# Softwareprojekt Wintersemester 2020/2021

am Fachgebiet Software Engineering, Leibniz Universität Hannover

# Spezifikation
# WB-Analyse-4

`SWP-WS2021-WB-Analyse-4-Spec-v02.pdf`

**Vorgelegt**

am 12.11.2020
von WB-Analyse-4

**Ausführende:**

| *Nachname* | *Vorname* | *Rolle* |
|---|---|---|
| Schmöcker | Robin | Projektleiter |
| Lorenz | Jasper | Qualitätsbeauftragter |
| Eberhardt | Adrian | Entwickler |
| Fardzadeh | Pedram | Entwickler |
| Schnell | Christoph | Entwickler |
| Wege | Jakob | Entwickler |

**Das Dokument enthält**

☒ Die Anforderungen aus Kundensicht (User Requirements)

☒ Anforderungen, wie das zu System zu gestalten ist (System Requirements)

---

Datum, Unterschrift des Projektleiters, auch für die anderen Projektangehörigen

**Kunden-Bewertung**

Der Kunde, Prof. Dr. Kurt Schneider, bestätigt mit seiner Unterschrift, diese Anforderungsspezifikation erhalten, geprüft und für inhaltlich ☐ **in Ordnung** | ☐ **weitgehend in Ordnung** | ☐ **deutlich zu verbessernd** | ☐ **nicht akzeptabel** befunden zu haben.

---

Datum, Unterschrift des Kunden; evtl. Vermerk.

# Inhaltsverzeichnis

# 1 Mission des Projekts

## 1.1 Erläuterung des zu lösenden Problems

Softwareentwicklung findet immer mehr von zu Hause statt, insbesondere im Rahmen der COVID-19-Pandemie. Um die Zusammenarbeit in Teams zu erleichtern und ein vertrautes Umfeld zu bieten, soll das Virtual House of Software (VirtuHoS) das Bürohaus in Form und Funktion virtuell nachbilden. Es ist hierbei wünschenswert, das Interaktionsverhalten zwischen Personen zu analysieren, sowohl aus Interessegründen als auch zur Bereitstellung von Funktionen wie der automatischen Gruppierung von Personen in Hallen abhängig von deren vorherigen Interaktionsverhalten. Das Projekt WB-Analyse soll dies mit Hilfe der sozialen Netzwerkanalyse bewerkstelligen.

## 1.2 Wünsche und Prioritäten des Kunden

Es folgen die Wünsche des Kunden, nach absteigender Priorität geordnet:

- Ausgewählte SNA-Maße sollen berechnet werden können, dies sind die folgenden:
  - Closeness Centrality
  - Eigenvector Centrality
  - Betweenness Centrality
  - Zentralisierung
- Visualisierung der SNA-Maße und Interaktionshäufigkeit. Dazu gehören visuelle Hervorhebungen von Kanten und Knoten sowie die Darstellung als Graph.

## 1.3 Domänenbeschreibung

Das VirtuoHoS wird von Softwareentwickelnden am heimischen PC genutzt. Die Analyse-Subkomponente ist jedoch rein serverseitig, da für alle Nutzer*innen die gleichen Interaktionsdaten analysiert werden sollen. Der Server wird von dem Fachgebiet Software Engineering des Instituts für praktische Informatik zur Verfügung gestellt. Zur Nutzung der Analyse-Komponente ist, wie auch für die Nutzung des Gesamtsystems, eine Internetverbindung erforderlich.

## 1.4 Maßnahmen zur Anforderungsanalyse

Die Hauptmaßnahme zur Erhebung der Anforderungen ist das wöchentliche Treffen mit dem Kunden im BigBlueButton. Dort stellen wir aktuelle Vorschläge und Zwischenstände vor, um Feedback des Kunden zu erhalten und gegebenfalls Änderungen zu diskutieren. Darüber hinaus werden Fragen, die sich über den Verlauf der letzten Woche ergeben haben, im Wochenmeeting gestellt und so die Anforderungen angepasst und präzisiert. Desweiteren nutzen wir das Forum im StudIP, um zwischen den wöchentlichen Meetings mit dem Kunden zu kommunizieren und kleinere Anfragen bezüglich der Anforderungen zu stellen.

# 2 Rahmenbedingungen und Umfeld

## 2.1 Einschränkungen und Vorgaben

Das System bietet primär eine mit den anderen VirtuHoS-Projekten vereinbarte Schnittstelle an, an die sich von beiden Seiten zu halten ist. Das Projekt soll in Java Version 15 programmiert werden. Andere Programmiersprachen dürfen nur wenn unbedingt notwendig verwendet werden, müssen dann aber mithilfe eines Wrappers oder ähnlichem versteckt werden. Das Virtual House of Software soll bis zu 160 Personen gleichzeitig unterstützen. Die dadurch anfallende große Menge an Interaktionsdaten muss verarbeitet werden können.

