



# topology designer

## Projektdokumentation

### Studienarbeit HS 2019

Version: 1.0

Autor:	Martin Hug Vanessa Janknecht	Betreuer:	Laurent Metzger Sebastian Hug
E-Mail:	<a href="mailto:martin.hug@hsr.ch">martin.hug@hsr.ch</a> <a href="mailto:vanessa.janknecht@hsr.ch">vanessa.janknecht@hsr.ch</a>	Themengebiet:	Webentwicklung
		Studiengang:	Informatik

Erstellt am: 19. September 2019

Letzte Änderung am: 18.12.2019

## I Abstract

Das Institut für Vernetzte Systeme betreibt im Rahmen verschiedener HSR Module Labs, in welchen virtuelle Netzwerke bzw. Geräte konfiguriert werden sollen. Zur Provisionierung und Verteilung der Topologien wird eine vorhergehende Studienarbeit (Lab Topology Builder) verwendet, welche die virtuelle Umgebung anhand manuell generierter YAML File aufsetzt. Die manuelle Generierung der YAML Files ist sehr umständlich und fehleranfällig.

Ziel der Arbeit ist eine Webanwendung, welche es ermöglicht, die Topologie am Computer zu zeichnen und daraus automatisch ein YAML generieren zu lassen. Die Webanwendung soll auf den gängigsten Web-Browsern betrieben werden können.

Es wurde eine Webanwendung mit REACT Frontend und DJANGO Backend mit einer SQLite Datenbank entwickelt. Die graphische Darstellung wurde mit visJS umgesetzt. Mit der Anwendung ist es möglich vom "Lab Topology Builder" vorgegebene Gerätetypen in einer graphisch dargestellten Topologie zu erstellen, miteinander zu verbinden und konfigurieren. Die erstellte Topologie kann schlussendlich mit sämtlichen Konfigurationen in ein YAML File exportiert werden.

## Inhalt

I	Abstract.....	2
1	<b>Aufgabenstellung.....</b>	<b>5</b>
2	<b>Management Summary.....</b>	<b>6</b>
2.1	Ausgangslage .....	6
2.2	Resultate.....	6
2.3	Technologien .....	7
2.3.1	Frontend.....	7
2.3.2	Backend.....	7
2.4	Datenbank.....	8
3	<b> Projektdokumentation .....</b>	<b>9</b>
3.1	Anforderungsspezifikation.....	9
3.1.1	Use Cases .....	9
3.1.2	Weitere Funktionen .....	17
3.1.3	Nicht-funktionale Anforderungen; Rahmenbedingungen .....	17
3.2	Analyse.....	19
3.2.1	Objektkatalog .....	19
3.2.2	Domain-Modell, Klassendiagramm (konzeptionell) .....	20
3.3	Design Entwurf .....	21
3.3.1	Wireframes .....	21
3.3.2	Sequenzdiagramm.....	30
3.4	Implementation (Entwicklung) und Tests .....	31
3.4.1	Infrastruktur.....	31
3.4.2	Implementation: Erläuterungen wichtiger konkreter Klassen.....	33
3.4.3	Deployment Diagramm.....	35
3.4.4	Automatische Testverfahren.....	36
3.4.5	Usability Testing .....	36
3.5	Resultate und Weiterentwicklung.....	37
3.5.1	Resultate.....	37
3.5.2	Möglichkeiten der Weiterentwicklung .....	38
3.6	Projektmanagement – Planung/Soll .....	39
3.6.1	Prozessmodell .....	39
3.6.2	Team, Rollen und Verantwortlichkeiten .....	40
3.6.3	Aufwandschätzung.....	40
3.6.4	Zeitplan.....	40
3.6.5	Iterationen.....	41
3.6.6	Meilensteine .....	42
3.6.7	Risikomanagement.....	43
3.6.8	Risikomatrix.....	45
3.7	Qualitätssicherung.....	46
3.8	Projektmonitoring - Ist.....	47
3.8.1	Soll-Ist-Zeitvergleich .....	47

3.8.2	Codestatistik.....	49
<b>4</b>	<b>Setupbeschreibung.....</b>	<b>51</b>
4.1	Installation.....	51
4.1.1	Pull der aktuellen Docker Container.....	51
4.1.2	Container Starten .....	51
4.2	Benutzerhandbuch .....	52
4.2.1	Werkzeugleiste.....	52
4.2.2	Import YAML .....	52
4.2.3	Add Edge .....	53
4.2.4	Network Device Konfigurieren .....	53
4.2.5	Restliche Devices Konfigurieren.....	54
4.2.6	Topologie als Bild exportieren.....	54
<b>5</b>	<b>Persönlicher Rückblick.....</b>	<b>55</b>
5.1	Martin Hug.....	55
5.2	Vanessa Janknecht .....	55
<b>6</b>	<b>Literatur und Quellenverzeichnis.....</b>	<b>57</b>
<b>7</b>	<b>Verzeichnisse.....</b>	<b>58</b>
7.1	Glossar und Abkürzungsverzeichnis.....	58
7.2	Abbildungen .....	58
7.3	Tabellen .....	59
<b>8</b>	<b>Anhang.....</b>	<b>60</b>
8.1	Redmine Auszug.....	60
8.1.1	Inception .....	60
8.1.2	Elaboration.....	60
8.1.3	Construction.....	61
8.1.4	Transition.....	61
8.2	Usability Test .....	62
8.2.1	Testkonzept.....	62
8.2.2	Testprotokoll .....	64

# 1 Aufgabenstellung

Semesterarbeit HS 2019

Webanwendung zum Erstellen und Bearbeiten von Netzwerktopologien für den Lab  
Topology Builder

Studierende: Martin Hug, Vanessa Janknecht

Betreuer: Laurent Metzger, Sebastian Hug

## 1 Ausgangslage

In einer vorangehenden Semesterarbeit wurde der Lab Topology Builder (LTB) gebaut. Mit ihm können YAML Konfigurationsdateien für Labs erstellt werden. Ein Lab kann dabei unterschiedliche Geräte enthalten. Dazu gehören momentan Netzwerkgeräte, virtuelle Maschinen und Docker Container. Im YAML werden die Geräte benannt, miteinander verbunden und konfiguriert.

Die Erstellung eines solchen YAML Dateien ist sehr mühsam. Zum einen bietet die Syntax viel Fehlerpotential und zum anderen müssen gerade für Verbindungen zwischen den Geräten die Topologien aufgezeichnet werden.

## 2 Aufgabenstellung

Ziel der Arbeit ist eine Webanwendung, welche es ermöglicht, die Topologie am Computer zu zeichnen und daraus automatisch ein YAML erstellen zu lassen.

Folgende Aufgaben sind umzusetzen:

1. Eine Topologie kann mittels Drag and Drop erstellt werden und danach als YAML exportiert werden.
2. Die Devicetypen (Virtual Network Device, Docker Container, etc.) können in einem Backend definiert werden.
3. Die Topologie ist als Bild exportierbar.
4. Vorhandene YAML Dateien können importiert und danach bearbeitet werden.
5. Die Anwendung läuft in allen aktuellen Browsern. Mobile Support ist nicht notwendig.
6. Die Entwicklung verläuft nach den Prinzipien moderner Softwareentwicklung und wird entsprechend dokumentiert.

## 3 Bewertung

Das Ergebnis der Arbeit und der Bericht werden bewertet.

Laurent Metzger



## 2 Management Summary

### 2.1 Ausgangslage

**Problemstellung,** Vi-Das Institut für vernetzte Systeme betreibt im Rahmen einzelner HSR Module verschiedene Labs, bei welchen virtuelle Netzwerk und Serversysteme zur Verfügung gestellt werden. Diese Systeme werden mit einem Tool (Lab Topologie Builder) provisioniert. Die Informationen bzw. Konfigurationen der Systeme werden mit Hilfe eines YAML Files in die LTB Umgebung eingegeben. Um die Vorbereitung bzw. Bereitstellung der Labs zu vereinfachen soll eine Web-Anwendung konzipiert und entwickelt werden, um die Topologien graphisch zu erstellen und zu konfigurieren. Sämtliche Konfigurationen sollen in ein YAML File exportierbar sein.

**Ziele**

- Die Anwendung soll auf den gängigen Browsern bedienbar sein
- Die gezeichnete Topologie ist als YAML File und als Bild zu exportieren

**Rahmenbedingungen** Die Applikation wird im Netz des INS betrieben. Lediglich Mitarbeiter des INS erhalten Zugriff auf die Anwendung.

**Methode / Vorgehen** Das Projekt wird mit Hilfe, der in SE1 und SE2 erlernten Projekt Methoden realisiert.

### 2.2 Resultate

**Zusammenfassung** Die Applikation konnte plangemäss entwickelt werden. Es wurde ein Frontend mit React und ein Django basiertes Backend realisiert. Die graphische Darstellung der Topologie wurde mit visJS umgesetzt. Als Datenbank wird eine SQLite Datenbank verwendet. Die Datenbank kann mit einer Administrationskonsole verwaltet werden. Die gesamte Umgebung wird auf zwei miteinander kommunizierende Docker Container betrieben.

**Zielerreichung** Sämtliche in der Aufgabenstellung vorgegebenen Ziele konnten mit einer Ausnahme komplett umgesetzt werden. Das Exportieren der Topologie als Bild konnte aufgrund eines crossorigin Problems leider nicht vollständig realisiert werden und wurde stattdessen mit einem Workaround realisiert. (mehr dazu im Kapitel 3.6).

**Ausblick** Bei dem aktuellen Entwicklungsstand werden die Koordination der einzelnen Nodes bzw. Devices nicht gespeichert. Dies führt dazu, dass bei sämtlichen Modifizierungen die Topologie neu gezeichnet wird und sich dadurch die Anordnung ändert. Zudem könnte die Docker Architektur bzw. die Kommunikation zwischen den Dockern so angepasst werden, dass die crossorigin Probleme obsolet werden.

## 2.3 Technologien

### 2.3.1 Frontend

**react.js** Im Frontend verwenden wir React<sup>1</sup> und JavaScript. Dies war eine Anforderung unseres Betreuers, da React im INS verwendet wird. Somit ist der Support der Applikation nach der Studienarbeit gewährleistet.  
**react-graph-vis.js**  
**axios.js**  
**yaml.js**

Zudem benutzen wir für die graphische Darstellung die Bibliothek react-graph-vis.js<sup>2</sup>. Zur Kommunikation mit der API vom Backend wird axios<sup>3</sup> verwendet. Um ei

Für die Importieren bestehender YAML Files wurde yaml.js verwendet, um dies in JSON Objekte zu konvertieren.

### 2.3.2 Backend

**Django** Das Backend sollte in Python umgesetzt werden. Dies war eine Anforderung unseres Betreuers, da im INS das Knowhow vorhanden ist und somit die Wartung und Weiterentwicklung sichergestellt ist. In Frage folgende Frameworks:

**Django REST Framework**  
**django-cors-headers**

Django<sup>4</sup>

- « all-inclusive» → grosses Framework
- Admin Panel, Datenbank Interface, ORM (Object relational Mapping) und Verzeichnis-aufbau für die Anwendung direkt «out-of-the-box»
- Gut für straight-forward Applikationen
- Built in Security gegen CSRF, XSS, SQL injection, etc.

Flask<sup>5</sup>

- Einfachheit
- Flexibilität
- feinkörnige Kontrolle.
- Es lässt einen selbst entscheiden wie etwas implementiert wird.
- Mehr Kontrolle über die Komponenten, die genutzt werden.
- Absolute Kontrolle, wie man die Anwendung gestalten möchte

Wir haben uns für Django entschieden, da unser Backend ziemlich klein straight-forward ist. Wir denken nicht, dass wir die starke Flexibilität von Flask benötigen. Da der Grosse Teil der Arbeit das Frontend ausmacht, wollen wir auch möglichst viel Zeit darauf verwenden.

---

<sup>1</sup> <https://reactjs.org/>

<sup>2</sup> <https://github.com/crubier/react-graph-vis>

<sup>3</sup> <https://github.com/axios/axios>

<sup>4</sup> <https://www.djangoproject.com/>

<sup>5</sup> <https://www.palletsprojects.com/p/flask/>

Bei Flask müssten wir noch einiges an Zeit aufwenden, um geeignete Bibliotheken zu finden. Daher kommt uns der «all-inclusive» Service von Django gerade recht. Zur Erstellung der Web API verwenden wir das Django REST Framework<sup>6</sup>. Für die Kommunikation mit dem Frontend wird `django-cors-headers`<sup>7</sup> verwendet.

## 2.4 Datenbank

### SQLite

Als Datenbank verwenden wir SQLite. Diese Datenbank wird direkt von Django erstellt. Wir haben uns entschieden, uns auf eine SQLite Datenbank zu beschränken. In der Datenbank werden lediglich die Device Informationen abgelegt. Darin enthalten sind:

- **Name** Name des Gerätetyps (vorgegeben vom LTB)
- **Type** Typ des Gerätes (vorgegeben vom LTB)
- **Default Name** Name welcher im Frontend bei der Erstellung angezeigt wird
- **Image** PNG Bild, welches im Frontend angezeigt wird

Der Datenbankzugriff wird nur einmal beim Initialen Aufruf des Frontend gemacht. Anschliessend werden die Informationen im Frontend zwischengespeichert.

Selbstverständlich wäre es möglich das Django Backend mit einer PostgreSQL Datenbank zu Verknüpfen. Da in unserem Falle die benötigte Logik auf der Datenbank sehr gering ist reicht die direkt mitgelieferte SQLite Datenbank völlig aus.

---

<sup>6</sup> <https://www.django-rest-framework.org/>

<sup>7</sup> <https://github.com/adamchainz/django-cors-headers>

## 3 Projektdokumentation

### 3.1 Anforderungsspezifikation

#### 3.1.1 Use Cases

Der Topology Designer wird ausschliesslich im Netz des INS verwendet. Auf die Benutzeroberfläche können somit nur jene User zugreifen, die sich direkt oder via VPN im INS Netz befinden. Zudem benötigt kein Anwender der Applikation Zugriff auf das Backend, da sämtliche Topologie bezogenen Daten unabhängig in einem YAML-File abgespeichert werden. Deshalb wird im Use Case Diagramm ein einzelner Aktor dargestellt.

##### 3.1.1.1 Use Case Diagramm



Abbildung 2 Use Case Diagramm

### 3.1.1.2 Use Cases Brief

<b>UC01</b> Import Topology	Ein User kann ein bestehendes yaml-File in das Programm einlesen um sie anschliessend zu bearbeiten
<b>UC02</b> Reorganize Topology	Ein User kann die angezeigte Topologie automatisch sortieren lassen.
<b>UC03</b> Clear Topology	Für den Fall, dass der User nach dem Exportieren eine neue Topologie erstellen will, gibt es einen Button «Clear Topology» welcher sämtliche Devices und Connections löscht und so die Topologie leert.
<b>UC04</b> CRUD Device	Ein User kann aus einer Liste von Komponenten eine auswählen und via Drag'n'Drop bewegen.
<b>UC04a</b> Modify Interface Configuration	Ein User kann ein Netzwerk Interface konfigurieren. (Name, Ip Adresse, Netzmaske)
<b>UC04b</b> Modify Run Configuration	Ein User kann die Run-Konfiguration einer Netzwerk Komponente verändern. (hinzufügen von Loopback Interfaces, OSPF, BGP usw.)
<b>UC05</b> CRUD Connections	Ein User kann zwischen zwei Netzwerk Komponenten eine Verbindung herstellen.
<b>UC05a</b> CRUD Connection Name	Ein User kann eine Verbindung benennen.
<b>UC05b</b> CRUD Interface Description	User kann einem Interface eine Beschreibung hinzufügen
<b>UC06</b> Export Topology as YAML	Ein User kann die Topologie als YAML File exportieren um anschliessend damit die Topologie auf dem Lab Topology Builder zu deployen.
<b>UC07</b> Export Topology as Picture	Ein User kann die erstellte Topologie als Bild oder PDF exportieren
<b>UC08</b> CRUD Devicetype	Ein User kann neue Gerätetypen definieren und erstellen. Z.B. spezifische Router Typen, Switches usw.
<b>UC09</b> Restore Topology from Cache	Die Topologie soll im Browsercache gespeichert werden. Im Falle eines Netzwerkunterbruchs bzw. wenn die Seite neu geladen wird, soll die aktuell erstellte Topologie aus dem Browsercache wiederhergestellt werden.

Tabelle 1 Use Cases Brief

### 3.1.1.3 Use Case Casual

UC01 Import Topology	Wenn der User auf den Button «Import Topology» klickt erscheint ein Explorer Fenster. Dort kann der User den Pfad zu seinem vorhandenen yaml-File angeben. Wenn der User auf «Ok» klickt erscheinen die Daten des yaml-Files im topology designer und können verändert werden. Wenn der User auf «Cancel» klickt schliesst sich das Explorer Fenster und nichts passiert.
UC02 Reorganize Topology	Durch einen Klick auf den Button «Reorganize» wird die vorhandene Topologie neu angeordnet. Wenn die Topologie bereits gut organisiert ist geschieht nichts.
UC03 Clear Topology	Wenn der User auf den Button «Clear Topology» klickt erscheint ein Fenster, welches ihn fragt, ob er diese Aktion wirklich ausführen will. Es weist den User darauf hin, dass nach dem Löschen keine Daten wiederhergestellt werden. Klickt der User auf «Ok» schliesst sich das Fenster und die Topologie wird unwiderruflich gelöscht. Klickt der User auf «Cancel» schliesst sich das Fenster und es geschieht nichts.
UC07 Export Topology as Picture	Wenn der User auf den Button «Export as Picture» klickt, öffnet sich ein Fenster, in dem man den Pfad für das zu exportierende Bild angeben kann. Durch einen Klick auf «Ok» wird die Topologie am angegebenen Ort gespeichert. Ist kein Pfad angegeben erscheint eine Warnung, dass kein Pfad angegeben wurde. Durch einen Klick auf «Cancel» schliesst sich das Fenster und nichts geschieht.
UC08 CRUD Devicetype	Es soll möglich sein neue Gerätetypen zu erfassen sowie die bestehenden zu Bearbeiten bzw. zu Löschen
UC09 Restore Topology from Cache	Die Topologie soll im Cache zwischengespeichert werden. Wenn der User die Seite aktualisiert (durch einen Klick auf «Diese Seite neu laden» oder F5) soll der aktuelle Stand der Topologie nicht verloren gehen.

Tabelle 2 Use Cases Casual

### 3.1.1.4 Use Case Fully Dressed

#### 3.1.1.4.1 UC04 CRUD Device

<b>Use Case ID</b>	UC04
<b>Use Case Name</b>	CRUD Device
<b>Created By</b>	Martin Hug
<b>Date Created</b>	03.10.2019
<b>Actors</b>	User
<b>Description</b>	Ein User zieht per Drag'n Drop ein Gerätetyp der vorgegebenen Geräteleiste in die Topologie hinein und erstellt es somit. Anschliessend kann er jederzeit die Geräteeinstellungen ändern (Siehe UC04a bzw. UC04b). Ebenfalls kann das Gerät jederzeit gelöscht werden.
<b>Preconditions</b>	<ol style="list-style-type: none"> <li>1. Die Applikation hat die Geräteleiste initialisiert.</li> </ol>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. Der Topology Designer zeigt das erstellte Gerät bzw. die Änderung innerhalb der Topologie an.</li> </ol>
<b>Normal Flow</b>	<p>Create</p> <ol style="list-style-type: none"> <li>1. User klickt auf das gewünschte Gerätesymbol</li> <li>2. User zieht mit geklickter linker Maustaste das Gert in den Topologie-Bereich.</li> <li>3. User lässt die Maustaste los</li> <li>4. Topology Designer zeigt das erstellte Gerät an.</li> </ol> <p>Read</p> <ol style="list-style-type: none"> <li>1. User klickt auf das Gerät</li> <li>2. Informationen werden angezeigt</li> </ol> <p>Update</p> <ol style="list-style-type: none"> <li>1. User klickt auf das Gerät</li> <li>2. Informationen werden angezeigt und können bearbeitet werden.</li> </ol> <p>Delete</p> <ol style="list-style-type: none"> <li>1. User klickt auf Gerät</li> <li>2. User klickt auf «Gerät Löschen»</li> <li>3. Gerät wird gelöscht und verschwindet von der Anzeige.</li> </ol>
<b>Alternative Flow</b>	User klickt auf dem Keyboard auf «delete», wenn er ein Gerät angewählt hat, um es zu löschen. (optional)
<b>Exceptions</b>	<ol style="list-style-type: none"> <li>1. Gerät besitzt Connections bei Löschvorgang</li> </ol>

	Topology Designer informiert den User, dass Verbindungen zu anderen Geräten bestehen. Der User wird gefragt ob er das Gerät trotzdem löschen will. Falls ja, werden die Verbindungen ebenfalls gelöscht. (optional)
Priority	Hoch
Frequency of Use	Hoch
Special Requirements	1. Änderungen an der Topologie sind rein optisch und nur für die Anzeige geltend. Es werden keine Gerätekonfigurationen persistent gespeichert.
Assumptions	Topologie ist erstellt oder geladen.
Notes and Issues	1. User verliert die Verbindung beim Bearbeiten Daten werden im Browsercache gespeichert und können wiederhergestellt werden.

