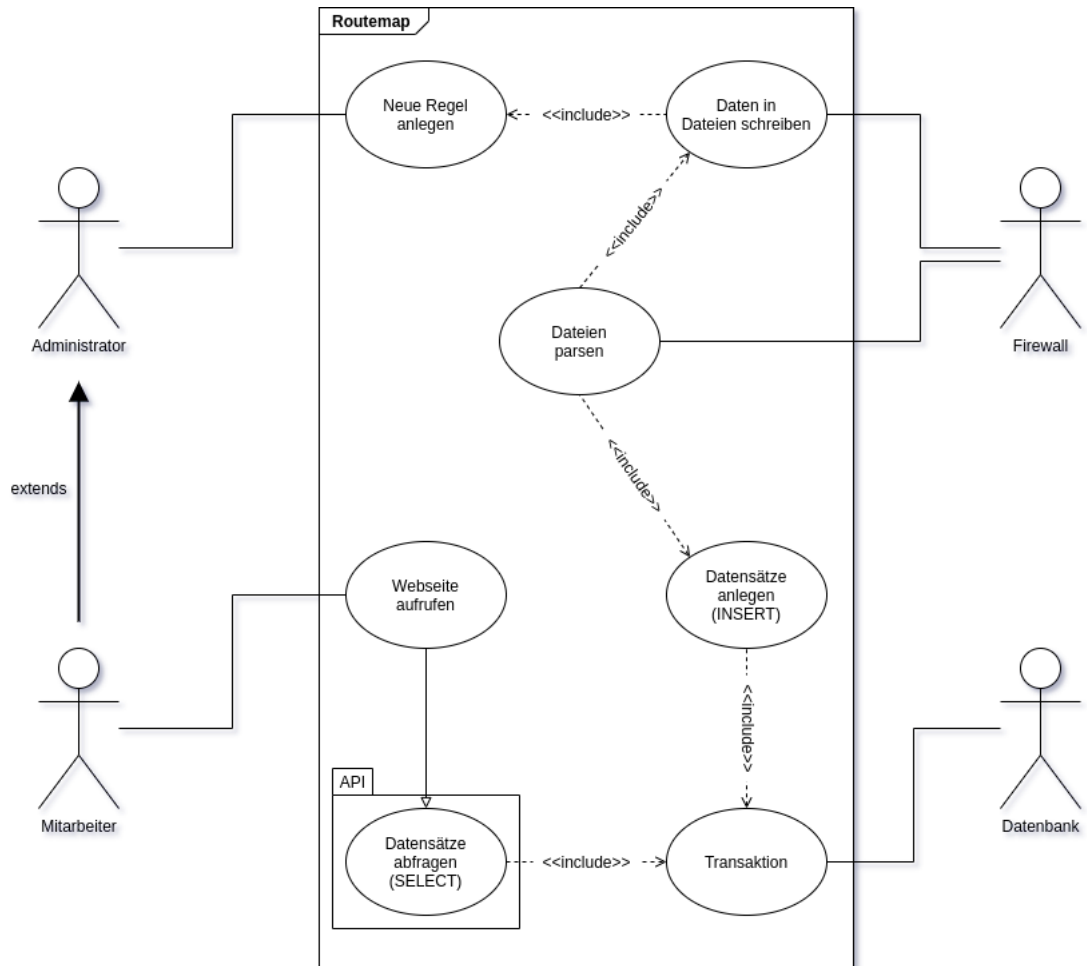


Abbildung 1: Use Case-Diagramm

## A.6 Pflichtenheft (Auszug)

### Umsetzung der Anforderungen

- Das C++-Programm zum Parsen der Routingregeln soll mit mehreren Kommandozeilenparametern aufrufbar sein.
  - Ein Parameter, durch welchen das Programm die Daten eines einzelnen Routers parst.
  - Ein Parameter, welcher das Programm ausgeben lässt, welche Queries an die Datenbank geschickt werden oder werden würden.
  - Ein Parameter, welcher das Programm veranlasst, keine tatsächliche Verbindung mit der MariaDB-Datenbank aufzubauen.
- Das Programm soll unter dem angegebenen Verzeichnis dynamisch alle vorhandenen Dateien, welche Routingregeln enthalten finden und einlesen. Dies soll mit dem in C++17 implementierten Feature der Standard-Bibliothek `std::filesystem::directory_iterator` geschehen. Zur Kompatibilität mit älteren Compilern soll hier anhand eines Compiler-Parameters die Nutzung des Namespaces `std::experimental::filesystem` möglich gemacht werden.
- Das Programm soll die Routingregeln beim Parsen auf Anomalien prüfen (z. B. fehlende Parameter) und gegebenenfalls eine Fehlermeldung ausgeben.
- Da Routen, welche sich nicht auf ein am System anliegendes Netzwerk beziehen (Routen mit dem Scope ‘link‘), einen weiteren Router als Ziel referenzieren, soll bereits vorher die Scope-Link-Route gefunden werden, welche besagt, dass das Netz an dem jeweiligen Router anliegt, damit dieser Zusammenhang in der Datenbank bereits besteht, wenn mit diesen Daten gearbeitet werden soll. Daher sollen die Scope-Link-Routen zuerst bearbeitet werden und sollen daher nach dem Lesen der Datei(en) in zwei separaten Datenstrukturen gehalten werden, bevor sie geparst werden.
- Das Programm soll eine Konfigurationsdatei einlesen können, damit änderbare Daten (z. B. die Datenbank, mit der sich verbunden werden soll) einfach umkonfiguriert werden können.
- Das Programm soll eine Verbindung mit der konfigurierten Datenbank aufbauen und anhand der Struktur der Regeln die Daten angepasst auf das Schema der Datenbank einpflegen.

- Zur Verbesserung der Performanz und zum Verhindern des Entstehens eines korrupten Datensatzes in der Datenbank im Falle eines unerwarteten Abbruchs der Programmausführung sollen automatische Commits<sup>30</sup> für die implizit erstellte Transaktion deaktiviert werden.