## 2.2 Anwender

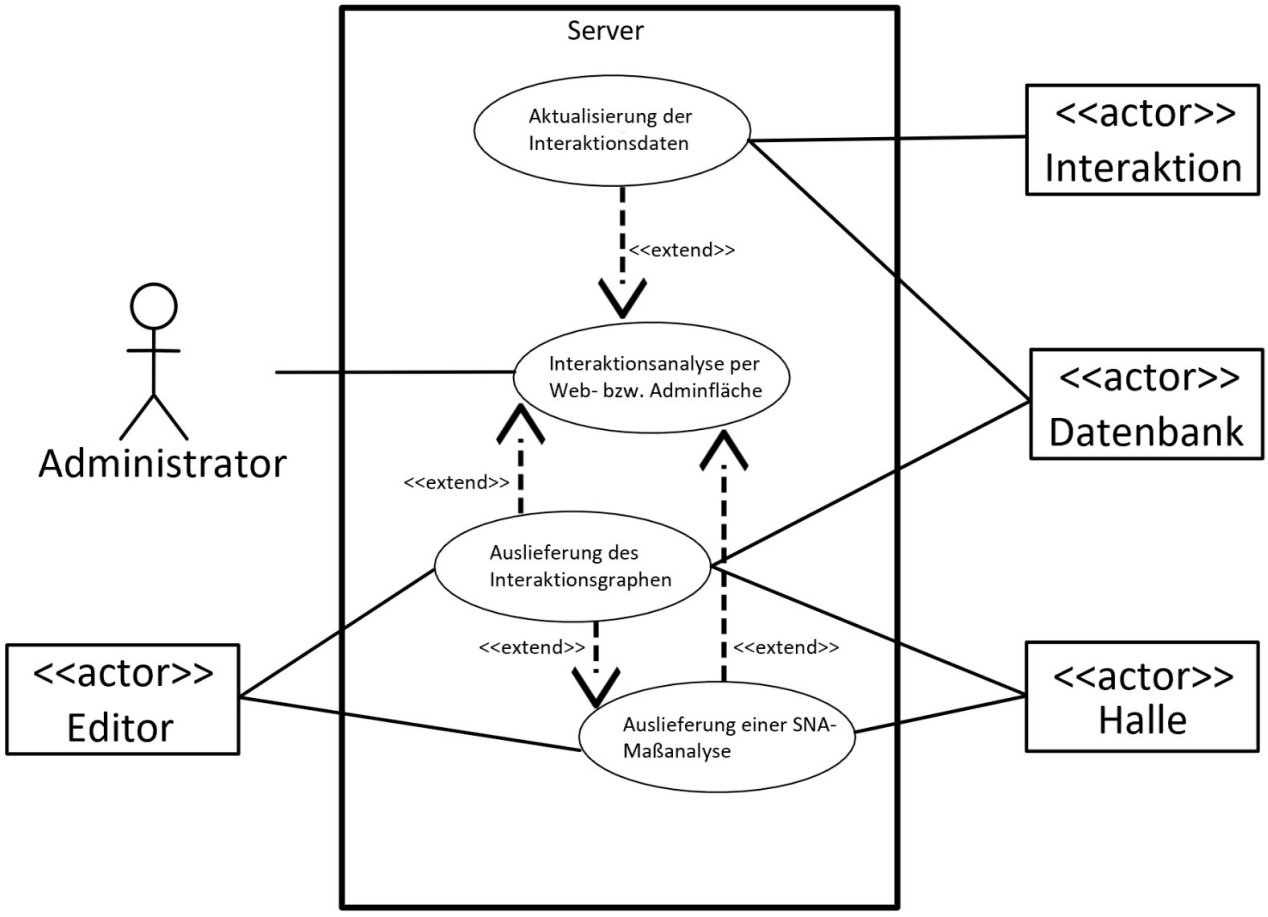
Das Gesamtsystem wird von Softwareentwickelnden am heimischen PC genutzt. Mit der Analyse-Subkomponente interagiert der/die Nutzer*in des VirtuHoS Projekts nicht direkt, sondern die anderen WB-Gruppen greifen über unsere API auf die Analysedaten zu. Zusätzlich können per Web- bzw. Adminoberfläche die Analysedaten manuell angefragt werden.