Tabelle 3 Use Case Fully Dressed UC04

### 3.1.1.4.2 UC05 CRUD Connections

<b>Use Case ID</b>	UC05
<b>Use Case Name</b>	CRUD Device
<b>Created By</b>	Martin Hug
<b>Date Created</b>	03.10.2019
<b>Actors</b>	User
<b>Description</b>	Ein User klickt auf das Verbindungselement in der Werkzeugleiste und verbindet 2 Geräte in dem er ein Gerät anklickt und mit gedrückter Maustaste über ein anderes Gerät fährt und anschliessend die Maustaste loslässt. Anschliessend zeigt der Topology Designer die «gezeichnete» Verbindung. Um die Verbindung zu löschen muss sie der User auswählen und via Delete Symbol löschen
<b>Preconditions</b>	<ol style="list-style-type: none"> <li>1. Es wurden minimal 2 Geräte erstellt</li> </ol>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. Der Topology Designer zeigt die erstellte Verbindung bzw. die Änderung innerhalb der Topologie an.</li> </ol>
<b>Normal Flow</b>	<p>Create</p> <ol style="list-style-type: none"> <li>1. User klickt auf das gewünschte Verbindungssymbol</li> <li>2. User zieht mit gedrückter linker Maustaste eine Linie zwischen 2 bestehenden Geräten.</li> <li>3. User lässt die Maustaste los</li> <li>4. Topology Designer zeigt die erstellte Verbindung an.</li> </ol> <p>Read</p> <p>Nicht nötig, da keine Informationen in der Verbindung gespeichert sind. Welche Geräte die Verbindung verbindet ist anhand der Linie erkennbar.</p> <p>Update</p> <ol style="list-style-type: none"> <li>1. Nicht nötig, da eine Verbindung nur explizit zwischen 2 Geräten besteht. Falls eine Verbindung zwischen 2 anderen Geräten erstellt werden soll, muss eine neue Verbindung erstellt werden.</li> </ol> <p>Delete</p> <ol style="list-style-type: none"> <li>1. User klickt auf die Verbindung</li> <li>2. User klickt auf das Löschesymbol</li> <li>3. Verbindung wird gelöscht und nicht mehr angezeigt</li> </ol>
<b>Alternative Flow</b>	User klickt auf dem Keyboard auf «delete», wenn er die Verbindung angewählt hat, um sie zu löschen.
<b>Exceptions</b>	<ol style="list-style-type: none"> <li>1. Nur 1 Gerät erstellt</li> </ol> <p>Falls nur ein Gerät zur Topologie hinzugefügt wurde, kann der Topology Designer keine Verbindung erstellen. Beim Klick auf das Verbindungssymbol erscheint eine Fehlermeldung</p>

<b>Priority</b>	Hoch
<b>Frequency of Use</b>	Hoch
<b>Special Requirements</b>	1. Änderungen an der Topologie sind rein optisch und nur für die Anzeige geltend. Es werden keine Verbindung persistent gespeichert.
<b>Assumptions</b>	Topologie ist erstellt oder geladen. Minimum 2 Geräte sind erstellt
<b>Notes and Issues</b>	1. User verliert die Verbindung beim Bearbeiten Daten werden im Browsercache gespeichert und können wiederhergestellt werden.

Tabelle 4 Use Case Fully Dressed UC05

### 3.1.1.4.3 UC 06 Export Topology

<b>Use Case ID</b>	UC06
<b>Use Case Name</b>	Export Topology as YAML
<b>Created By</b>	Martin Hug
<b>Date Created</b>	03.10.2019
<b>Actors</b>	User
<b>Description</b>	Ein User klickt auf das «Export as YAML» Symbol, um die Topologie zu exportieren. Es muss ein Speicherort angegeben werden. Dabei werden sämtliche Konfigurationen ausgelesen und in eine YAML File geschrieben.
<b>Preconditions</b>	<ol style="list-style-type: none"> <li>1. Es wurde eine Topologie erstellt</li> <li>2. Es wurde minimal ein Gerät erstellt</li> </ol>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. Das YAML File wurde heruntergeladen und am angegebenen Ort abgespeichert.</li> </ol>
<b>Normal Flow</b>	<ol style="list-style-type: none"> <li>1. User klickt auf «Exportieren als YAML».</li> <li>2. Exportdialog öffnet sich</li> <li>3. Speicherort wird ausgewählt</li> <li>4. Aktion wird bestätigt</li> <li>5. YAML File wird im Speicherort abgelegt.</li> </ol>
<b>Alternative Flow</b>	
<b>Exceptions</b>	<ol style="list-style-type: none"> <li>1. Kein Gerät erstellt Falls noch kein Gerät zur Topologie hinzugefügt wurde, kann der Topology Designer nichts exportieren. Beim Klick auf das Exportsymbol erscheint eine Fehlermeldung</li> </ol>
<b>Priority</b>	Hoch
<b>Frequency of Use</b>	Hoch
<b>Special Requirements</b>	<ol style="list-style-type: none"> <li>1. Die eingegebenen Konfigurationen werden beim Exportvorgang nicht auf Korrektheit geprüft. Falls beispielsweise bei der Run-Config eines Routers ein Fehler konfiguriert wurde wird dies beim Export nicht beachtet.</li> </ol>
<b>Assumptions</b>	Topologie ist erstellt oder geladen. Minimum 1 Gerät ist erstellt
<b>Notes and Issues</b>	<ol style="list-style-type: none"> <li>1. User verliert die Verbindung beim Bearbeiten</li> </ol>

	Daten werden im Browsercache gespeichert und können wiederhergestellt werden. 2. Fehlkonfigurationen werden nicht geprüft
--	--

Tabelle 5 Use Case Fully Dressed UC06

### 3.1.2 Weitere Funktionen

**Caching** Als optionaler Use Case wurde definiert, dass die Topologie beim Aktualisieren der Seite aus dem Browsercache geladen werden soll. Bei der Bearbeitung der Topologie kann es durchaus passieren, dass es zu einem kurzen Verbindungsunterbruch kommen kann und dadurch die Seite neu geladen werden muss. Um dabei den aktuellen Fortschritt der Topologie nicht zu verlieren ist die Speicherung im Cache sehr hilfreich.

### 3.1.3 Nicht-funktionale Anforderungen; Rahmenbedingungen

**Funktionalität** *Angemessenheit:*  
Die Endbenutzer des Topology Designers sollen auf keine Hindernisse bei der Bedienung der Software stossen. Das GUI wird möglichst intuitiv gestaltet und mit Hilfe von Usability Tests auf Klarheit geprüft.

*Richtigkeit:*

Die im Tool konstruierte Topologie soll vollständig und mit sämtlichen Konfigurationen und Einstellungen in ein YAML File exportiert werden. Die tatsächliche Korrektheit der eingegebenen Konfiguration wird von der Software nicht geprüft. Dies soll heissen, dass nicht gewährleistet ist, dass ein aus der Software exportiertes YAML File direkt im Lab Topology Builder<sup>8</sup> deployed werden kann.

*Interoperabilität:*

Durch die Möglichkeit YAML Files aus der Applikation zu exportieren wird die Interoperabilität zu der Bestehenden Lösung «Lab Topology Builder» gewährleistet.

*Sicherheit:*

Die Applikation wird innerhalb des INS Netzwerks deployed und ist somit auch einschliesslich für jene zugänglich, die bereits via VPN oder direkt am Netz authentisiert sind. Zudem sind ausgenommen von den angebotenen Gerätetypen sämtliche Daten nicht persistent gespeichert.

Aufgrund dessen, wird für die Applikation kein zusätzlicher Login Prozess implementiert.

**Zuverlässigkeit** *Fehlertoleranz:*  
Falsche Eingaben führen nicht zum Absturz des Programms. Fehlerhafte oder unvollständige Eingaben werden geprüft und dem Benutzer anhand einer Fehlermeldung bekanntgegeben. Fehleingaben bei den gerätespezifischen Run-Configs (z.B. BGP

<sup>8</sup> Semesterarbeit für das INS Siehe <https://eprints.hsr.ch/734/>

Neighbors usw.) können nicht geprüft werden. Die fehlerhafte Eingabe wird nicht erkannt, sondern beim Export in ein YAML File fehlerhaft übernommen.

## Benutzbarkeit

### *Verständlichkeit:*

Die Oberfläche des Topology Designers soll für den Endbenutzer so einfach und intuitiv wie möglich gestaltet werden. Sämtliche Buttons enthalten nebst einem Piktogramm ebenfalls ein Stichwort und Missverständnisse bzw. Missinterpretationen vorbeugen zu können.



Abbildung 3 Button Beispiel

### *Erlernbarkeit:*

Ein unerfahrener User sollte den Umgang und die Funktionen des Toplogy Designers nach maximal einer halben Stunde erlernt haben.

### *Bedienbarkeit:*

Der Einsatz von den einfachen Click Funktionen bei den einzufügen neuen Elementen in die Topologie erlaubt eine schnelle Erstellung einer neuen Topologie

## Effizienz

### *Zeitverhalten:*

Die Ladezeit beim Seitenaufruf soll nicht länger als 500 ms dauern. Sämtliche Operationen innerhalb der Applikation müssen innerhalb 500ms durchgeführt und den Benutzer rückgemeldet werden.

### *Verbrauchsverhalten:*

Die Applikation soll bis zu 20 parallele Zugriffe von verschiedenen Usern performant erlauben

## Übertragbarkeit

### *Anpassbarkeit:*

Die Applikation ist mit Python (Django) implementiert und plattformübergreifend nutzbar. Das Frontend ist ebenfalls Plattformunabhängig, da es in eine statische Webseite gebildet wird.

## Tabelle 6 Nicht Funktionale Anforderungen

## 3.2 Analyse

### 3.2.1 Objektkatalog

<b>Node</b>	Ein Node ist ein abstraktes Objekt für die verschiedenen Geräte der Topologie. Es werden folgende Informationen in den Nodes gespeichert: <ul style="list-style-type: none"><li>- Den vom Lab Topology Builder vorgegebenen Gerätenamen (z.B. virtual_machine)</li><li>- Der Gerätetyp (Hardware Image z.B. csr1000v) ebenfalls vorgegeben</li><li>- Einen Default Name für das Gerät</li></ul>
<b>Edge</b>	Edges stellen die Verbindungen zwischen den Geräten dar. Ein Node kann ein oder mehrere Verbindungen besitzen. In einer Edge wird gespeichert welche Geräte durch sie verbunden werden.
<b>Virtual Network Device</b>	Die Nodes müssen zusätzlich unterschieden werden. Netzwerkgeräte benötigen eine Run Konfiguration, welche ebenfalls erfasst und gespeichert werden soll.
<b>Device</b>	Im Vergleich zu den Netzwerkgeräten benötigen «normale» Geräte lediglich eine IP-Adresse und ein Gateway. Weiterführende Konfigurationen werden nach dem Deployment via Lab Topology Builder manuell eingegeben.
<b>YAML File</b>	Die YAML Files bilden ebenfalls ein Objekt. Es werden YAML Files zum Import sowie zum Export der Topologie Informationen verwendet. Die Formatierung bzw. das Gerüst der Files ist vom Lab Topology Builder vorgegeben und wurde gemäss Vorgabe adaptiert. Sämtliche abgebildeten bzw. konfigurierten Werte werden in die Files geschrieben.

Tabelle 7 Objektkatalog

### 3.2.2 Domain-Modell, Klassendiagramm (konzeptionell)

#### 3.2.2.1 Domain Modell

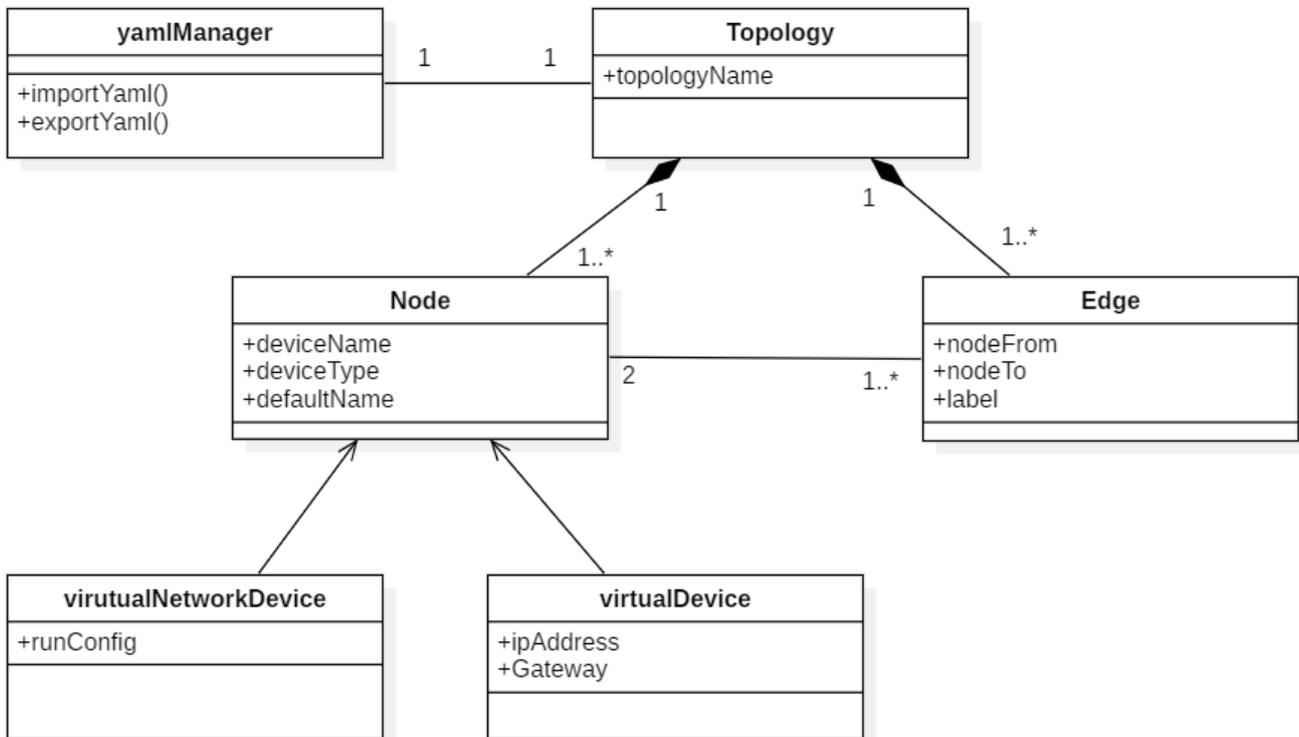


Abbildung 4 Domain Modell

Im Domain Modell werden die wichtigsten Assoziationen zwischen den Objekten dargestellt. Dabei wurde darauf geachtet, keine unnötige Komplexität einzubauen, um das Projekt nicht unnötig aufzublasen.

## 3.3 Design Entwurf

### 3.3.1 Wireframes

Die Wireframes wurden mit dem Tool «Balsamiq-Cloud»<sup>9</sup> erstellt. Balsamiq ermöglicht den Einsatz von «Clickable Wireframes» welche einen ersten klickbaren Eindruck in die Applikation erlauben.

#### Start Screen

Beim Aufruf der Seite wird eine «leere» Topologie erstellt. Auf der Seite wird eine «Werkzeugleiste» geladen. Die Leiste ist in «2 Teile» unterteilt. Im oberen Bereich sind die verschiedenen Geräte aufgelistet, dessen «Informationen» (Konfigurationsoptionen etc.) aus dem Backend geladen werden. Zudem sind im unteren Bereich sind die Befehle aufgelistet.

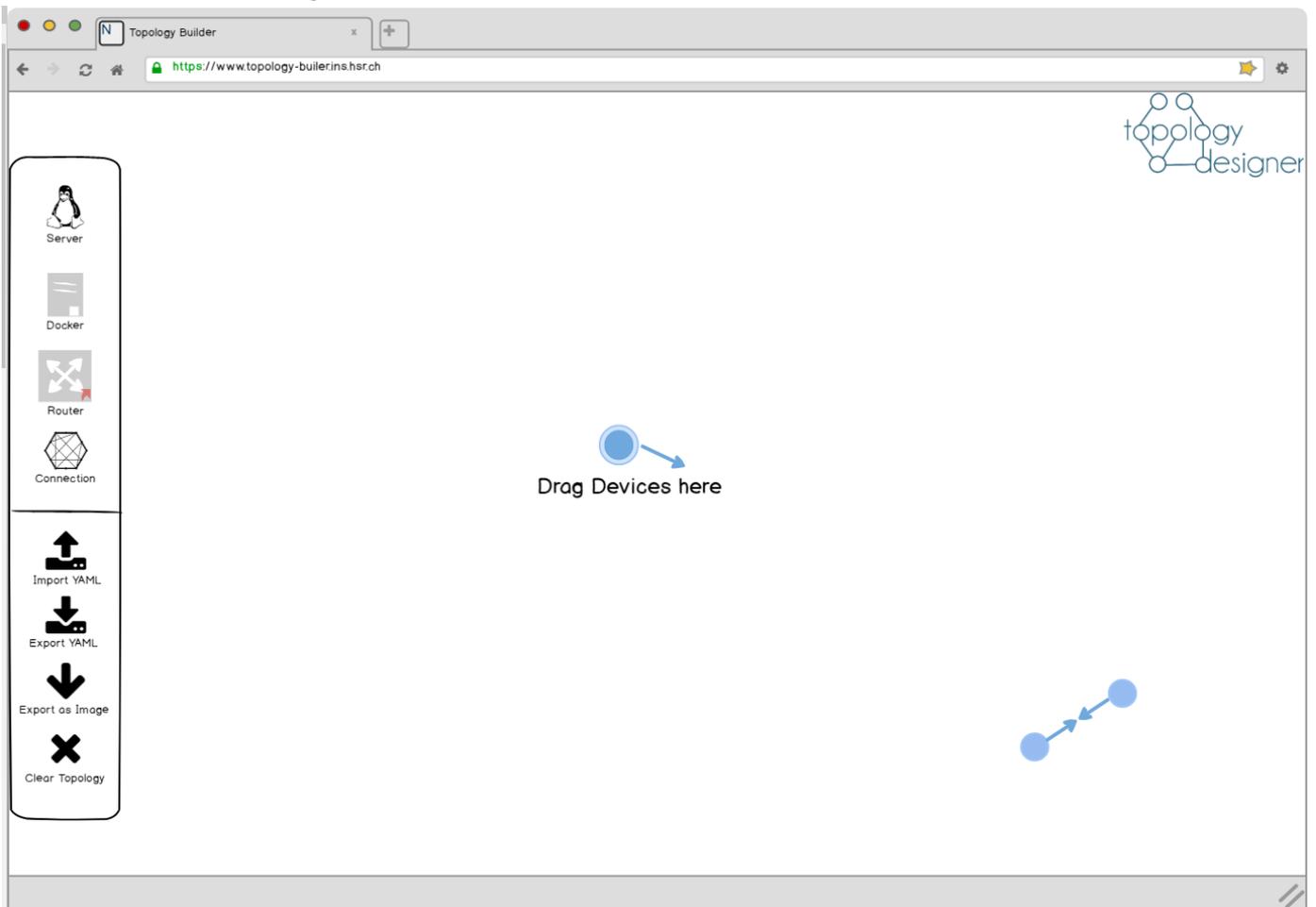


Abbildung 5 Topology Designer Startscreen

<sup>9</sup> <https://balsamiq.com/wireframes/cloud/>

## Neues Gerät hinzufügen

Neue Geräte können einfach aus der Werkzeugliste in den Zeichenbereich gezogen werden.