- Das [API](#), welches mit der Programmiersprache Python entwickelt werden soll, soll mit der flup<sup>31</sup>-Programmbibliothek die Nutzung des Apache2-Moduls `mod_fcgid` ermöglichen. Hierdurch soll erreicht werden, dass nicht bei jedem [API](#)-Aufruf ein neuer Prozess erzeugt wird, sondern eine bestimmte Anzahl an Prozessen besteht, welche die Anfragen bearbeiten.
- Das Python-[API](#) soll die im Querystring der [URL](#) angegebenen Parameter auslesen und sich mittels der dort angegebenen Benutzerdaten gegen die Datenbank authentifizieren sowie die Daten zum angegebenen Netzwerk von der Datenbank abfragen. Hierfür soll eine rekursive [CTE](#) genutzt werden.
- Im Fall der zusätzlichen Angabe eines Routernamens sollen alle Routen eines Routers ausgegeben werden, welche Pakete aus dem angegebenen Netzwerk routen.
- Das [API](#) soll die ermittelten Datensätze in die [JSON](#) umwandeln, bevor sie zurückgegeben werden
- Die Javascript Webanwendung soll die gewünschte Netzwerkadresse vom Benutzer erfragen, die Eingabe validieren und die dem Netzwerk zugehörigen Daten von dem [API](#) erfragen. Danach müssen die Daten mithilfe der Force-Graph Programmbibliothek visualisiert werden.
- Per Klick auf einen Knoten sollen die Routingregeln des jeweiligen Routers von dem [API](#) erfragt und angezeigt werden können.
- Beim Entwickeln der Webanwendung soll sich grob an das Mock-Up in der [Abbildung 6](#) gehalten werden.
- Sowohl bei den Queries des Parsing-Programms als auch des [API](#) soll das Input sanitized<sup>32</sup> werden, bevor die Queries an die Datenbank gesendet werden. Dies passiert allerdings durch die jeweils genutzte offizielle [MariaDB-Connector](#) Programmbibliothek.
- Sowohl die Webanwendung als auch das [API](#) soll mit einem bestehenden Apache2-Webserver im Intranet ausgeliefert werden.

---

<sup>30</sup>Bei einem Commit werden bis dahin noch temporäre Änderungen geschrieben.

<sup>31</sup>Vgl. <https://pypi.org/project/flup/>

<sup>32</sup>Sanitization beschreibt die Validierung des Inputs, bevor es verarbeitet wird. In diesem Fall, bevor die Query gesendet wird.

- Ein Cronjob soll so konfiguriert werden, dass das Programm einmal wöchentlich alle Dateien einliest und die Datenbank komplett neu aufbaut.
- Es soll eine [CI](#)-Pipeline im GitLab konfiguriert werden, damit Artefakte der pre-kompilierten Ausführungsdatei verfügbar gemacht werden können. Dies soll das Deployment vereinfachen.
- In der bestehenden MariaDB-Instanz muss ein Schema nach dem Tabellenschema in der [Abbildung 3](#) erstellt werden.