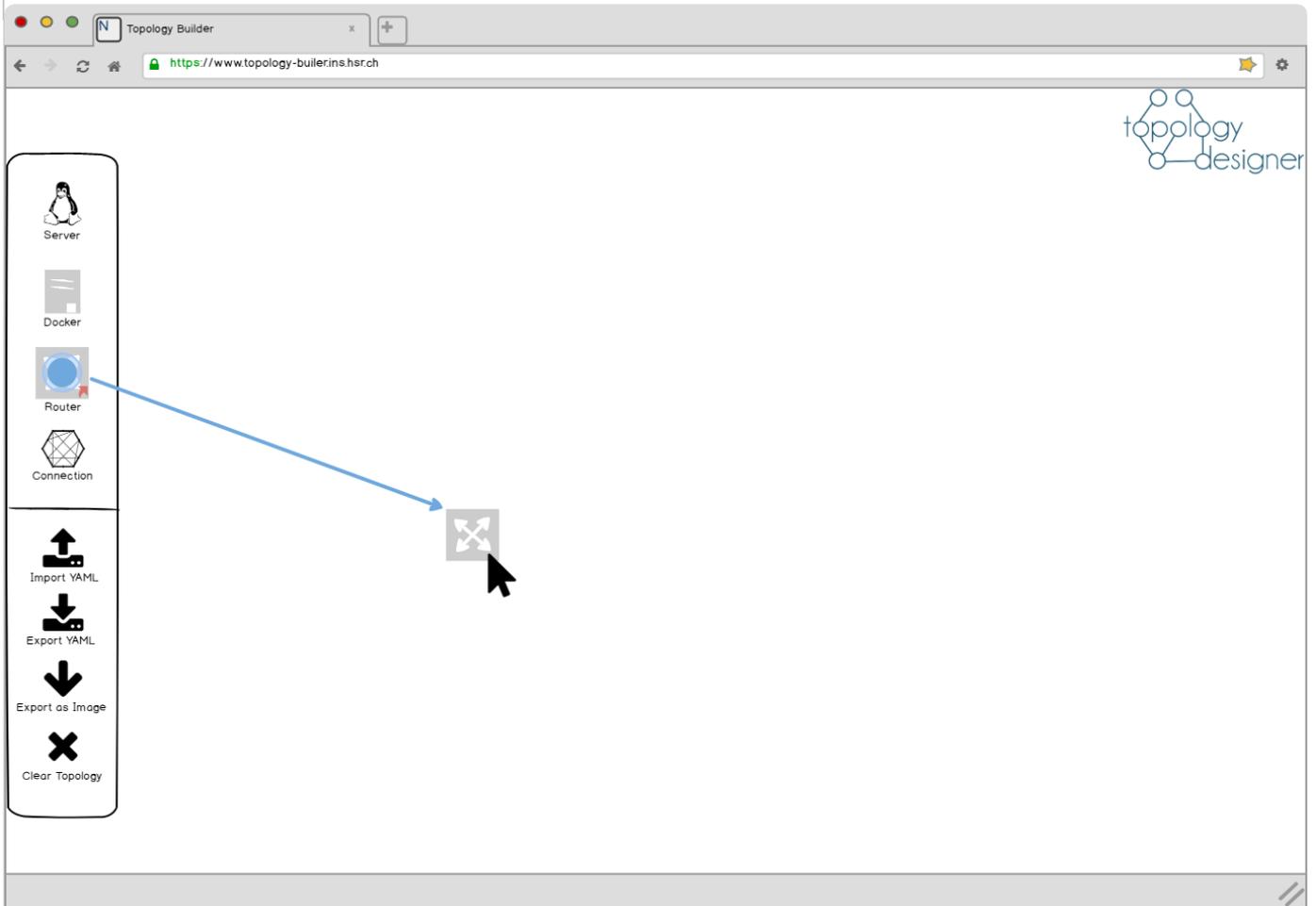


Abbildung 6 Topology Designer Create Device

## Geräte verbinden

Um eine Verbindung zwischen 2 Geräten herzustellen muss zuerst auf den «Connection Button» geklickt werden, anschliessend kann die Verbindung mit geklickter Maustaste von einem Gerät zum anderen gezogen werden.

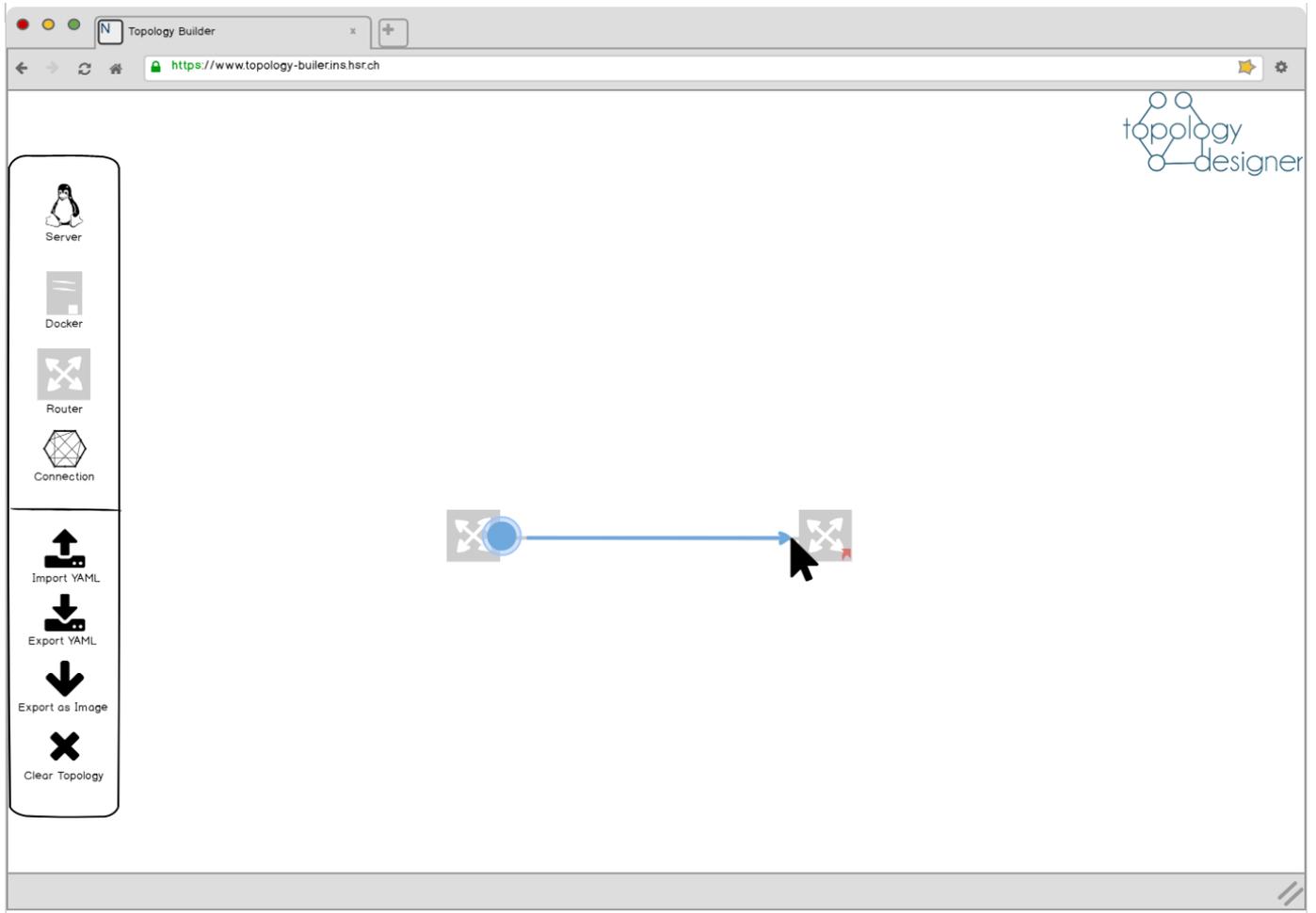


Abbildung 7 Topology Designer Create Connection

## Gerät konfigurieren

Beim Klick auf ein erstelltes Gerät erscheint die Konfigurationsleiste mit welcher sich die Basisinformationen sowie die Run-Configs eingeben lassen.

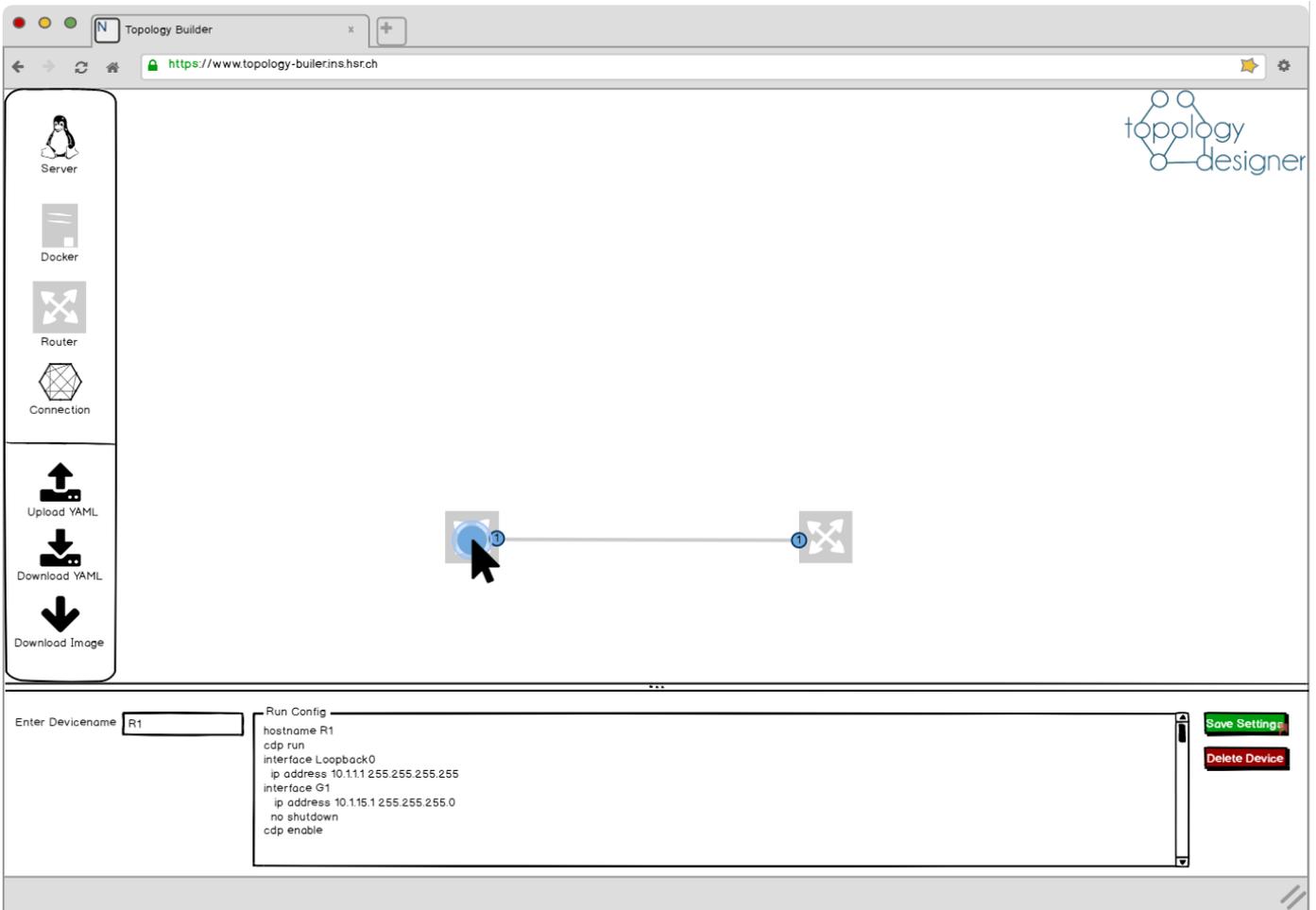


Abbildung 8 Topology Designer Device Configuration

## Import Topology

Es ist auch möglich eine Topologie anhand mit einem YAML File zu importieren. Beim Klick auf «Import Topology» öffnet sich ein Dialog in welchem man das YAML File angeben muss. Beim Klick auf «OK» wird die Topologie importiert, modifiziert und anschliessend wieder exportiert werden.

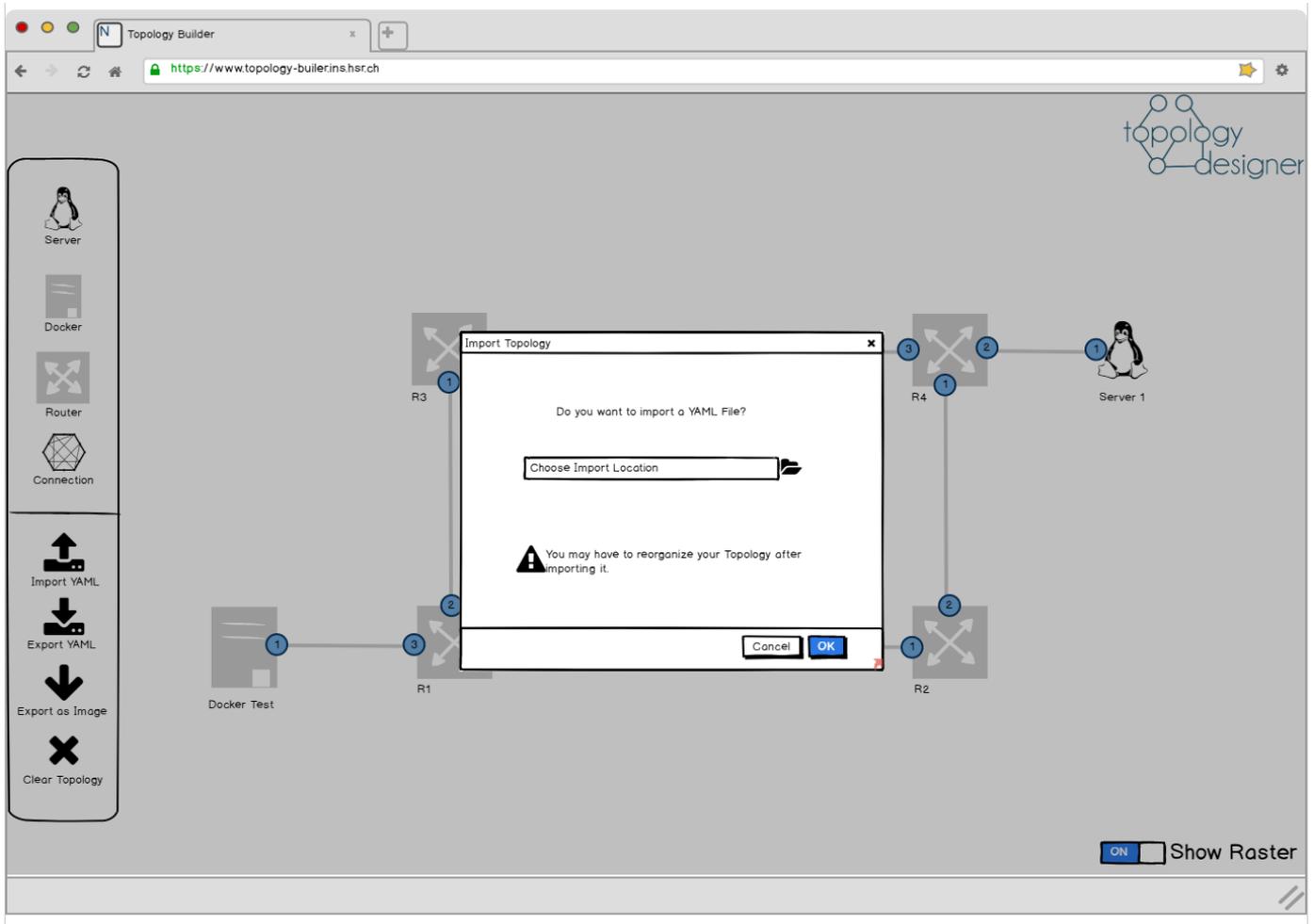


Abbildung 9 Topology Designer Import Topology

## Erstellte Topologie

Eine fertig konfigurierte Topologie könnte wie folgt aussehen:

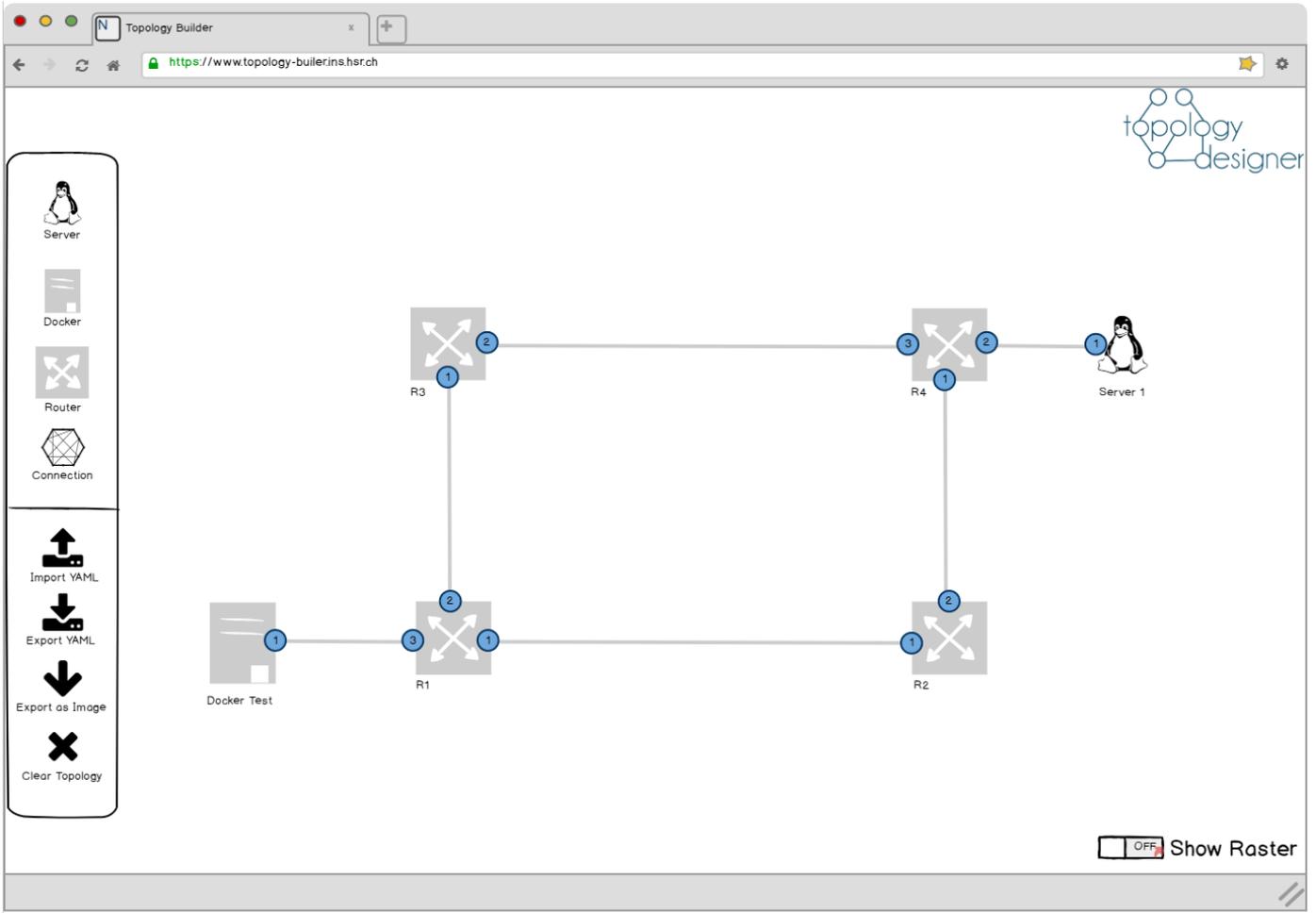


Abbildung 10 Topology Designer Fertige Topologie

## Topologie exportieren

Die erstellte Topologie kann in ein YAML File exportiert werden, welches anschliessend mit dem Tool Lab Topology Builder<sup>10</sup> eingelesen, und deployed wird. Beim Klick auf «Export YAML» öffnet sich eine Dialogbox. Es muss der Speicherort des YAML Files definiert werden. Zudem wird ein Hinweis angezeigt, dass die Korrektheit der Topologie sowie den Konfigurationen nicht geprüft wird und Fehler ebenfalls ins Yaml File übernommen werden.

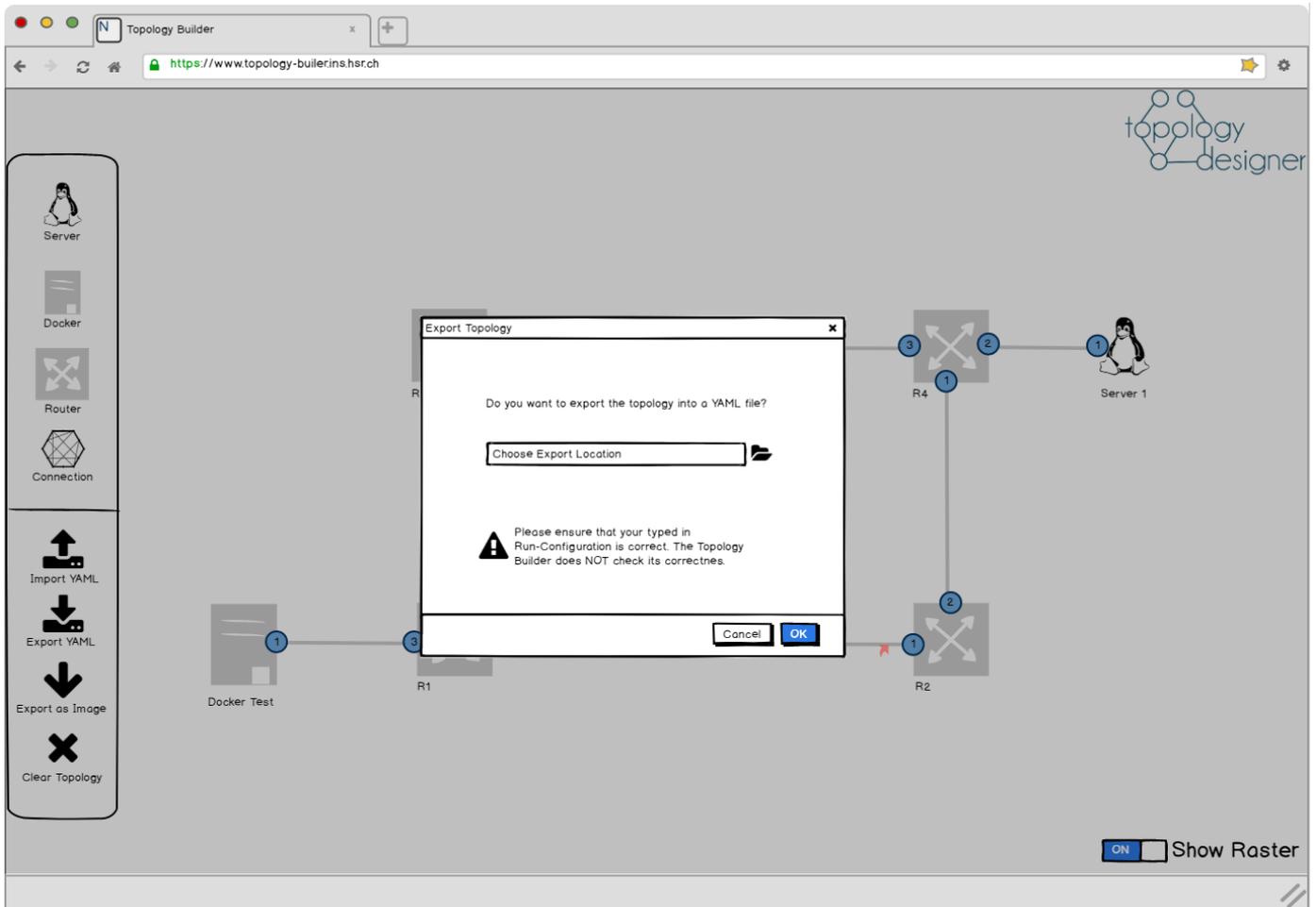


Abbildung 11 Topology Designer Export YAML

<sup>10</sup> Vorhergehende Studienarbeit für das INS <https://eprints.hsr.ch/734/>

## Export Image

Die Topologie kann ebenfalls als Bild exportiert werden. Das Bild kann bei der Durchführung eines Netzwerklabs den Studenten zur Orientierung mit der Aufgabenstellung abgegeben werden.

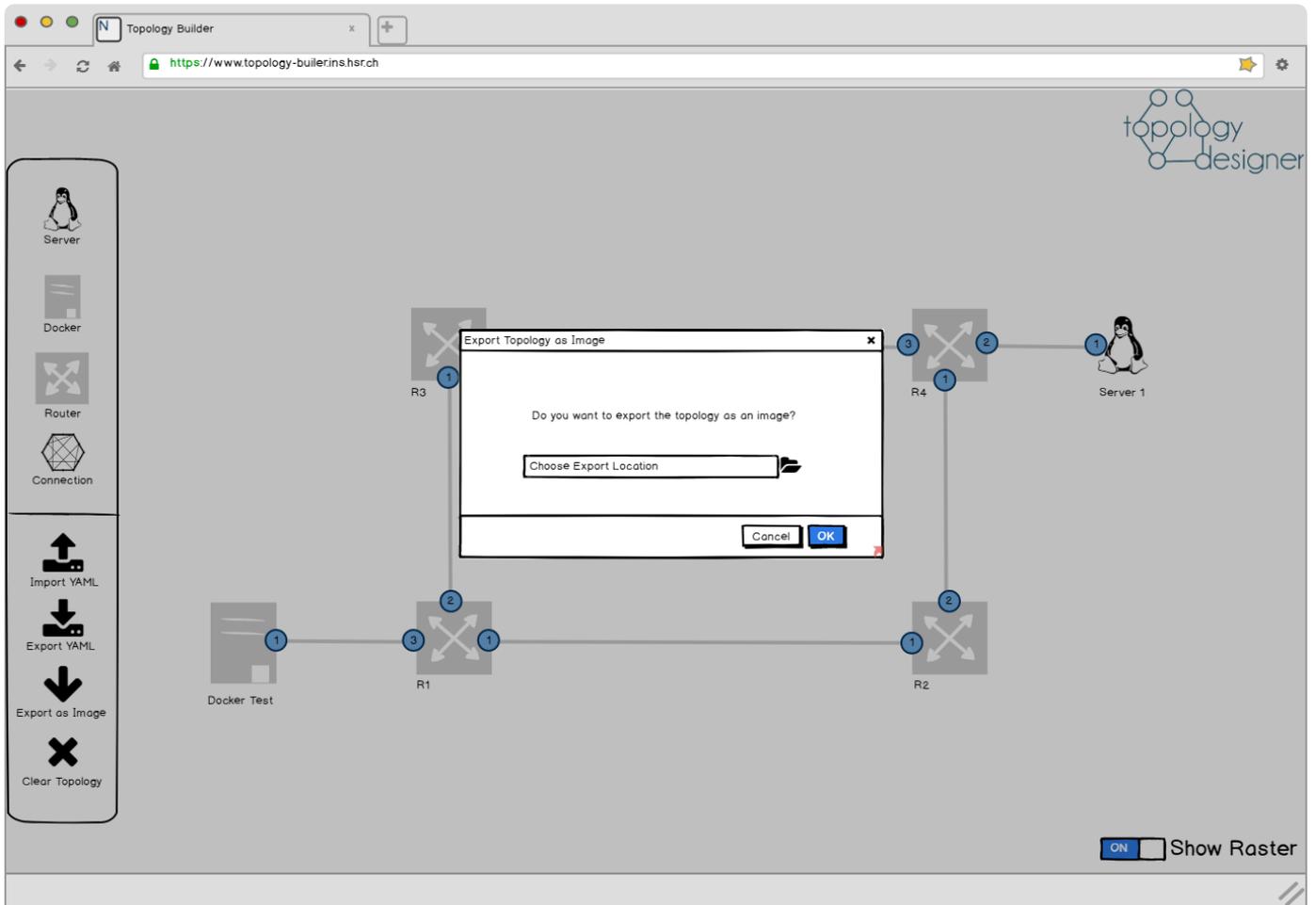


Abbildung 12 Topology Designer Export Image

## Clear Topology

Für den Fall, dass mehrere Topologien hintereinander erstellt werden sollen, ist es möglich die aktuell angezeigte zu «löschen». Beim Klick auf Clear Topology wird zuerst ein Warndialog eingeblendet, dass beim Klick auf «Yes» die Topologie gelöscht wird.

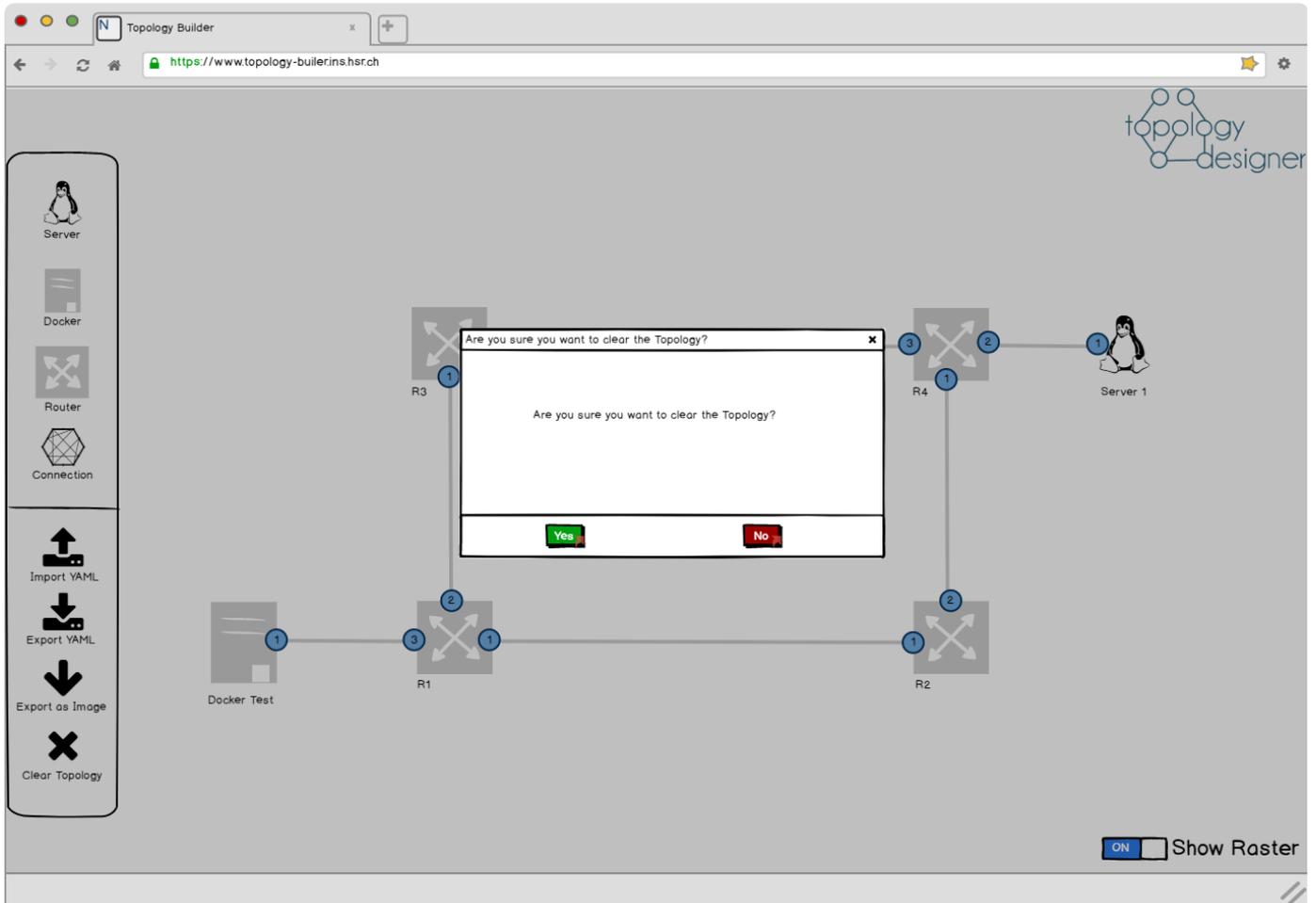


Abbildung 13 Topology Designer Clear Topology

Die Wireframes haben uns schon früh eine Designgrundlage geboten. Wir haben uns in der Construction Phase stark daran orientiert, was uns viele Design Entscheide erspart hat.

### 3.3.2 Sequenzdiagramm

Für die Wichtigsten Abläufe sind Sequenzdiagramme definiert. Die Trennung von Frontend und Backend ist dabei besonders wichtig, damit diese getrennt voneinander entwickelt werden können. Die Kommunikation erfolgt über eine Rest API.

Die Website wird geladen

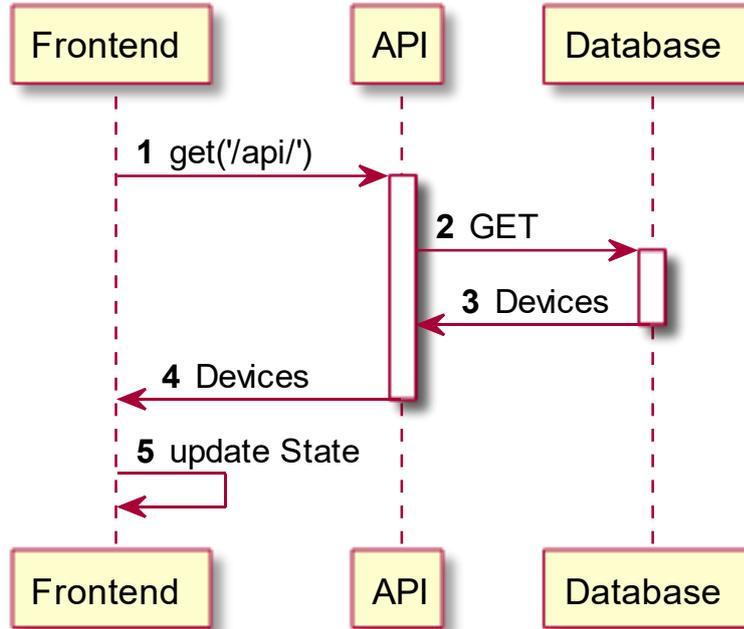


Abbildung 14 Sequenzdiagramm Zugriff auf Database

Tabelle 8 Sequenzdiagramme

## 3.4 Implementation (Entwicklung) und Tests

### 3.4.1 Infrastruktur

#### 3.4.1.1 Workflow

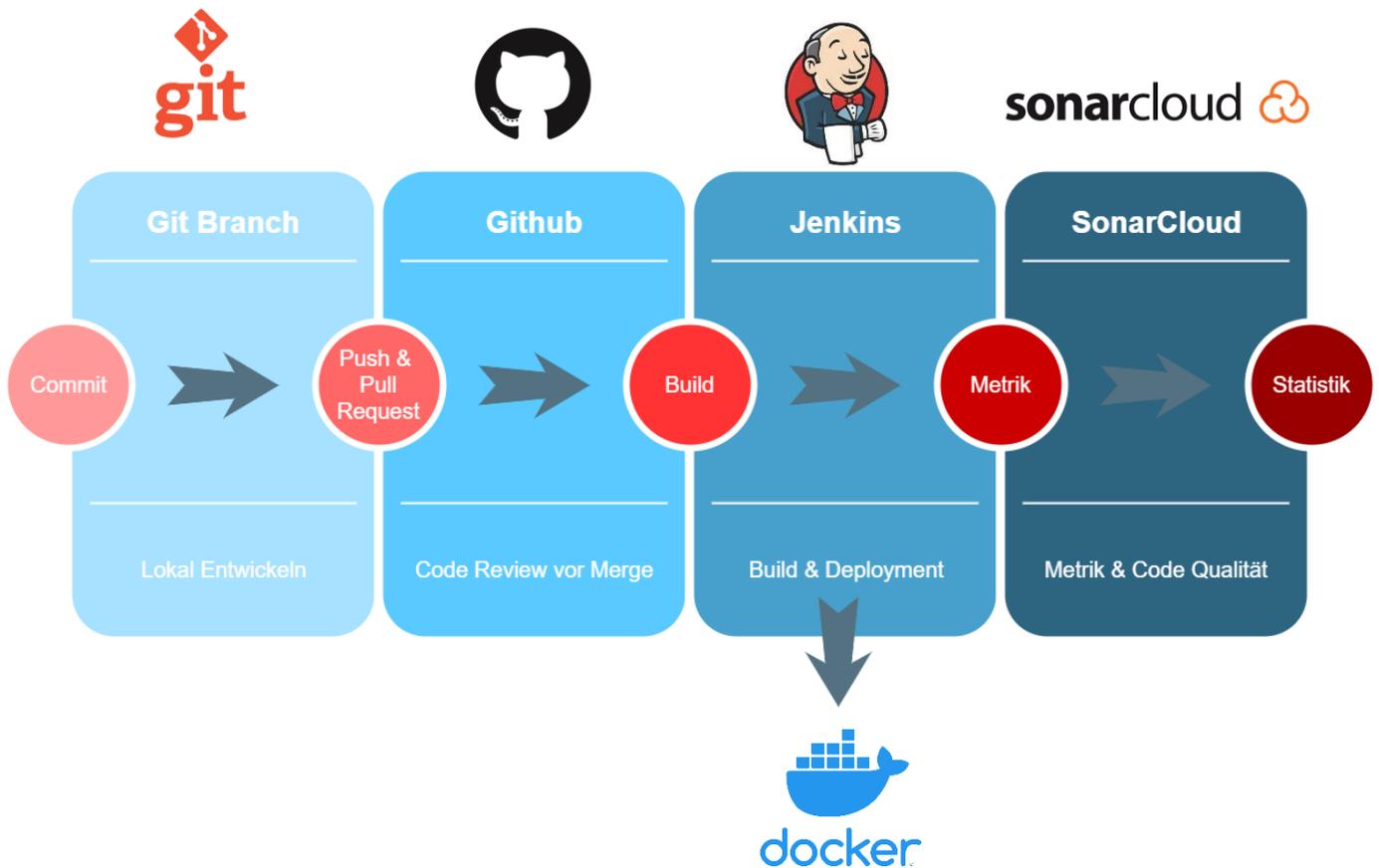


Abbildung 15 Workflow

Für die Entwicklung an der eigenen Arbeitsstation verwenden wir WebStorm von Jet Brains für das Frontend sowie Visual Studio Code für das Backend. Sämtlicher Arbeitsfortschritt wird mit lokal via git-Commit getrackt. Vor der Durchführung des Commits, führt die IDE automatisch eine Formatierung, eine Optimierung der Imports sowie eine Code Analyse durch. Somit werden unnötige Differenzen zwischen den Branches ausgemerzt und es dient als zusätzliche Denkstütze sauberen Code zu comitten. Der aktuelle Status wird auf einen separaten Branch in einem GitHub Repository gepusht. Grundsätzlich werden die Branches bei Sprintende in den Masterbranch gemerged. Da es jedoch vorkommen kann, dass wir untereinander von den Funktionen des «Partnerbranches» abhängig sind, kann auch vor Sprintende manuell ein Pull Request eröffnet werden.

Der Pull Request wird immer vom Team Partner ausgeführt. An dieser Stelle wird der neu generierte Code nochmals geprüft. Allfällige Unklarheiten bzw. Meinungsverschiedenheiten werden ausdiskutiert.

Sobald ein der Pull Request gemerged wurde, wird ein Build Job auf dem Jenkins Server angestossen. Der Build Job generiert 2 Docker Container, welche in die Docker Registry gepusht werden und von da aus gestartet werden können.

Um einen gewissen Code Standard einhalten zu können und um Statistiken generieren zu können wurde das Projekt mit dem Online Metrik Tool Sonar Cloud verknüpft.

### 3.4.1.2 Genutzte Systeme

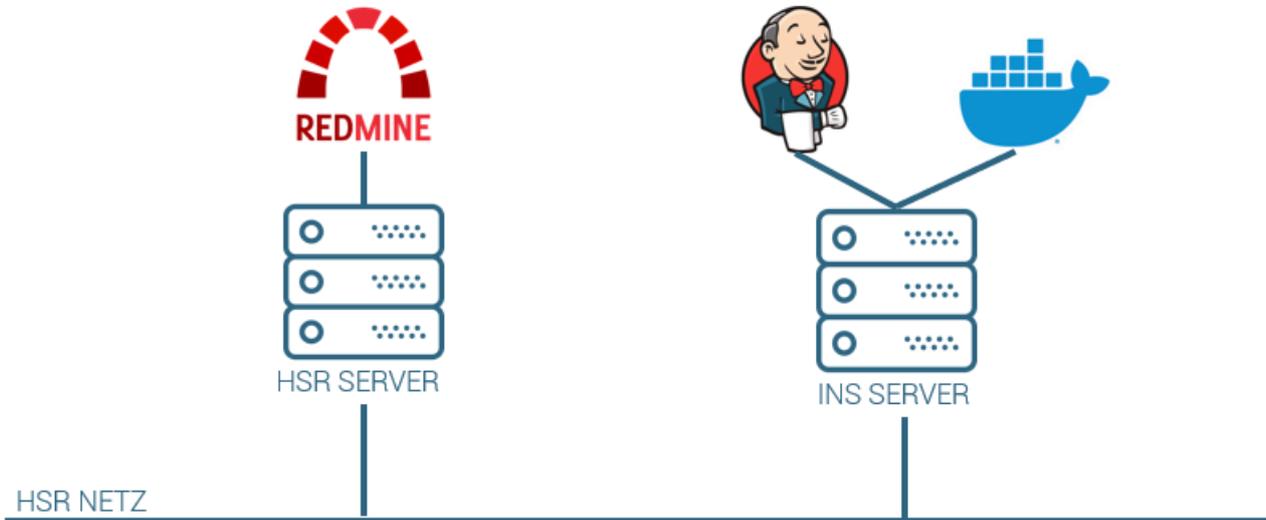


Abbildung 16 Genutzte Systeme

Es wurde beim Serverkiosk der HSR ein Projektmanagement Server mit vorinstalliertem Redmine Server bestellt. Dieser wurde genutzt zur Erfassung der Arbeitspakete sowie der Zeiterfassung. Ausserdem wurde ein vom INS bereitgestellter Server benutzt um die CI/CD Pipelines mit Jenkins zu realisieren sowie die daraus gebildeten Docker zu hosten.