## A.7 Datenmodell Diagramme

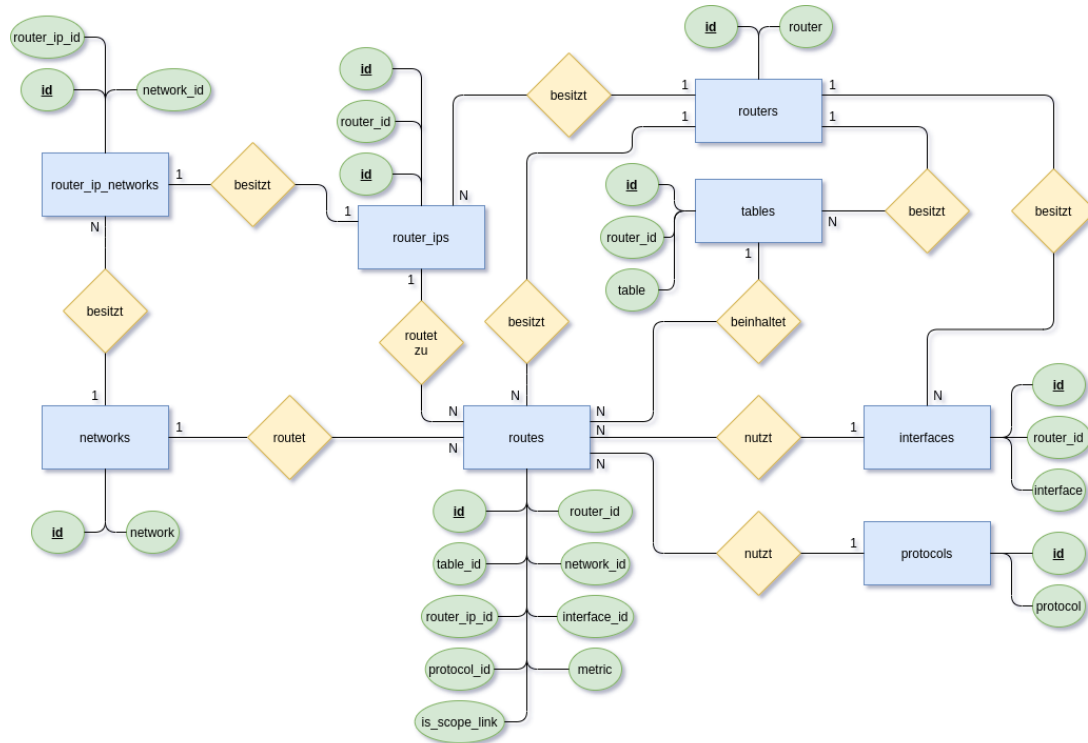


Abbildung 2: Vereinfachtes ER-Modell

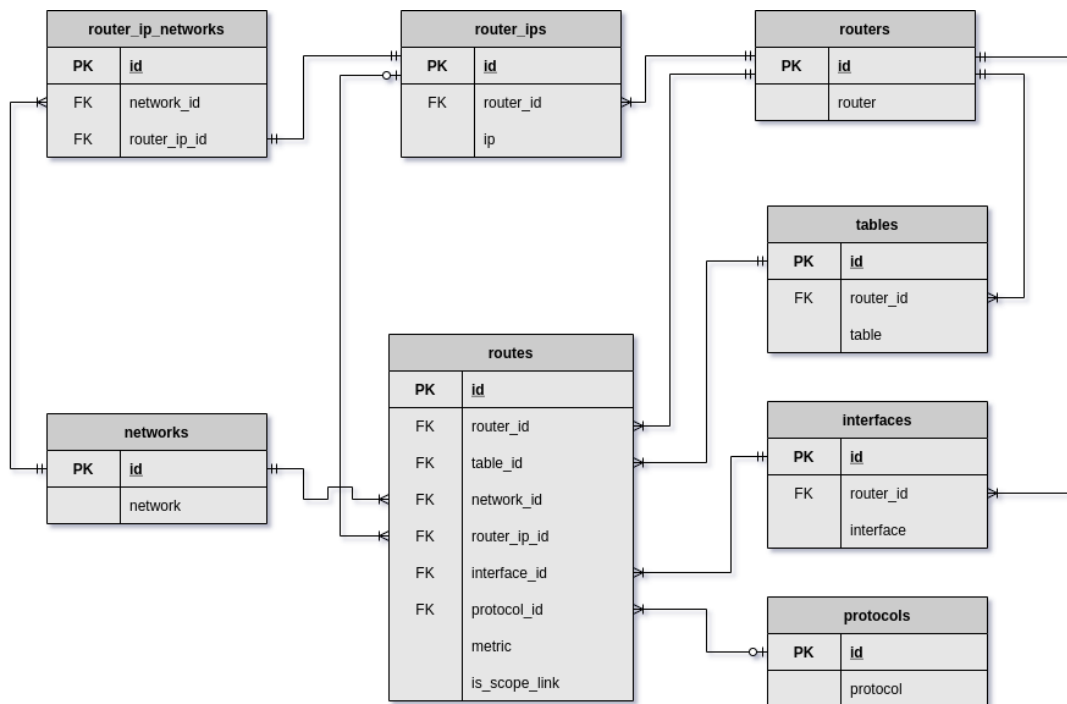


Abbildung 3: Tabellenschema der Datenbank



## A.8 Ereignisgesteuerte Prozesskette

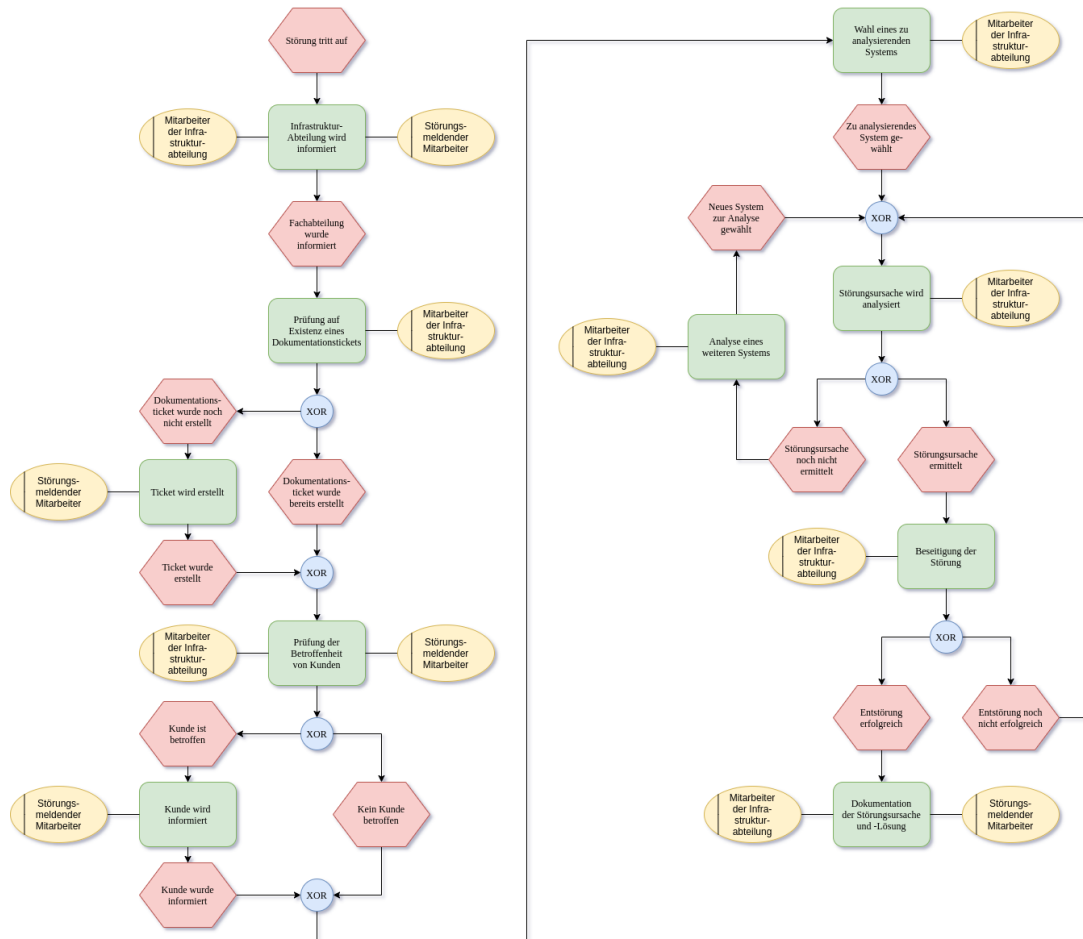


Abbildung 4: EPK des Prozesses der Störungsbearbeitung

## A.9 Sequenzdiagramm

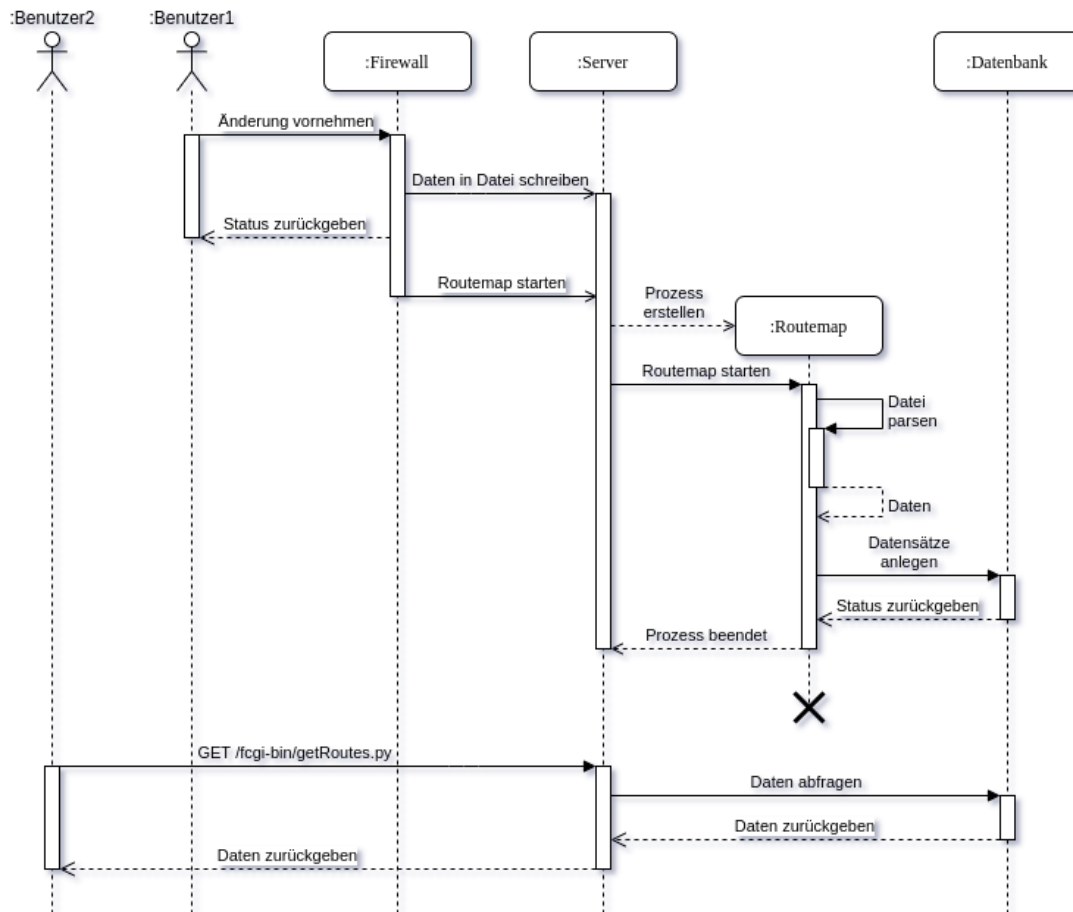