### 3.4.2 Implementation: Erläuterungen wichtiger konkreter Klassen

#### 3.4.2.1 Frontend

Name	Beschreibung
SingleDrawing	<p>Kernklasse des Frontends. Enthält den Constructor und das Rendering des Graphen.</p> <p>Es werden hier alle Funktionen aufgerufen, die für das Erstellen, Löschen, Bearbeiten von Nodes und Edges, sowie für das Importieren und Exportieren einer Topologie notwendig sind.</p>
DeleteTopologyDialog	<p>Dieser Dialog erscheint, wenn man die ganze Topologie löschen möchte. Abfrage ob man wirklich löschen möchte. Bei Bestätigung wird die Topologie gelöscht, bei Abbruch geschieht nichts.</p>
EditNodeDialog	<p>Dialog zum Bearbeiten von Virtual Network Devices. Der Name des Devices, sein Typ und seine runconfig können angepasst werden. Die Devices, mit denen er verbunden ist, werden mit dem entsprechenden Port dargestellt. Die Änderungen können gespeichert oder verworfen werden.</p>
EditEdgeDialog	<p>Dialog zum Bearbeiten einer Verbindung. Der Name der Verbindung kann angepasst werden. Wenn ein nicht Virtual Network Device Teil der Verbindung ist, kann man den Namen, den Typ, die IP-Adresse und das Gateway anpassen. Die Änderungen können gespeichert oder verworfen werden. Verbindungen zwischen zwei Virtual Network Devices können nicht bearbeitet werden.</p>

Tabelle 9 Frontendklassen

#### 3.4.2.2 Backend

Name	Beschreibung
Devices	<p>Definiert das Model der Devices. Enthält alle Attribute, die in der Datenbank abgespeichert werden:</p> <ul style="list-style-type: none"> <li>•name: Bezeichnung des Device Typs (z.B. virtual_network_devices, docker_containers, virtual_machines)</li> <li>•type: Enthält default Wert für den Typ des Devices</li> <li>•defaultName: wird als Label des Devices in der Applikation angezeigt</li> <li>•icon: Darstellung des Devices in der Applikation</li> </ul>
DeviceSerializer	<p>Der Serializer ist dafür zuständig die Daten in der Datenbank in ein, für Webapplikationen, lesbares Format (z.B. JSON) umzuwandeln.</p>
DeviceListView	<p>Definiert die Darstellung aller Devices in der Datenbank in der API.</p>
DeviceDetailView	<p>Definiert die Darstellung eines Bestimmten Devices in der Datenbank in der API</p>

Tabelle 10 Backendklassen

### 3.4.2.3 Klassendiagramm

In der Component Klasse wird im React State die eigentliche Topologie gespeichert. Sämtliche Dialoge wurden im Package UI realisiert. Die Klassen im Functions Package erweitern die SingleDrawing Klasse mit Funktionen für die Modifizierung der Topologie.

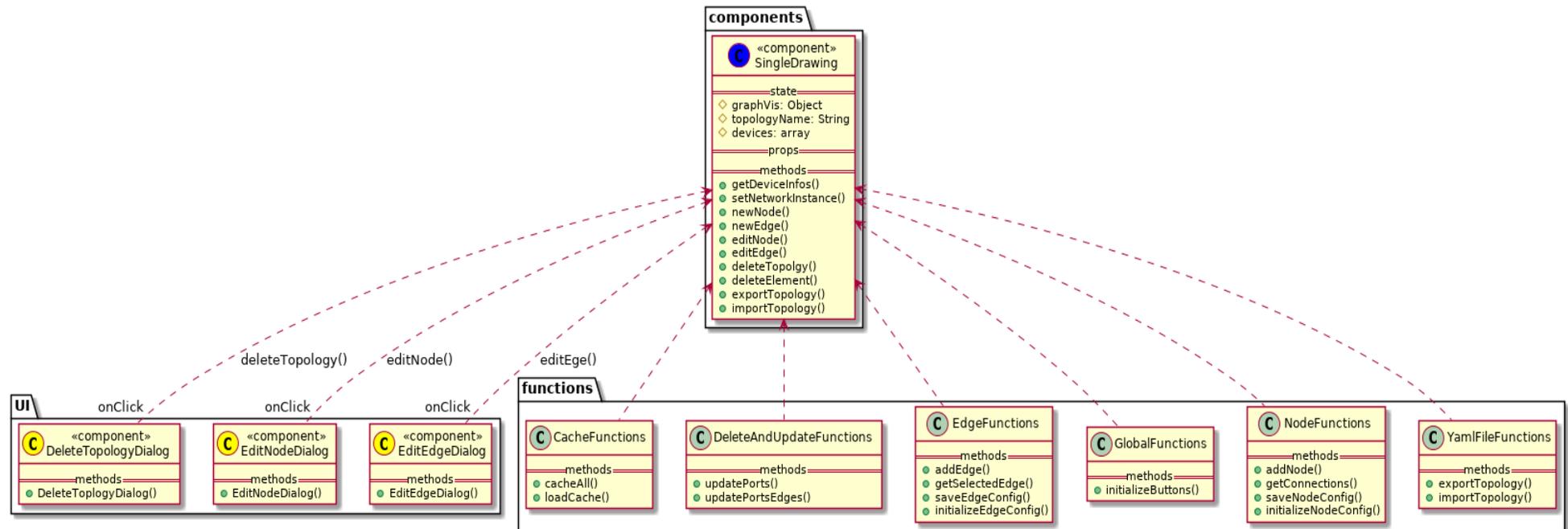


Abbildung 17 Klassendiagramm

### 3.4.3 Deployment Diagramm

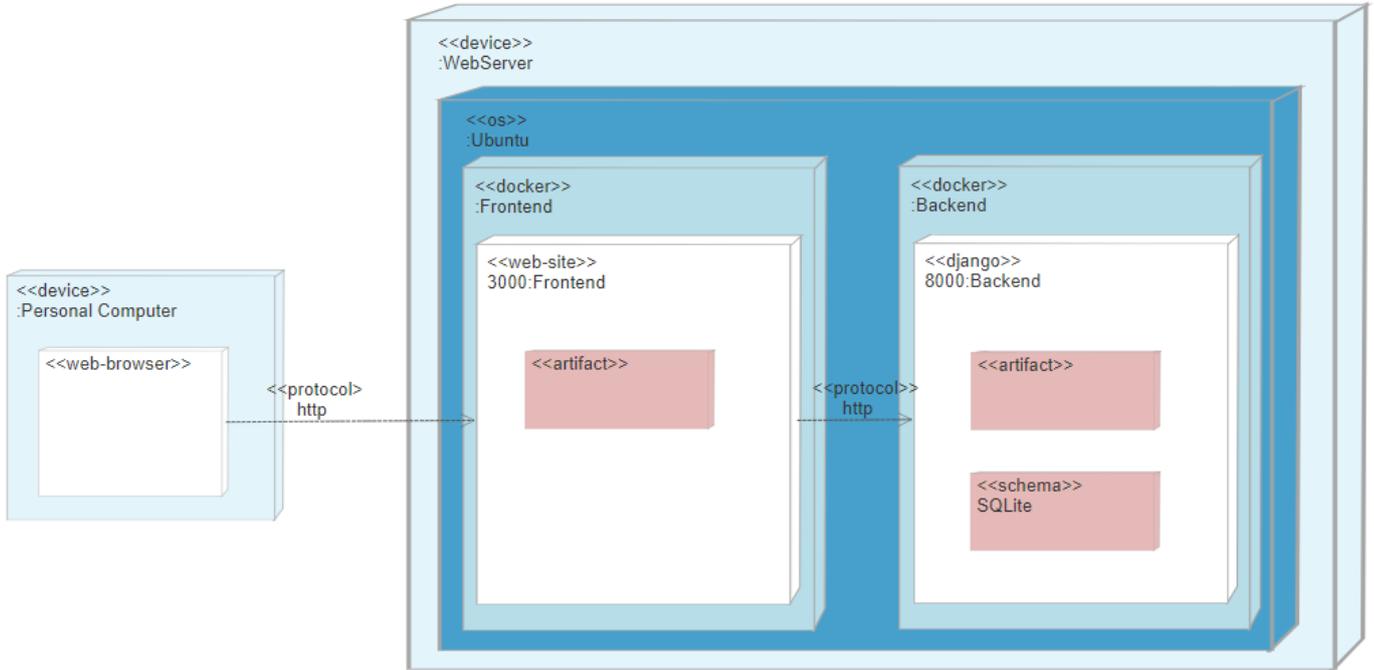


Abbildung 18 Deployment Diagramm

Die Applikation wird auf zwei Docker Containern ausgeführt. Der Client greift via HTTP auf das Frontend zu. Das Frontend wurde auf einem Container mit NodeJs realisiert. Sämtliche Applikationslogik wurde im Frontend umgesetzt, da der Schwerpunkt der Applikation auf die Gestaltung der Topologie gelegt wurde. Die angezeigten Geräte im Frontend werden beim Aufruf dynamisch aus dem Backend generiert. Das Backend wurde mit Django realisiert und beinhaltet eine Administrationskonsole, ein API sowie eine SQLite Datenbank. Das Frontend benötigt keinen Schreib-Zugriff auf das Backend, da im Frontend keine Daten persistent gespeichert werden müssen. Die erstellten Topologien im Frontend müssen nach Beendigung nicht mehr abrufbar sein, da sie in ein YAML File exportiert wurden. Falls Änderungen an der exportierten Topologie vorgenommen werden sollen, kann das YAML einfach im Frontend wieder importiert werden. Der Einzige Zugriff auf das Backend ist die Abfrage der Geräteinformationen beim Aufruf der Webseite.

### 3.4.4 Automatische Testverfahren

<b>Frontend</b>	Im Frontend werden die Funktionen exportYaml und importYaml getestet. Dazu haben wir das Testing Framework jest <sup>11</sup> verwendet.
<b>Backend</b>	Da das Backend keinerlei Funktionalität enthält und wir keine eigenen Methoden definiert haben, haben wir uns dazu entschlossen, keine automatischen Tests zu schreiben.

Tabelle 11 Automatische Testverfahren

### 3.4.5 Usability Testing

Es wurde ein Usability Test durchgeführt, um die Benutzerfreundlichkeit bzw. Intuition unserer Anwendung zu testen. An dieser Stelle werden lediglich die Erkenntnisse aufgeführt. Das Testkonzept sowie -protokoll werden in Anhang aufgeführt.

#### Ergebnis

Der Usability Test hat gezeigt, dass die meisten Funktionen sehr intuitiv und einfach sind. Die Testpersonen hatten einen guten Eindruck von der Anwendung, auch wenn sie ein Paar Verbesserungsvorschläge platziert haben.

- Konfiguration der Docker ebenfalls direkt auf den Geräten anstelle der Verbindungen
- Verbindungen ziehen können (nicht 2 Geräte auswählen und mit Klick verbinden)

Im Rahmen der Construction Phase konnten die Verbesserungsvorschläge leider nicht mehr rechtzeitig implementiert werden.

---

<sup>11</sup> <https://jestjs.io/>

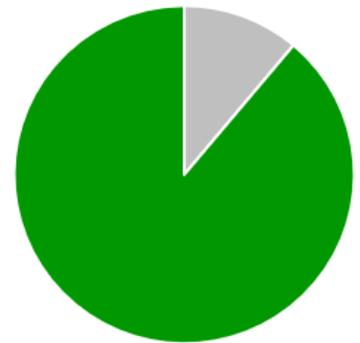
## 3.5 Resultate und Weiterentwicklung

### 3.5.1 Resultate

- UC01 IMPORT TOPOLOGY
- UC03 CLEAR TOPOLOGY
- UC04 CRUD DEVICE
- UC05 CRUD CONNECTIONS
- UC06 EXPORT TOPOLOGY AS YAML
- UC07 EXPORT TOPOLOGY AS PICTURE

- 
- UC02 REORGANIZE TOPOLOGY
  - UC05 CRUD DEVICETYPE
  - UC09 RESTORE TOPOLOGY FROM CACHE

IMPLEMENTIERT



- IMPLEMENTIERT
- TEILWEISE IMPLEMENTIERT

Abbildung 19 Übersicht über alle erfüllten und nicht erfüllten Use Cases

Die von uns gesetzten Ziele haben wir alle erreicht. Wir konnten alle Use Cases implementieren, teilweise mit kleineren Abweichungen.

Der UC07 Export Topology as Picture konnte nur teilweise umgesetzt werden. Es ist möglich ein Bild zu exportieren, dies geschieht jedoch nicht über einen von uns implementierten Button, sondern über die Funktion des Webbrowsers (siehe 4.2.6 Topologie als Bild exportieren).

Die Erstell Funktion des UC04 mussten wir etwas abändern. Anstatt dass das Device mit Drag'n Drop erstellt wird, gibt es nun einen Button. Wenn dieser gedrückt wird erscheint ein Device in der Topologie, dieses kann nun frei in der Topologie bewegt werden.

Betreffend des UC05, kann man nun auf den Verbindungen Konfigurationen vornehmen, dies war zuerst nicht geplant. Auf Grund der Natur der Virtual Machines und der Docker Container haben wir beschlossen, dass die Konfiguration dieser zwei Devices über die Verbindung gemacht wird.

### 3.5.2 Möglichkeiten der Weiterentwicklung

<b>Einbindung in den LTB</b>	<p>Der Topology Designer wurde entwickelt, um die Benutzung des Lab Topology Builders (LTB) zu vereinfachen. Der Topology Designer erstellt ein YAML File, welches anschließend im LTB eingelesen wird. Idealerweise kann das YAML File nach dem Erstellen direkt auf den LTB gepusht werden.</p> <p>Ein Zugriff aus dem LTB auf den Topology Designer könnte auch in Betracht gezogen werden.</p>
<b>UC07 fertig implementieren</b>	<p>Da der UC07 nicht ganz vollständig implementiert wurde könnte man diesen noch fertig stellen. Wir hatten bereits eine konkrete Idee wie man dies umsetzen könnte, jedoch hatten wir keine Zeit mehr diese zu implementieren.</p> <p>Der Export hat ein crossorigin Problem mit den Bildern aus unserem Backend. Dieses Attribut kann man nicht setzen mit der Bibliothek, die wir benutzen. Man kann einen Proxy vor die beiden Docker Container des Frontend und des Backend setzen. Dadurch haben das Frontend und das Backend die gleiche origin und das Problem ist nicht mehr vorhanden.</p>
<b>Konfigurationen prüfen</b>	<p>Die eingegebenen Konfigurationen in den Devices werden nicht auf ihre Korrektheit überprüft. Der Benutzer ist selbst dafür verantwortlich diese richtig einzugeben. Man könnte also einen Checker implementieren, der vor abspeichern der Nodes oder Edges überprüft ob die Eingaben korrekt sind.</p>
<b>Templates vordefinieren</b>	<p>Man könnte Templates von nicht konfigurierten Topologien vorgeben, damit das Erstellen von Topologien noch einfacher wird. Man müsste einige Files erstellen und diese im backend oder Frontend hinterlegen. Das Lesen bzw. Laden dieser Topologien kann mit dem importTopology erfolgen.</p>
<b>Schnittstelle zu den physischen Geräten</b>	<p>Es kann sein, dass in einem Lab physische Geräte zur Topologie hinzugefügt werden sollen. Im Rahmen der Studienarbeit wurde dieses Szenario nicht weiter berücksichtigt. Physische Geräte könnten jedoch ebenfalls im Backend hinterlegt werden und anschliessend im Topology Designer eingefügt und konfiguriert werden.</p>
<b>Neuordnung der Topologie nach Laden unterbinden</b>	<p>Wenn die Seite neu geladen wird, ordnet sich die Topologie wieder neu an. Man könnte versuchen über Positionsdaten der Nodes die Topologie wieder genau gleich erscheinen zu lassen wie vor dem Verlassen der Seite.</p>

Tabelle 12 Mögliche Weiterentwicklungen

## 3.6 Projektmanagement – Planung/Soll

### 3.6.1 Prozessmodell

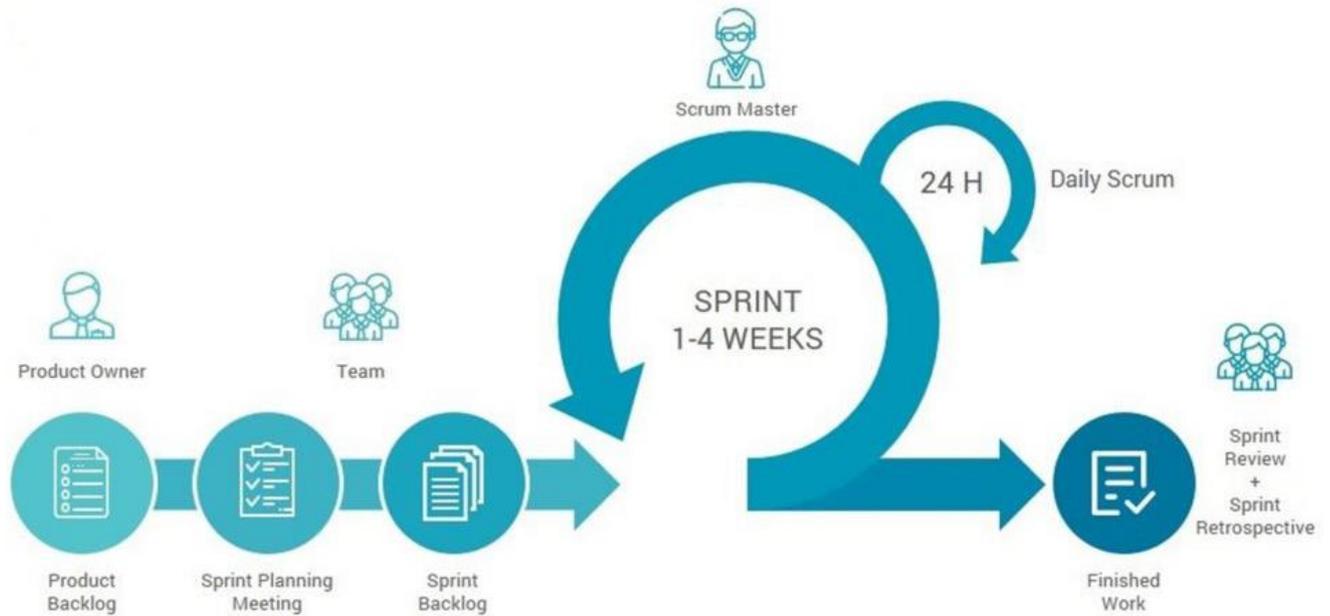


Abbildung 20 Prozessmodell

#### Scrum mit Unified Prozess

Als Prozessmodell haben wir uns für Scrum mit Unified Process entschieden. Dieses Modell haben wir beide schon bei unserem Engineering Projekt verwendet und haben damit gute Erfahrungen gesammelt. Es bietet uns eine gute Struktur mit festen Terminen, damit am Ende ein funktionierendes Produkt steht und trotzdem sind wir noch agil genug.

### 3.6.2 Team, Rollen und Verantwortlichkeiten

<b>Vanessa Janknecht</b>	Entwicklung, Qualitätskontrolle, Dokumentation, Testing, Design, Analyse, Zeiterfassungskontrolle
<b>Martin Hug</b>	Entwicklung, Qualitätskontrolle, CI, Design, Dokumentation, Analyse, Zeiterfassungskontrolle
<b>Laurent Metzger</b>	Betreuer, Kunde
<b>Sebastian Hug</b>	Betreuer, Kunde

Tabelle 13 Rollen und Verantwortlichkeiten

### 3.6.3 Aufwandschätzung

Jedes Teammitglied arbeitet 16 Stunden pro Woche über 13.5 Wochen hinweg. Das Projekt startet am 16.09.2019 und endet am 17.12.2019. Dies erbiert einen Totalaufwand von 216 Stunden pro Person. Insgesamt sind dies 432 Stunden.

Die 432 Stunden werden vollständig aufgebraucht. Wenn Zeit übrigbleibt, werden zusätzliche Funktionalitäten implementiert.

### 3.6.4 Zeitplan

Die Zeitliche Planung wird über Redmine gestaltet. Die Planung ist möglichst Agile. Es gibt nur folgende festgelegte Termine in der Planung:

- **6. Dezember** Feature Freeze
- **16. Dezember** Abgabe Abstract
- **17. Dezember** Projektabgabe
- **13. Dezember** Schlusspräsentation

Der Rest wird fortlaufend agil, mit einer voraussichtlichen Iterationslänge von 2 Wochen, in Arbeitspaketen geplant.

### 3.6.5 Iterationen

#### Überblick

Inception	Elaboration		Construction				Transition
I1 16.09.-27.09	E1 28.09.-11.10	E2 12.10-18.10	C1 19.10-01.11	C2 02.11-15.11	C3 16.11-29.11	C4 30.11-06.12	T1 07.12-20.12
	M1	M2	M3	M4	M5	M6	M7

#### Kurzbeschreibung

##### Inception

<b>Sprint 1</b>	Start: 16.09 Ende: 27.09	Projektumfang festlegen, Technologien kennenlernen, Einrichtung GitHub und Redmine
-----------------	-----------------------------	--

##### Elaboration

<b>Sprint 2</b>	Start: 28.09 Ende: 11.10	Projektplanung: NFR, Qualitätsmanagement, Risikomanagement, Domain Modell, Bibliotheken suchen, Wireframes
-----------------	-----------------------------	--

<b>Sprint 3</b>	Start: 12.10 Ende: 18.10	CI/CD aufsetzen, Projektplan: Architektur geplant, Testingmodell, Demo Applikation mit Zugriff auf Datenbank
-----------------	-----------------------------	--

##### Construction

<b>Sprint 4</b>	Start: 19.10 Ende: 01.11	Devices können erstellt werden, Connections möglich, Editierfenster hinzugefügt, Export als yaml möglich.
-----------------	-----------------------------	---

<b>Sprint 5</b>	Start: 02.11 Ende: 15.11	Bilder der Devices sind abrufbar. Yaml-File exportieren fertig. Bild exportieren möglich. Editierfunktionen für Devices und Connections.
-----------------	-----------------------------	--

<b>Sprint 6</b>	Start: 16.11 Ende: 29.11	Import Yaml-File. Design. Alle Device-Typen übernommen. Code Clean-Up.
-----------------	-----------------------------	--

<b>Sprint 7</b>	Start: 30.11 Ende: 06.12	Cache. User Tests. Zusätzliches Thema.
-----------------	-----------------------------	--

##### Transition

<b>Sprint 8</b>	Start: 07.12 Ende: 17.12	Letzte Bugfixes, letzte Tests, Schlussdokumentation, Präsentation
-----------------	-----------------------------	---

Tabelle 14 Projektplan

### 3.6.6 Meilensteine

Inception M1	27.09: Projektumfang geklärt, Zeitplan erstellt, Use Cases fertig
Elaboration M2-M3	11.10: NFR abgeklärt, Domain Modell erstellt, Wireframes 18.10: Architektur klar, GUI Entwurf, Projektplan fertig, CI/CD
Construction M4-M7	01.11: Export eines yaml-Files ist möglich, Verschiedene Devices können ausgewählt werden 15.11: Editieren der Devices und Connections möglich. 29.11: Import Yaml-File. Dynamisches Laden der Buttons. 06.12: Funktionalität von topology designer ist fertig.
Transition M8	17.12: Präsentation. Abgabe Projekt

Tabelle 15 Meilensteine

### 3.6.7 Risikomanagement

Dies ist nur ein kleiner Ausschnitt. Die genaue Beschreibung der Risiken ist in einem separaten File «Risiken.xlsx» abgespeichert.

Nr.	Titel	Beschreibung	Vorbeugung	Verhalten beim Eintreten
R1	Personenausfall	Eine Person fällt krankheitsbedingt für längere Zeit aus.	Die Dokumentation wird immer aktuell gehalten. Das Wissen wird geteilt. An den wöchentlichen Sprintsitzungen werden behandelte Themen besprochen, dass das ganze Team das Wissen besitzt. Zudem wird mit den Pull Requests vorausgesetzt, dass der geschriebene Code gereviewed wird.	Gespräch mit dem Betreuer wird gesucht. Arbeitspakete aufteilen. Übergabe planen.
R2	Funktionalität von Dritten fällt aus	GitHub / Redmine / Server-Anbieter haben einen Ausfall	Lokale Backups, Anbieter Lock-in vermeiden. Dadurch, dass lokale Backups gemacht werden, können die Daten schnell zu einem anderen Anbieter gewechselt werden.	Es wird auf einen anderen Anbieter gewechselt
R3	Unzureichende Fähigkeiten der Mitglieder	Ein oder mehrere Projektmitglieder verfügen nicht über genügend technische Fähigkeiten, um ihre Aufgaben erfüllen zu können	Die Arbeitspakete werden entsprechend den Fähigkeiten der Teammitglieder zugeteilt.	Hilfe von Betreuer bzw. anderen Studenten aufsuchen. Arbeitspakete aufteilen oder umteilen.
R4	Fehlende oder unzureichende Planung	Arbeitspakete fehlerhaft erstellt.	Sämtliche Tätigkeiten werden in einem Arbeitspaket behandelt.	Bei einer Fehlplanung wird im Team das weitere Vorgehen besprochen. Nicht dringende Probleme können

bei der nächsten Sitzung abgehandelt werden. Bei dringenden Problemen wird eine Krisensitzung einberufen.

R5	Fehlende oder unzureichende Dokumentation	Teammitglied kann nicht nachvollziehen was das andere Teammitglied geleistet hat.	Der aktuelle Stand soll im Projektdokument dokumentiert sein.	Die Dokumentation wird laufend aktualisiert.
R6	Soziale Unruhen	Streitigkeiten innerhalb des Teams, Machtkämpfe	Diskussion innerhalb des Teams suchen. Da das Team jedoch nur aus 2 Personen besteht und die Zuständigkeiten klar geregelt sind, sind soziale Unruhen sehr unwahrscheinlich.	Falls das Problem nicht innerhalb des Teams zu lösen ist, wird eine unvoreingenommene Drittperson hinzugezogen, um eine neutrale Sicht über das Problem zu schaffen und eine Entscheidung erleichtern soll.

Tabelle 16 Risikoanalyse

### 3.6.8 Risikomatrix

Die Grösse der Blasen setzt sich zusammen aus dem Produkt von Eintrittswahrscheinlichkeit und maximalem Schaden.

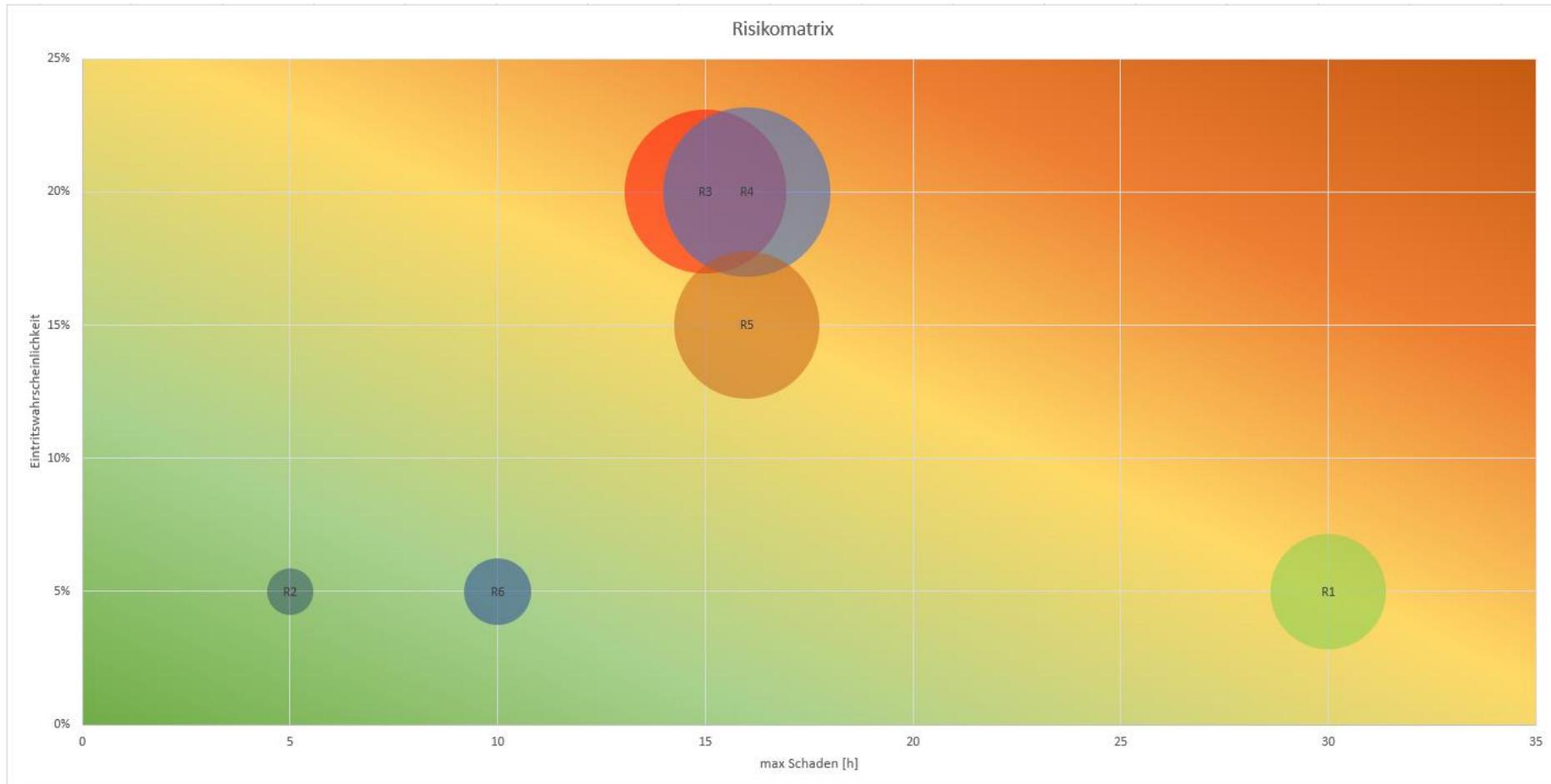


Abbildung 21 Risikomatrix

### 3.7 Qualitätssicherung

<b>Wöchentliche Meetings mit Betreuern</b>	Jede Woche wird eine Sitzung mit den Betreuern abgehalten. In der Sitzung werden die aktuell bearbeiteten Themen präsentiert und die laufenden Resultate vorgestellt. Dies dient der Selbstkontrolle ob wir mit unserer Entwicklung auf den Richtigen Weg sind. Die Meetings werden protokolliert. Die Protokolle werden auf Redmine abgelegt.
<b>Wöchentliche Teammeetings</b>	Mangelnde Kompetenzen oder Fehler möglichst früh erkennen. Planung der nächsten Woche. Absicherung, dass alles erledigt wurde.
<b>4-Augen-Prinzip</b>	Wir bearbeiten beide dasselbe Dokument. Sämtliche Änderungen werden geloggt und können vom Teampartner nachvollzogen werden. Die erstellten Kapitel im Dokument werden gegengelesen und allenfalls korrigiert.
<b>Code Reviews</b>	Bei jedem Pull Request wird der vom Teammitglied generierte Code gereviewed. Dadurch wird zum einen sichergestellt, dass der implementierte Code korrekt ist und zum anderen wird garantiert, dass beide Mitglieder den kompletten Code der Anwendung verstehen.
<b>Automatisierte Tests / CI/CD</b>	Wenn ein Entwicklungs-Branch in den Masterbranch gemerged wird, wird automatisch mit Hilfe von Jenkins eine Pipeline gestartet, welche automatisierte Tests durchführt und 2 Docker Container buildet.
<b>Manuelle Tests</b>	Mit Checkliste, jeweils vor Sprintende.
<b>SonarLint</b>	Frontend Styleguide
<b>PyLint</b>	Backend Styleguide

Tabelle 17 Qualitätssicherung

### 3.8 Projektmonitoring - Ist

#### 3.8.1 Soll-Ist-Zeitvergleich

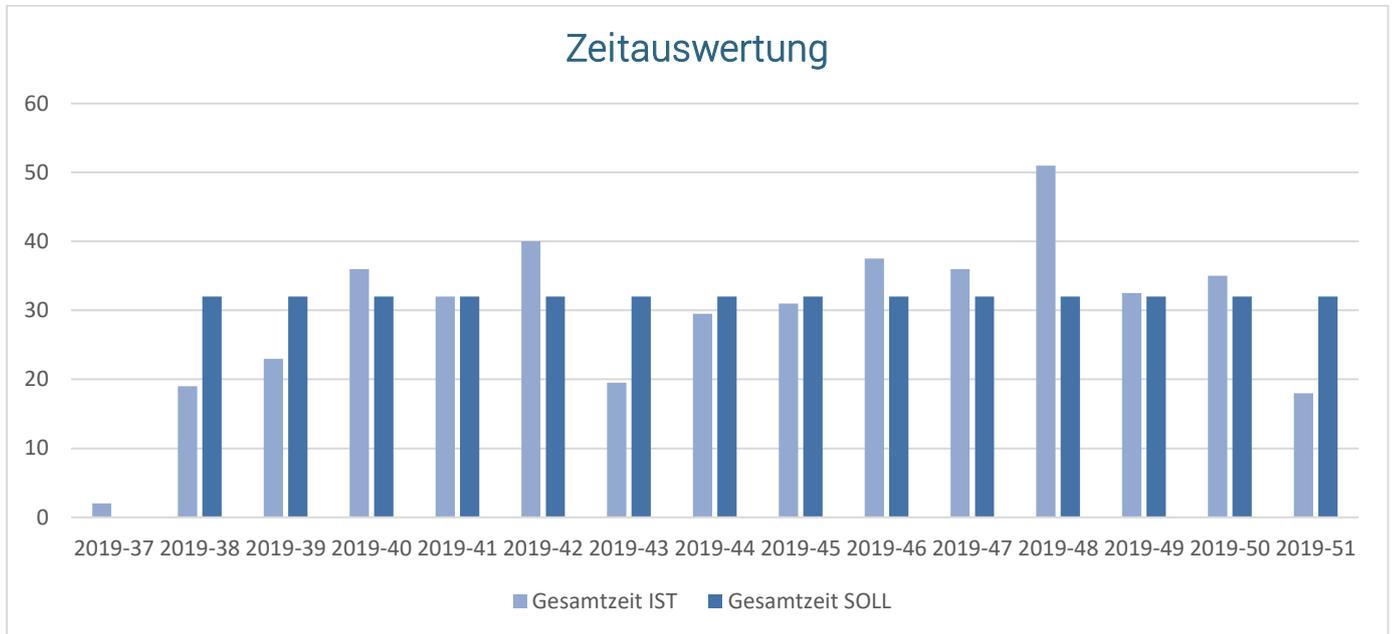


Abbildung 22 Zeitauswertung

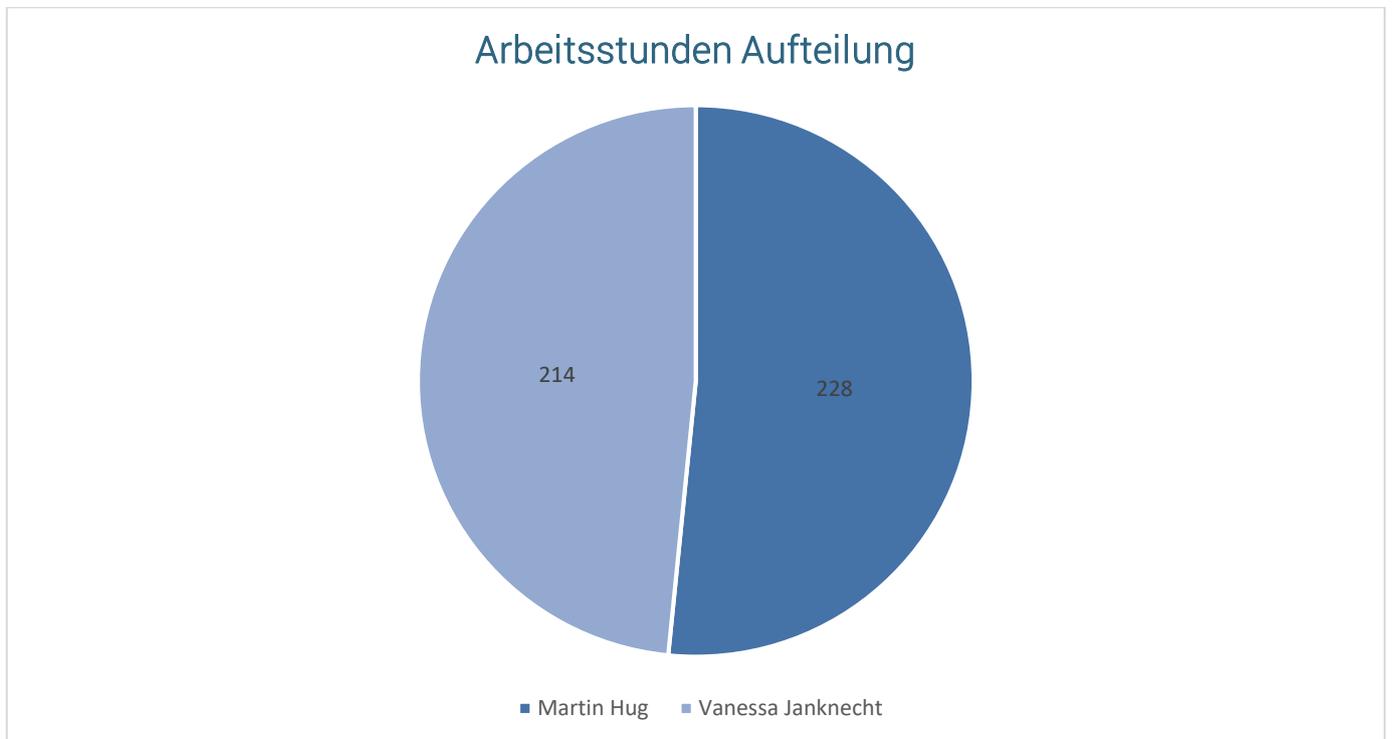


Abbildung 23 Arbeitsstunden Aufteilung

### Aufteilung nach Aktivitäten

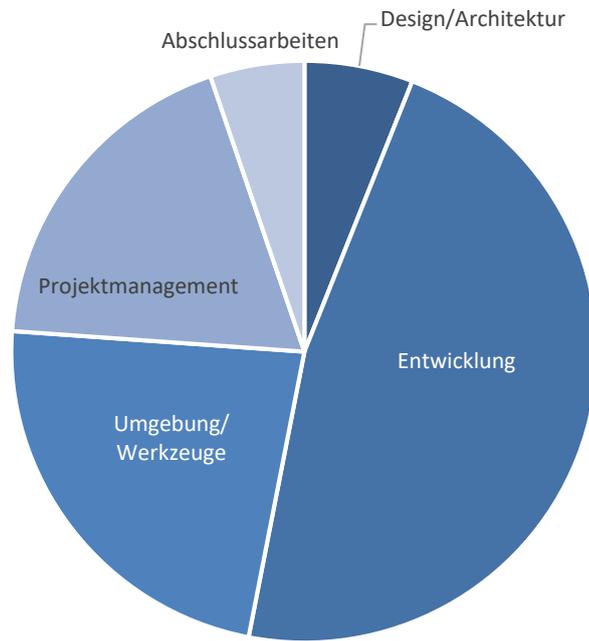


Abbildung 24 Aufteilung nach Aktivitäten

### Aufteilung nach Phasen

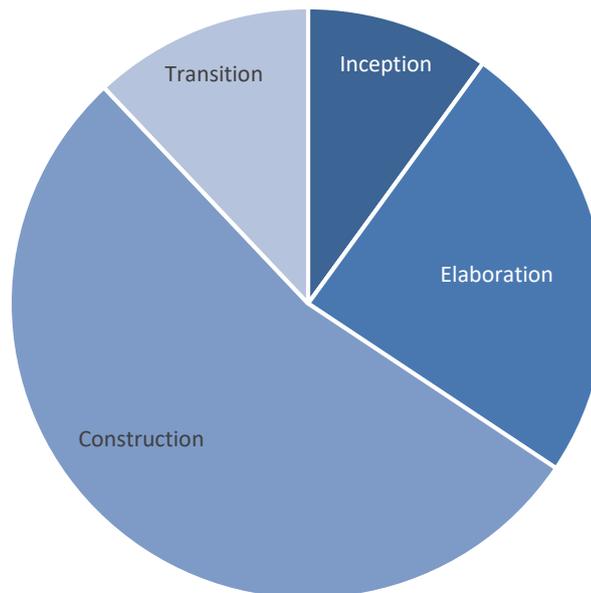


Abbildung 25 Aufteilung nach Phasen

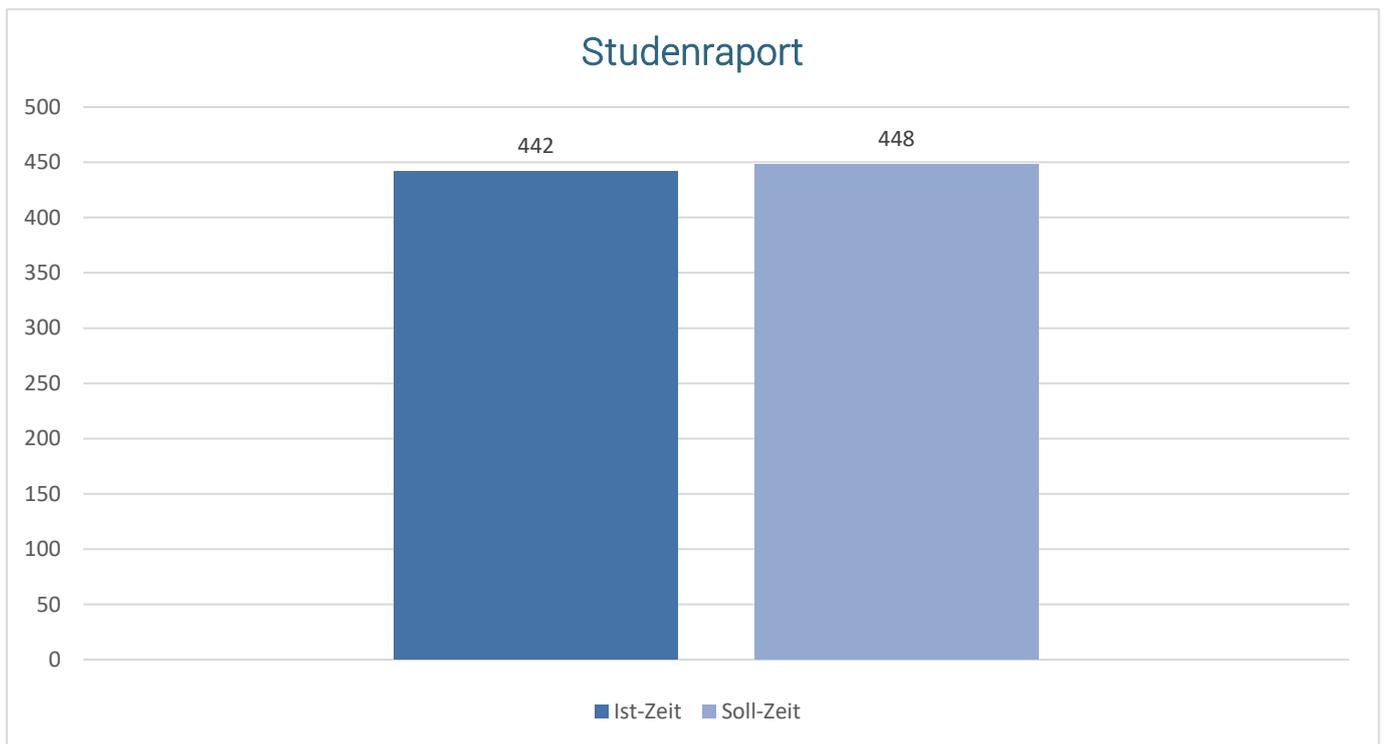


Abbildung 26 Stundenrapport

### 3.8.2 Codestatistik

Codelinien	JavaScript:	1359
	CSS:	288
	Python:	218
	HTML:	19
	Total:	1884
Kommentare	89 (4.5%)	
Klassen	14	
Funktionen	82	

Ergänzend dazu noch der generierte Sonar Cloud Report.