Abbildung 5: Sequenzmodell der Anwendung der verschiedenen Komponenten

## A.10 Oberflächenentwurf

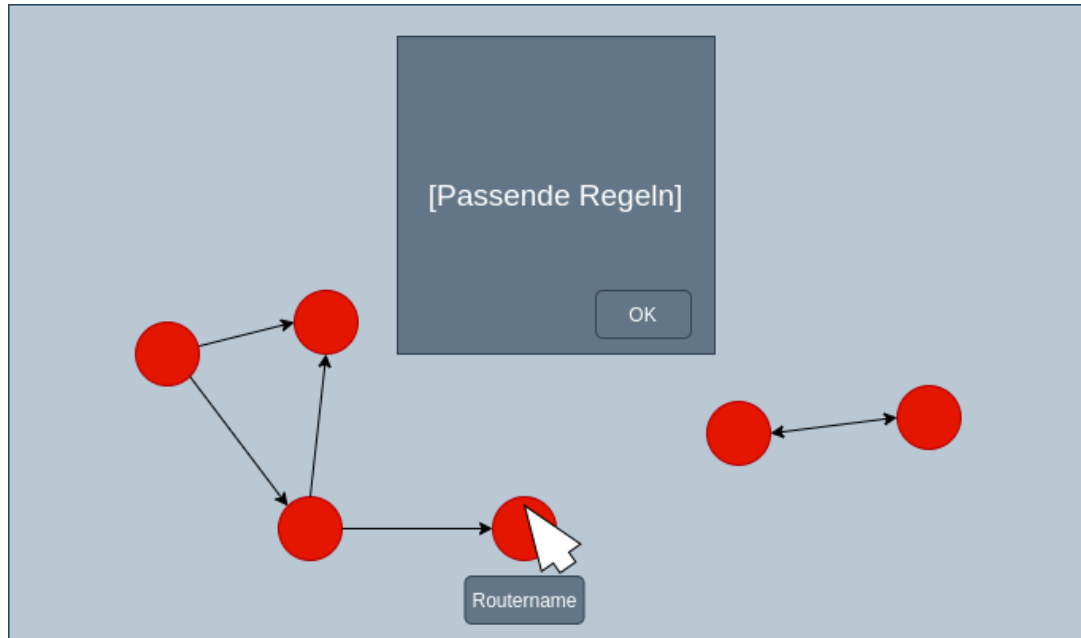


Abbildung 6: Mock-Up der Benutzeroberfläche

## A.11 Screenshot der Verzeichnisstruktur

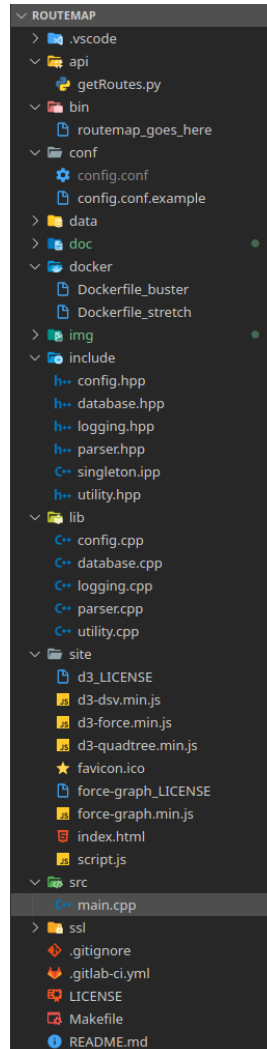


Abbildung 7: Jegliche Verzeichnisse und Dateien des Projektes