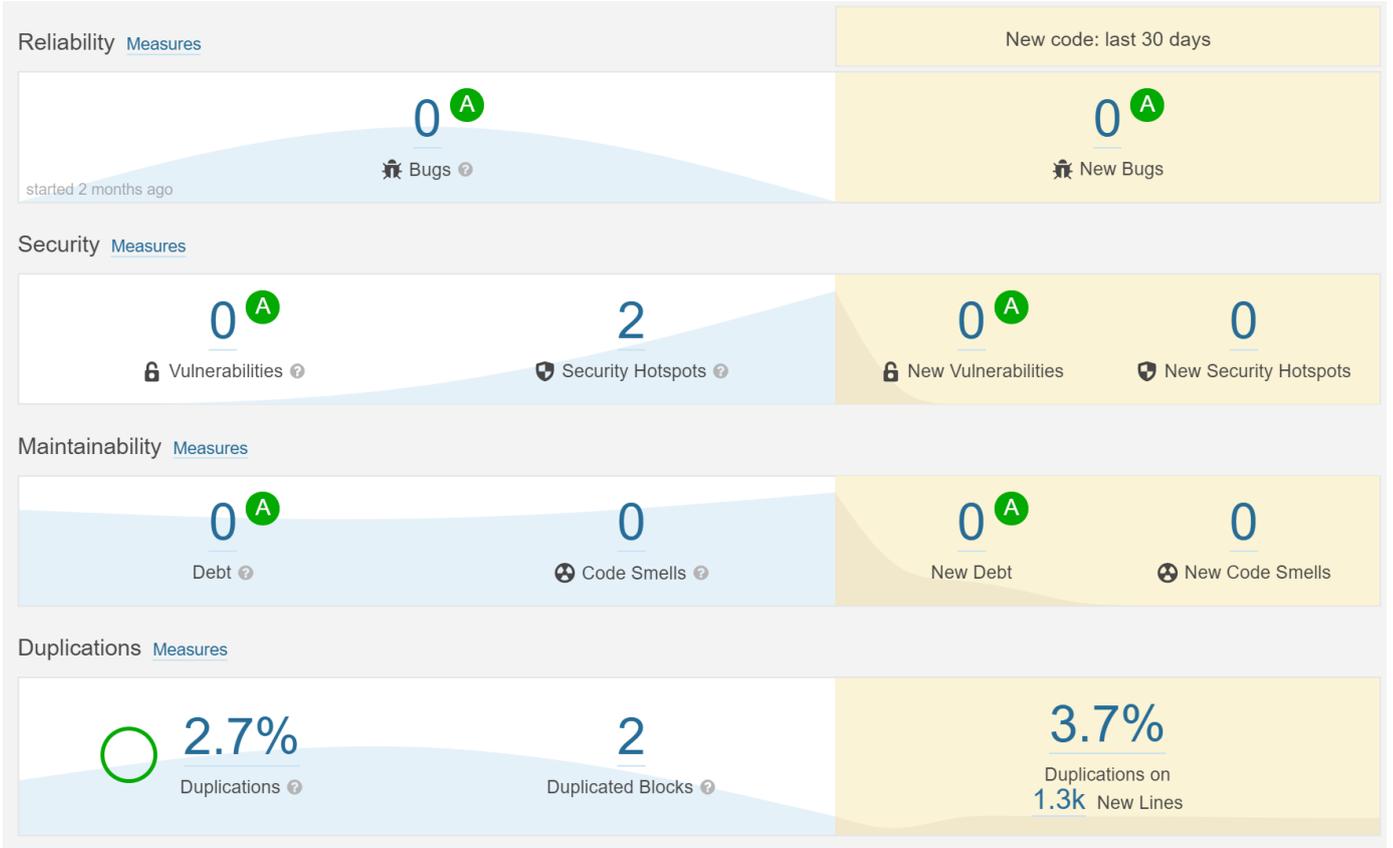


Abbildung 27 Codemetrik Sonarcloud vom 16.12.2019

## 4 Setupbeschreibung

### 4.1 Installation

Für die Installation der Applikation müssen folgende Befehle auf dem Zielsystem ausgeführt werden. Im Setup wird lediglich das Starten der Container behandelt. Sämtliche weitere Anforderungen wie beispielsweise DNS-Host Auflösung usw.

#### 4.1.1 Pull der aktuellen Docker Container

##### Backend

```
docker pull martin hug/topology-designer-backend:final
```

##### Frontend

```
docker pull martin hug/topology-designer-frontend:final
```

#### 4.1.2 Container Starten

##### Backend

```
docker run -d -p 8000:8000 --name topology-designer-backend martin hug/topology-designer-backend:final
```

##### Frontend

```
docker run -d -p 3000:3000 --name topology-designer-frontend martin hug/topology-designer-frontend:final
```

Anschliessend kann mit den gängigen Browsern auf die Applikation zugegriffen werden:

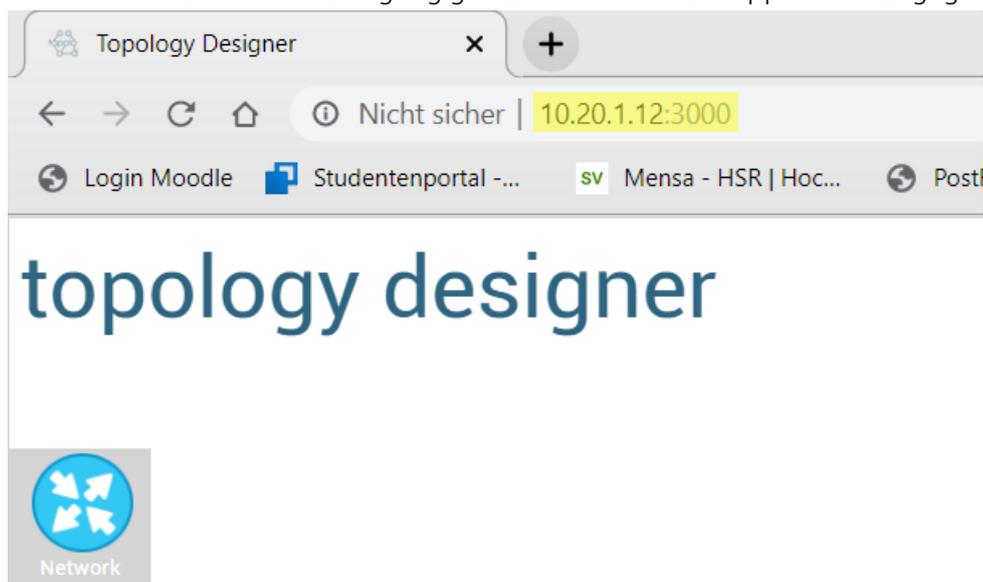


Abbildung 28 Starten des Topology Designers

## 4.2 Benutzerhandbuch

In diesem Kapitel werden die wichtigsten Funktionen kurz beschrieben.

### 4.2.1 Werkzeugeleiste

Der Topology Designer wird hauptsächlich über die Werkzeugeleiste bedient. In diesem Kapitel wird nur kurz die Funktionalität der einzelnen Buttons erläutert. Eine detailliertere Ausführung folgt in den nächsten Kapiteln.

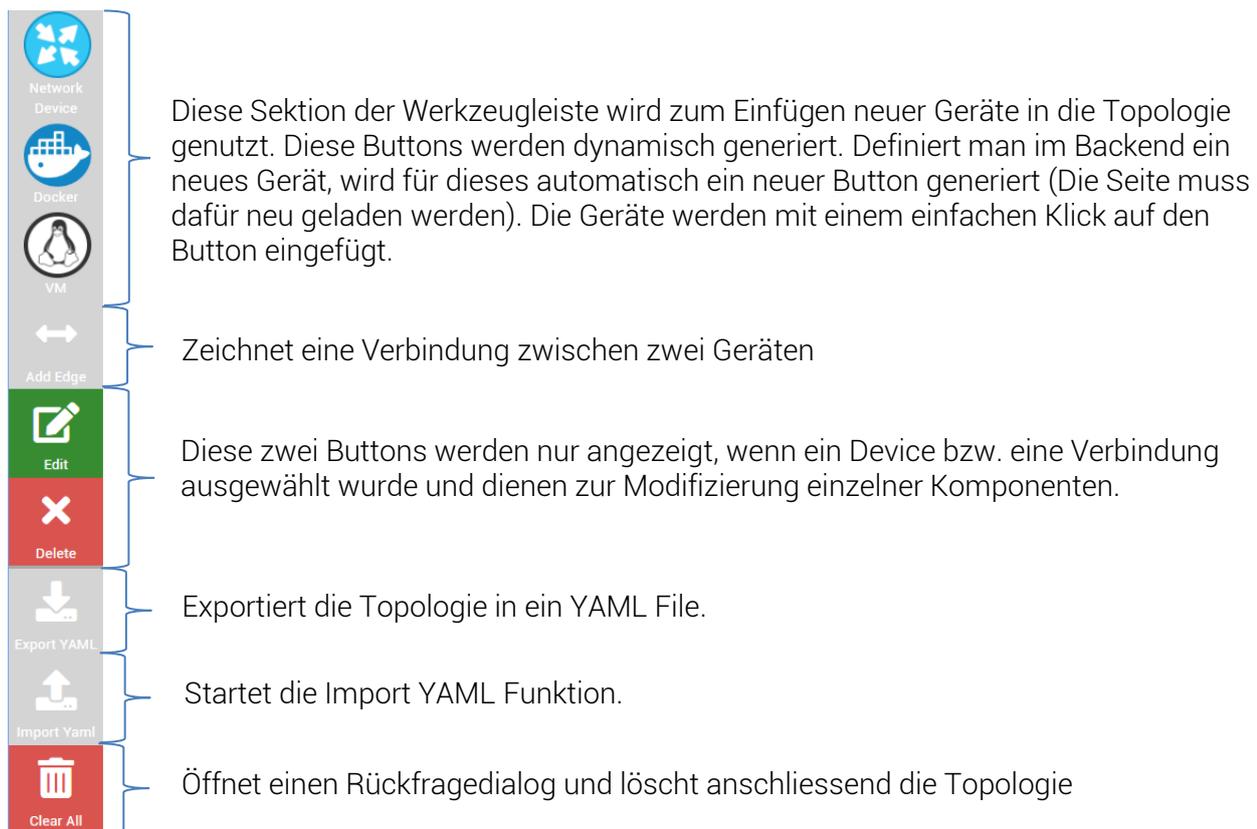


Abbildung 29 Werkzeugeleiste

### 4.2.2 Import YAML

Beim Klick auf den Button «Import YAML» öffnet sich ein Explorer Fenster, in welchem ein YAML File ausgewählt werden kann. Nach der Auswahl des Files wird die Topologie mit sämtlichen Konfigurationen importiert. Evtl. muss die Anordnung der Geräte kurz angepasst werden.

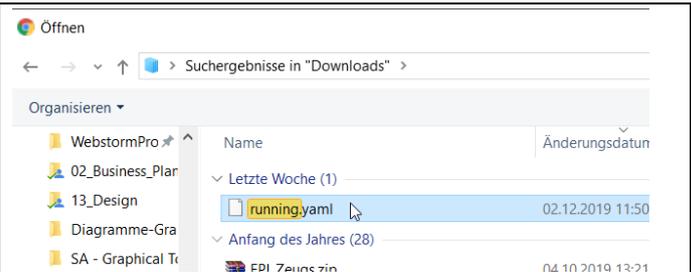
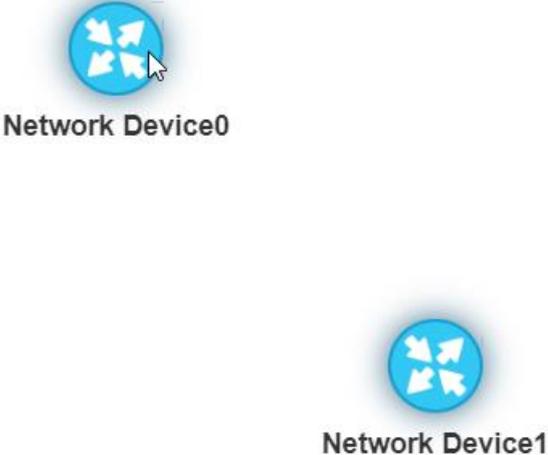


Abbildung 30 Download Folder, Ort an dem YAML File abgespeichert wird

### 4.2.3 Add Edge

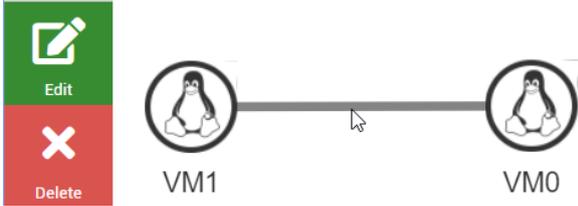
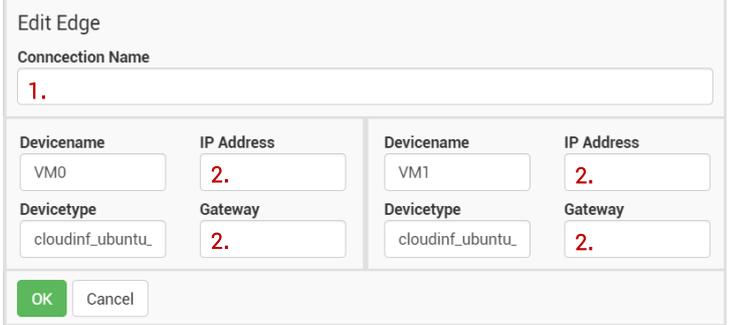
<p>Um 2 Geräte zu Verbinden müssen sie mit gedrückter CTRL Taste ausgewählt werden.</p>	 <p>Abbildung 31 Zwei angewählte Devices</p>
<p>Anschliessend wird die Verbindung bei einem Klick auf den Add Edge Button generiert.</p>	 <p>Abbildung 32 Zwei verbundene Devices</p>

### 4.2.4 Network Device Konfigurieren

<p>Die Konfiguration der Network Devices erfolgt direkt auf den Devices. Wenn ein Device ausgewählt wird, wird der Edit Button ersichtlich. Beim Klick darauf öffnet sich der Konfigurationsdialog.</p>	 <p>Abbildung 33 Virtual Network Device bearbeiten</p>
<p>Folgende Konfigurationen können vorgenommen werden:</p> <ol style="list-style-type: none"> <li>1. Device Name Hier kann der Devicename geändert werden. Wird beim Klick auf OK in der Topologie angezeigt.</li> <li>2. Device Type Der Device Type kann verändert werden, jedoch wird dieser vom Lab Topology Builder vorgegeben. Falls hier eine Fehleingabe getätigt wird, kann die Topologie nicht deployed werden.</li> <li>3. Run Config Hier wird die eigentliche Run Config für das Device definiert. Neben der Textbox wird</li> </ol>	 <p>Abbildung 34 Edit Device Dialog</p>

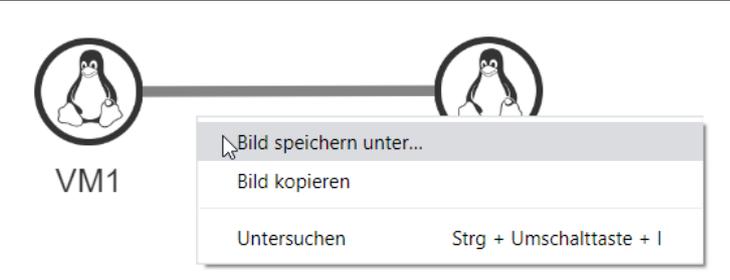
<p>angezeigt, welche Verbundenen Geräte auf welchen Ports verbunden sind.  <b>Achtung! Die Eingaben werden inhaltlich nicht geprüft. Sämtliche Fehlkonfigurationen werden einfach ins YAML File exportiert.</b></p>	
---	--

#### 4.2.5 Restliche Devices Konfigurieren

<p>Die restlichen Devices werden über die <b>Edges</b> bzw. über die <b>Verbindungen</b> konfiguriert. Beim Anwählen einer Verbindung erscheint der Edit Button.</p>	 <p>Abbildung 35 Andere Devices bearbeiten</p>
<p>Folgende Konfigurationen können vorgenommen werden:</p> <ol style="list-style-type: none"> <li>1. Connection Name          Der Verbindung kann einem Namen hinzugefügt werden. Dieser wird anschliessend in der Topologie angezeigt. (z.B. Uplink to XY) Die Namen sind jedoch <b>nur</b> im Tool ersichtlich und werden nicht ins YAML exportiert.</li> <li>2. Es kann eine IP-Adresse und ein Gateway definiert werden. Sämtliche weitführenden Konfigurationen werden nach dem Deployment via Lab Topology Builder direkt manuell auf den Geräten konfiguriert.</li> </ol>	 <p>Abbildung 36 Edit Edge Dialog</p>

#### 4.2.6 Topologie als Bild exportieren

Aufgrund eines crossorigin Problems mit dem genutzten Framework konnte der Bildexport nicht mit als Button in der Applikation realisiert werden. Deshalb funktioniert der Export mit einem Workaround.

<p>Rechtsklick auf die Topologie und «Bild speichern unter...» auswählen.</p>	 <p>Abbildung 37 Abspeichern eines Bildes</p>
---	---

## 5 *Persönlicher Rückblick*

### 5.1 *Martin Hug*

Die Studienarbeit war mein bis jetzt grösstes Software Projekt. Zuvor konnte ich im Rahmen des EPJ bereits ein wenig Software Engineering Luft schnuppern. Jedoch in einem komplett anderen Themengebiet. Vor unserer Studienarbeit war meine Webentwicklungserfahrung sehr gering. Deshalb stellte die Studienarbeit eine sehr spannende Herausforderung dar.

Die Zusammensetzung unseres Teams hat sich erst innerhalb der Semesterferien ergeben. Wir waren beide auf Teamsuche und sind im Rahmen eines WhatsApp Chats miteinander in Kontakt getreten. Zuvor haben wir uns nur sehr oberflächlich aus den Vorlesungen gekannt. Durch die eher späte Zusammensetzung unseres Teams erstellten wir unsere Stundenpläne komplett unabhängig voneinander. Zudem hatten wir beide neben der Studienarbeit eher viele Module angemeldet.

Trotzdem harmonierte die Zusammenarbeit. Wir haben uns jede Woche getroffen, Arbeitspakete definiert und klar untereinander aufgeteilt.

Meine Arbeitspakete im Projekt konzentrierten sich vor allem auf das Frontend und auf CI CD. Da wir jedoch bei jedem Pull Request gegenseitige Code Reviews durchgeführt haben, ist das Verständnis über sämtlichen geschriebenen Code vorhanden.

Im Verlaufe des Projekts durften wir grosse Fortschritte verzeichnen. Anfangs mussten wir sehr viel Zeit in die Einarbeitung der Technologien (React, Django usw.) investieren. Im Rahmen der Construction Phase konnten wir durchs Band unsere agilen Sprint Ziele erfüllen und erlitten keine grossen Rückschläge.

Die Arbeit am Projekt hat mir sehr gefallen und ich freue mich jetzt schon auf die Bachelorarbeit. Die praktische Anwendung von neuen Technologien und die Tatsache, dass wir auf ein produktiv eingesetztes Produkt hinarbeiten steigerte meine Motivation enorm. Trotz wackeligem Start bin ich überzeugt, dass wir unser Projekt mit allen Use Cases gut umsetzen konnten. Nun ist nur noch zu hoffen, dass unsere Arbeit möglichst bald im INS zum Einsatz kommen wird und dort die Lab Bereitstellung erleichtern wird.

### 5.2 *Vanessa Janknecht*

Ich habe mich sehr auf die Studienarbeit gefreut, da ich noch nie ein grösseres Software Projekt realisiert habe. Ein Projekt von Anfang bis Ende zu strukturieren, neue Technologien kennen zu lernen und die Arbeit im Team haben mich interessiert. Leider gab es zu Beginn einen starken Dämpfer, als sich zuerst niemand finden liess, der mit mir die Studienarbeit machen wollte. Ich hatte mich schon fast mit meinem Schicksal abgefunden das ganze alleine machen zu müssen, da hatte sich Martin bei mir gemeldet.

Die Arbeit im Team lief bei uns wie geschmiert. Wir haben uns während des Projekts gut verstanden und hatten keine Streitereien. Wir konnten gut miteinander diskutieren und jeder hat den Ideen des anderen geachtet. Somit konnten wir gute Lösungen für anfallende Probleme finden. Auch dass wir beide auf dem etwa gleichen technischen Stand waren war von Vorteil. So konnten wir beide viel lernen und der eine war nicht der Mitläufer vom anderen. Das einzig schwierige waren unsere Stundenpläne. Diese haben überhaupt nicht gut zusammengepasst und so hatten wir Probleme einen geeigneten Termin für unsere Besprechungen zu finden.

Mit den benutzten Technologien hatte ich bis dahin nur wenig Berührungspunkte. Ich hatte schon etwas Python programmiert und auch etwas HTML und JavaScript geschrieben aber keines davon ausführlich. Meine Aufgabe während des Projektes waren vor allem das Backend und Teile des Frontends. Da wir aber nach dem Zweiaugenprinzip arbeiteten hat jeder von uns den Codes des anderen angeschaut, kontrolliert und verstanden.

Unser Projekt lief so wie wir es geplant hatten. Die Elaborations Phase war vielleicht etwas kurz, aber wir wollten so viel Zeit wie möglich zum Programmieren haben. Da wir beide keine Erfahrung in der Webentwicklung hatten sind wir davon ausgegangen, dass wir nicht so schnell vorankommen werden. Wir konnten alle unsere Meilensteine erreichen und das Projekt wie geplant umsetzen.

Insgesamt, denke ich, war die Arbeit ein Erfolg, wir haben alle Use Cases umgesetzt und ein fertiges, funktionierendes Projekt abgeliefert und ich hoffe, dass es im INS Verwendung findet. Ich habe vieles gelernt im Verlauf des Projektes, dass mir in weiteren Projekten (z.B. die Bachelorarbeit im nächsten Semester) helfen kann.

## 6 *Literatur und Quellenverzeichnis*

**Herkunft der Vorlage** Das Dokument wurde auf der Basis einer Vorlage für Technische Berichte erstellt. Die Vorlage ist ein Element des „Werkzeugkastens Technische Berichte“ der Hochschule für Technik Rapperswil. Sie orientiert sich an Prinzipien des Strukturierten Schreibens.

- [1] E. Bystrov, «Hackernoon,» 18 Februar 2018. [Online]. Available: <https://hackernoon.com/continuous-delivery-of-react-app-with-jenkins-and-docker-8a1ae1511b86>. [Zugriff am 3 Oktober 2019].
- [2] Jenkins, «Jenkins,» [Online]. Available: <https://jenkins.io/doc/book/pipeline/docker/>. [Zugriff am 3 Oktober 2019].
- [3] React JS, «React Tutorial,» [Online]. Available: <https://reactjs.org/tutorial/tutorial.html>. [Zugriff am 17 September 2019].
- [4] vis.js, «Network Dokumentation,» vis.js, [Online]. Available: <https://visjs.github.io/vis-network/docs/network/>. [Zugriff am 10 September 2019].
- [5] axios, «axios,» [Online]. Available: <https://github.com/axios/axios>.
- [6] Django, «Django,» [Online]. Available: <https://www.djangoproject.com/>. [Zugriff am 19 September 2019].
- [7] Pallets, «Flask Documentation,» Pallets, 2010. [Online]. Available: <https://flask.palletsprojects.com/en/1.1.x/>. [Zugriff am 19 September 2019].
- [8] A. Chainz, «Django Cors Headers Source,» [Online]. Available: <https://github.com/adamchainz/django-cors-headers>. [Zugriff am 19 September 2019].
- [9] Django Rest Framework, «Django Rest Framework,» [Online]. Available: <https://www.django-rest-framework.org/#installation>.

## 7 Verzeichnisse

### 7.1 Glossar und Abkürzungsverzeichnis

<b>Build</b>	Ausführbare Programmdateien
<b>CI/CD</b>	Continuous Integration / Continuous Delivery
<b>CORS-Header</b>	«Cross-Origin Resource Sharing» wird verwendet, um HTTP REST API vor unbefugtem Zugriff durch Browserapplikationen zu schützen
<b>CSRF</b>	Cross-Site-Request-Forgery
<b>Deployment</b>	Prozess der Kompilierung und Verteilung des Programms
<b>INS</b>	Institute for Networked Solutions
<b>LTB</b>	Lab Topology Builder
<b>SQL Injection</b>	Ausnutzung von SQL-Sicherheitslücken
<b>UI</b>	User Interface
<b>XSS</b>	Cross Site Scripting
<b>YAML</b>	YAML Ain't Markup Language

### 7.2 Abbildungen

Abbildung 1 Topology Designer Logo.....	1
Abbildung 2 Use Case Diagramm.....	9
Abbildung 3 Button Beispiel.....	18
Abbildung 4 Domain Modell.....	20
Abbildung 5 Topology Designer Startscreen.....	21
Abbildung 6 Topology Designer Create Device.....	22
Abbildung 7 Topology Designer Create Connection.....	23
Abbildung 8 Topology Designer Device Configuration.....	24
Abbildung 9 Topology Designer Import Topology.....	25
Abbildung 10 Topology Designer Fertige Topologie.....	26
Abbildung 11 Topology Designer Export YAML.....	27
Abbildung 12 Topology Designer Export Image.....	28
Abbildung 13 Topology Designer Clear Topology.....	29
Abbildung 14 Sequenzdiagramm Zugriff auf Database.....	30
Abbildung 15 Workflow.....	31
Abbildung 16 Genutzte Systeme.....	32
Abbildung 17 Klassendiagramm.....	34
Abbildung 18 Deployment Diagramm.....	35
Abbildung 19 Übersicht über alle erfüllten und nicht erfüllten Use Cases.....	37
Abbildung 20 Prozessmodell.....	39
Abbildung 21 Risikomatrix.....	45

Abbildung 22 Zeitauswertung.....	47
Abbildung 23 Arbeitsstunden Aufteilung.....	47
Abbildung 24 Aufteilung nach Aktivitäten.....	48
Abbildung 25 Aufteilung nach Phasen.....	48
Abbildung 26 Stundenrapport.....	49
Abbildung 28 Codemetrik Sonarcloud vom 16.12.2019.....	50
Abbildung 29 Starten des Topology Designers.....	51
Abbildung 30 Werkzeugleiste.....	52
Abbildung 31 Download Folder, Ort an dem YAML File abgespeichert wird.....	52
Abbildung 32 Zwei angewählte Devices.....	53
Abbildung 33 Zwei verbundene Devices.....	53
Abbildung 34 Virtual Network Device bearbeiten.....	53
Abbildung 35 Edit Device Dialog.....	53
Abbildung 36 Andere Devices bearbeiten.....	54
Abbildung 37 Edit Edge Dialog.....	54
Abbildung 38 Abspeichern eines Bildes.....	54

### 7.3 Tabellen

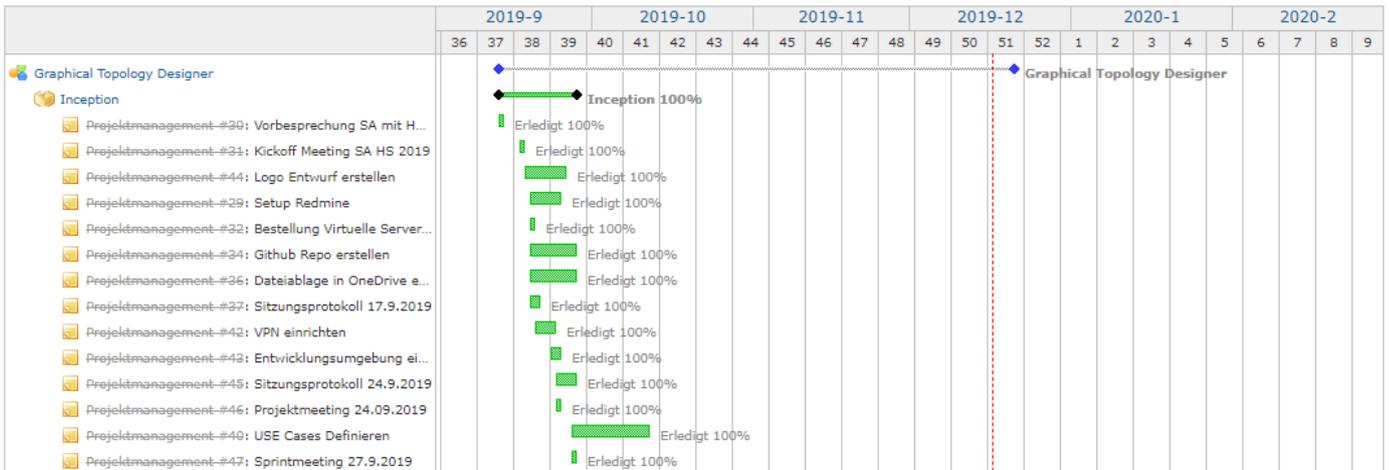
Tabelle 1 Use Cases Brief.....	10
Tabelle 2 Use Cases Casual.....	11
Tabelle 3 Use Case Fully Dressed UC04.....	13
Tabelle 4 Use Case Fully Dressed UC05.....	15
Tabelle 5 Use Case Fully Dressed UC06.....	17
Tabelle 6 Nicht Funktionale Anforderungen.....	18
Tabelle 7 Objektkatalog.....	19
Tabelle 8 Sequenzdiagramme.....	30
Tabelle 9 Frontendklassen.....	33
Tabelle 10 Backendklassen.....	33
Tabelle 11 Automatische Testverfahren.....	36
Tabelle 12 Mögliche Weiterentwicklungen.....	38
Tabelle 13 Rollen und Verantwortlichkeiten.....	40
Tabelle 14 Projektplan.....	41
Tabelle 15 Meilensteine.....	42
Tabelle 16 Risikoanalyse.....	44
Tabelle 17 Qualitätssicherung.....	46

## 8 Anhang

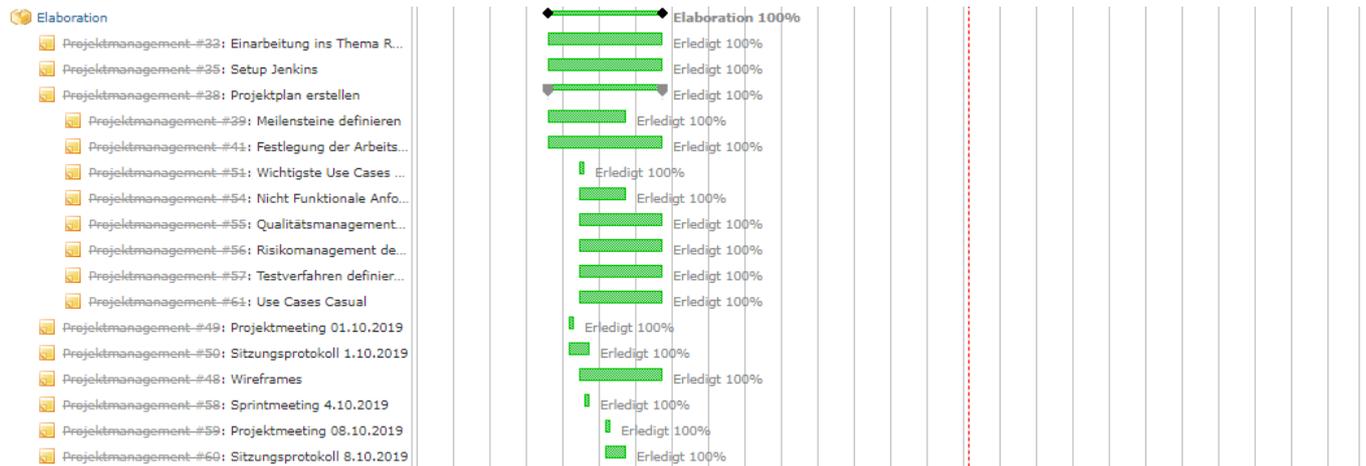
### 8.1 Redmine Auszug

Sämtliche Arbeitspakete aus Redmine aufgeteilt auf die Projektphasen:

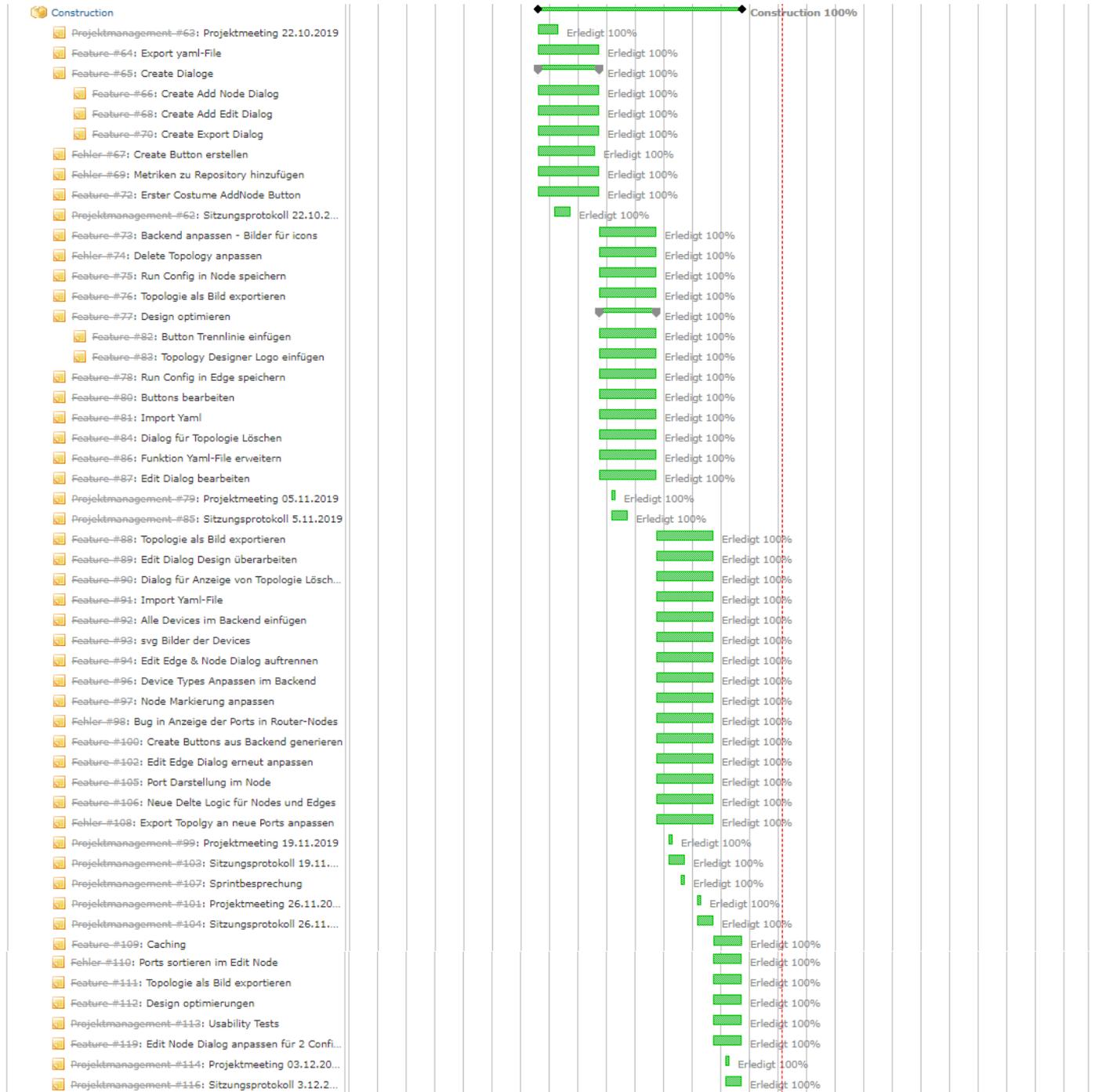
#### 8.1.1 Inception



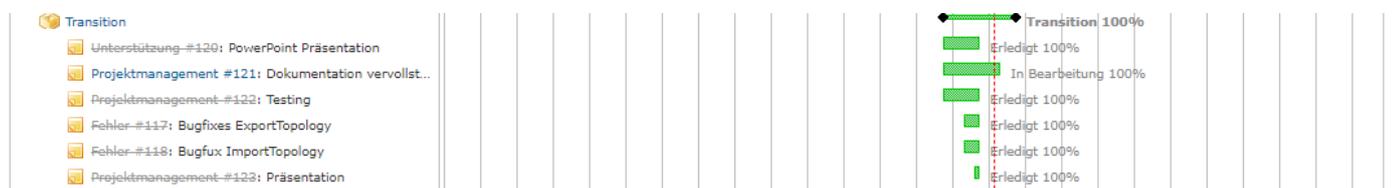
#### 8.1.2 Elaboration



### 8.1.3 Construction



### 8.1.4 Transition



## 8.2 Usability Test

### 8.2.1 Testkonzept

#### Wissensziele

- Ist das Menü selbsterklärend
- Kann der Benutzer ohne Erklärung ein neues Device einfügen?
- Ist der Benutzer in der Lage ohne Erklärung 2 Geräte miteinander zu verbinden.
- Ist der Import/ Export verständlich
- Können die Konfigurationen einfach gemacht werden.

#### Testform und -inhalt

Der Usability Test wurde bewusst nicht mit einer Person vom INS durchgeführt. Personen vom INS kennen den bestimmten Einsatz des Tools und sind ebenfalls mit der Erstellung von Topologien vertraut. Um die reine Bedienbarkeit des Tools zu testen, sind die Resultate aussagekräftiger, wenn eine objektive Person die Software testet. Wir haben deshalb innerhalb unseres Studienjahres nach einer passenden Person gesucht.

#### Testszenarien

##### *T1 Ändern des Topologienamens*

Beim ersten Start der Applikation wird eine «leere» Topologie geladen. Als erstes soll der Name der Topologie geändert werden.

# topology designer



### *T2 Geräte hinzufügen*

Es sollen folgende Geräte hinzugefügt werden:

- 1 Router
- 2 Docker
- 1 VM

### *T3 Gerät löschen*

Ein Docker Container soll wieder gelöscht werden.

### *T4 Geräte miteinander Verbinden*

Die Geräte sollen miteinander Verbunden werden.

### *T5 Geräte konfigurieren*

Folgende Konfigurationen sollen vorgenommen werden:

- Docker
  - o Namen ändern auf Container1
  - o Ip Adresse 192.168.10.10
  - o Gateway 192.168.10.1
- VM
  - o Namen ändern auf Linux1
  - o Ip Adresse 10.1.1.10
  - o Gateway 10.1.1.1
- Router
  - o Namen ändern auf R1
  - o Interface zu Docker → 192.168.10.1
  - o Interface zu VM → 10.1.1.1

### *T6 Export YAML durchführen*

Die erstellte Topologie soll in ein YAML File exportiert werden.

### *T7 Aktuelle Topologie löschen*

Sämtliche Geräte sollen gelöscht werden.

### *T8 YAML File Import*

Das zuvor importierte YAML File soll importiert werden.

## 8.2.2 Testprotokoll

Szenario Nr.	Protokoll
T1	Der Name konnte direkt geändert werden.
T2	Zuerst wollte die Testperson die Geräte in die Topologie hineinziehen. Wäre evtl. ein Feature für eine zukünftige Version
T3	Das Löschen der Geräte funktionierte ohne Probleme
T4	Das Verbinden der Geräte sorgte für ein wenig Verwirrung. Die Testperson wollte zuerst die Connections zwischen den Geräten ziehen. Es brauchte 2 Anläufe bis die Person beide Nodes ausgewählt hatte.
T5	Die Konfiguration des Routers verursachte keine Probleme. Bei den beiden anderen Geräten jedoch war der Testperson nicht klar, dass die Konfigurationen auf den Edges bzw. den Connections selbst getätigt werden müssen.
T6	Der Export funktionierte schnell und korrekt
T7	Das Löschen war ebenfalls kein Problem.
T8	Der Import funktionierte ebenfalls auf Anhieb.