Abbildung 8: Virtualhost-Konfiguration des Apache2-Webservers

[illegible]

Henry Knopsmeier

## A.14 Screenshot der CI-Pipeline

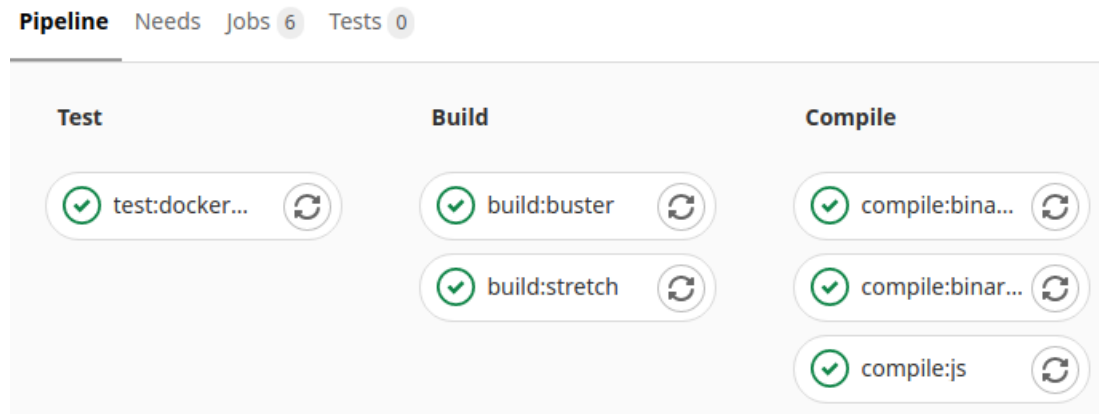


Abbildung 10: Erfolgreich durchgeführte Pipeline

## A.15 Screenshot der Anwenderdokumentation (Ausschnitt)

### Prerequisites

- Mariadb Development Libraries (libmariadbclient-dev, libmariadb-dev)
- gcc/g++ (At least gcc v6.3.0)
- build-tools (pkgconf, build-essential)

### Installation

- Clone this repository
- Compile (see below)
- Copy `conf/config.conf.example` to `conf/config.conf` and fill in the blanks
- Create Cronjob to run the binary every week

### Compilation

A precompiled binary as well as the minified javascript files can be downloaded as a job-artifact from CI/CD.

Alternatively, these artifacts can be downloaded with this command:

```
curl --location --output artifact.zip --header "PRIVATE-TOKEN: [TOKEN]"  
"https://gitlab.boreus.de/api/v4/projects/1703/jobs/artifacts/master/download?job=[JOB]"
```

Where [TOKEN] is your private access token (can be issued from within your GitLab settings) and [JOB] is the name of the job, found under CI/CD of this project (e.g. `?job=compile:js`)

### Automatic

```
make
```

### Manual

- gcc v6.x.x

```
g++ -std=c++17 -D experimental -lmariadb -lstdc++fs src/main.cpp lib/*.cpp -I/usr/ -L/usr/ -o bin/routemap
```

- gcc v8.x.x

```
g++ -std=c++17 -lmariadb -lstdc++fs src/main.cpp lib/*.cpp -I/usr/ -L/usr/ -o bin/routemap
```

- gcc v10.x.x

```
g++ -std=c++17 -lmariadb src/main.cpp lib/*.cpp -I/usr/ -L/usr/ -o bin/routemap
```

### Usage

```
Usage:  
-h | --help           -> Display this help  
-d | --debug          -> Display debug info  
-q | --quiet           -> Do not output anything  
                        /\ Critical errors bypass this option  
-n | --no-execute     -> Do not send any queries to the database  
-f | --firewall [fwXX] -> Only process routing data for specified firewall  
                        /\ Intended to be used by the firewalls themselves
```

Abbildung 11: Installationsanleitung der Anwenderdokumentation



## A.16 Screenshot der API-Dokumentation (Ausschnitt)

### API

#### Endpoints

- getRoutes.py

#### Parameters

Parameter	Optional	Type	Value
user	No	String	Username to authenticate against the database with
pass	No	String	Password to authenticate against the database with
network	No	String	Network-adress to get information of
router	Yes	String	Router to get information of

#### Usage

If the router-parameter is present, the API returns every route of said router matching the given network in plain/text. Else the API returns a complex construct of every router routing said network in application/json.

#### Troubleshooting

- Missing parameter. Expected: user, pass, network. Got: [...]
  - Parameter requirement not met.
  - Parameters need to be passed via the querystring, not by header parameter
- [MariaDB Error]
  - Something went wrong either whilst authenticating or while sending the query.
  - Was the correct database server address configured?
    - if so and the problem persists, please create an issue

Abbildung 12: Ausschnitt der API-Dokumentation

## A.17 Screenshots der Anwendung



Abbildung 13: Anzeige des Graphen sowie der das Netzwerk routenden Regeln

## A.18 Iterationsplan

1. Erstellen des Schemas in der Datenbank
2. Implementierung des ‘Utility-Moduls’
3. Implementierung des ‘Config-Moduls’
4. Implementierung des ‘Logging-Moduls’
5. Implementierung des ‘Datenbank-Moduls’ mit Hilfe des Mariadb-Connectors
6. Implementierung des ‘Parsing-Moduls’
7. Implementierung des Python-[FCGI-API](#) mit Hilfe der flup-Programmbibliothek
8. Implementierung des Frontends inklusive des Zeichnens des Graphens

## A.19 Listings

### A.19.1 Listing des Datenbank-Moduls

```
1 #include "../include/database.hpp"
2 #include "../include/logging.hpp"
3 #include "../include/singleton.hpp"
4
5 #define uLogger makeUnique<Logger>::Instance()
6
7
8 void Database::connect(const char *host, unsigned int port, const char *user, const
    char *pass, const char *dbName, const char *caPath){
9
10     // create session-object
11     this->connection = mysql_init(nullptr);
12     // use SSL-CA-Certificate to authenticate database before creating a session
13     mysql_options(this->connection, MYSQL_OPT_SSL_CA, caPath);
14     // connect to database specified in config-file
15     if (!mysql_real_connect(this->connection, host, user, pass, dbName, port, nullptr,
        0)){
16         uLogger.log("Could not connect to " + (std::string)host + "! Error: " +
            mysql_error(this->connection), "Database::connect()", "CRITICAL");
17     } else {
18         uLogger.log("Connected to " + (std::string)host + "!", "", "INFO");
19     }
20     // turn off autocommits for committing changes atomically before closing the
        connection
21     mysql_autocommit(this->connection, (my_bool>false);
22
23 }
24
25
26 void Database::query(const char *query){
27
28     this->queryCounter++;
29
30     // mysql_query returns true if an error occurred
31     if (mysql_query(this->connection, query)){
32         uLogger.log("Could not query \"" + (std::string)query + "\"! Error: " +
            mysql_error(this->connection), "Database::query()", "ERROR");
33     } else {
34         this->successfulQueryCounter++;
35     }
36
37 }
38
39
40 void Database::query(table &queryResponse, const char *query){
```

```
41 Database::query(query);
42
43
44 MYSQL_RES *response = mysql_store_result(this->connection);
45 MYSQL_ROW row;
46
47 unsigned int numFields = mysql_num_fields(response);
48
49 // for every row in the response
50 while ((row = mysql_fetch_row(response))) {
51
52     std::vector<std::string> queryColumn;
53     // for every column in the row
54     for (int i = 0; i < numFields; i++) {
55
56         // value can not contain Null, thus "NULL" is inserted instead as a string
57         queryColumn.push_back(row[i] ? row[i] : "NULL");
58
59     }
60     queryResponse.push_back(queryColumn);
61
62 }
63 // flush result buffer
64 mysql_free_result(response);
65
66 }
67
68
69 void Database::disconnect(){
70
71     // commit transaction and end session
72     mysql_commit(this->connection);
73     mysql_close(this->connection);
74     uLogger.log("Sent " + std::to_string(this->queryCounter) + " queries, of which " +
75         std::to_string(this->successfulQueryCounter) + " were successful!", "", "INFO");
76     uLogger.log("Disconnected from database!", "", "DEBUG");
77
78 }
```

Listing 1: Auszug des Datenbank-Modul-Quelltextes

### A.19.2 Listing der CTE

```
1 WITH RECURSIVE paths (cur_path, cur_dest)
2 AS (
3     SELECT router, router
4     FROM v_routes
5     WHERE subnet_in_supernet(v_routes.network, ?)
```

```

6      AND v_routes.table != '220'
7      AND v_routes.is_scope_link = 1
8      UNION
9      SELECT CONCAT(paths.cur_path, ',', v_routes.router), v_routes.router
10     FROM paths
11     JOIN v_selfips on paths.cur_dest = v_selfips.router
12     JOIN v_routes on v_selfips.ip = v_routes.ip
13     WHERE subnet_in_supernet(v_routes.network, ?)
14     AND NOT FIND_IN_SET(v_routes.router, paths.cur_path)
15 )
16 SELECT * FROM paths WHERE cur_path != cur_dest;

```

Listing 2: SQL-Befehl zur Definition der CTE

Hinweis: Die WHERE-Klausel beinhaltet den Ausschluss der table 220, da dies einen Sonderfall abdeckt, der nicht durch dieses Projekt dargestellt werden soll. Die Daten befinden sich jedoch weiterhin in der Datenbank. Desweiteren ist das ? ein Platzhalter für die jeweils angegebene Netzwerkadresse, welche vor dem Einfügen in die Query sanitized wird.

### A.19.3 Listing der SQL-Funktion

```

1 CREATE FUNCTION subnet_in_supernet (supernet varchar(255) charset utf8, subnet varchar
  (255) charset utf8)
2 RETURNS boolean deterministic
3 RETURN
4 substring_index(supernet, "/", -1) <= substring_index(subnet, "/", -1)
5 AND
6 inet_aton(substring_index(supernet, "/", 1)) & ~((1 << (32-substring_index(supernet
  , "/", -1))) - 1)
7 =
8 inet_aton(substring_index(subnet, "/", 1)) & ~((1 << (32-substring_index(supernet,
  "/", -1))) - 1);

```

Listing 3: SQL-Befehl zur Definition der benötigten Funktion

## A.20 Soll-/Ist-Vergleich

Projektphase	Soll	Ist	Differenz
Planungs- und Analysephase	5 h	6 h	+1 h
Entwurfsphase	10 h	10 h	
Implementierungsphase	30 h	28 h	-2 h
Abnahmetest der Fachabteilung und Deployment	5 h	4 h	-1 h
Erstellen der Dokumentation	20 h	22 h	+2 h
<b>Gesamt</b>	<b>70 h</b>	<b>70 h</b>	