## 2.3 Schnittstellen und angrenzende Systeme

Mit dem Team der Hallekomponente existiert eine Schnittstelle zum Einholen der Analyseergebnisse zwecks Platzierung von Personen in der Halle. Diese wird mithilfe von Use Case 2 und 3 realisiert.
Mit dem Team der Editorkomponente existiert eine Schnittstelle zum Einholen der Analyseergebnisse zwecks Präsentation der Daten. Diese wird mithilfe von Use Case 2 und 3 realisiert.
Mit dem Team der Interaktionskomponente existiert eine Schnittstelle, welche die zu analysierenden Rohdaten liefert. Diese wird mithilfe von Use Case 1 realisiert.
Die ausführliche Schnittstellen Definition gibt es in Kapitel 5.2 Mögliche Anfragen an den HTTP-Server.

# 3 Funktionale Anforderungen

## 3.1 Use Case-Diagramm




**Erläuterungen und Details**

- Im Use Case-Diagramm ist der/die Enwickler*in als Akteur*in nicht zu erkennen, da dieser/diese über die Interaktion zu unserem Teilsystem verbunden ist. Ändert beispielsweise ein/eine Enwickler*in über Drag and Drop die Position, so gelangt diese Information zunächst zur Interaktion und darüber zu uns der Analyse.

## 3.2 Use Case-Beschreibungen

### 3.2.1 Aktualisierung der Interaktionsdaten

| Use Case Nr. 01 | **Aktualisierung der Interaktionsdaten** |
|---|---|
| Erläuterungen | Um die SNA-Maße und den Interaktionsgraphen zu berechnen, werden Daten benötigt. Diese Daten werden von der Interaktionsgruppe bereit gestellt. |
| Systemgrenzen (Scope) | HW für den Gesamtserver sowie Betriebssystem und gemeinsame Datenbank existieren bereits. Die Software für den Server gehört zum System. |
| Ebene | Hauptebene |
| Vorbedingung | Der Server ist erreichbar. |
| Mindestgarantie | Die vorhandenen Daten werden nicht beschädigt. |
| Erfolgsgarantie | Die neue Interaktion zweier Akteure wird in die Datenbank übernommen und eine Bestätigung wird zurück geschickt. |
| Stakeholder | - Der Interaktionsteil des Projektes wird diese Funktion nutzen um die Daten zu verschicken.<br>- Der Editor und die Halle sind auf die, aus den Daten resultierenden, Interaktionsgraphen angewiesen. |
| Hauptakteur*in | Anfragesteller*in i.d.R. die Interaktionsgruppe |
| Auslöser | Auftragsteller*in liefert die aktuellen Daten der BBB-Räume |
| Hauptszenario | 1. Auftragsteller*in liefert die aktuellen Daten der BBB-Räume<br>2. System bestätigt den Erhalt der Daten |
| Erweiterung | 2a. WENN die Daten im falschen Format sind DANN liefert der Server einen Fehler |
| Priorität | Unverzichtbar |
| Verwendungshäufigkeit | Regelmäßig |

**Erläuterungen und Details**

- Die Funktion ist sowohl für die Editorgruppe, die Interaktionsgruppe als auch für die Hallegruppe von größter Bedeutung, weswegen die Priorität als 'Unverzichtbar' gekennzeichnet ist.

### 3.2.2 Auslieferung des Interaktionsgraphen

| Use Case Nr. 02 | **Auslieferung des Interaktionsgraphen** |
|---|---|
| Erläuterungen | Der Interaktionsgraph wird vom Server berechnet, in ein passendes Format für den Editor bzw. die Halle gebracht und anschließend ausgeliefert. |
| Systemgrenzen (Scope) | HW für den Gesamtserver sowie Betriebssystem und gemeinsame Datenbank existieren bereits. Die Software für den Server gehört zum System. |
| Ebene | Hauptebene |
| Vorbedingung | Der Server ist erreichbar. |
| Mindestgarantie | Die Datenbank wird nicht verändert. |
| Erfolgsgarantie | Der korrekte Interaktionsgraph wird im passenden Format zurückgegeben. |
| Stakeholder | - Der Editor benötigt den Graphen insbesondere die Gewichtung der Kanten zwischen den Nutzer*innen, um die Strichdicke zwischen den Nutzer*innen anzuzeigen.<br>- Die Halle benötigt die Daten um die Personen in der Halle zu gruppieren. |
| Hauptakteur*in | Anfragesteller*in i.d.R. Editor bzw. Halle |
| Auslöser | Anfragesteller*in stellt eine Anfrage für den Interaktionsgraphen. |
| Hauptszenario | 1. Anfragesteller*in stellt eine Anfrage für den Interaktionsgraphen.<br>2. System berechnet den Graphen und sendet diesen dem/der Anfragsteller*in. |
| Erweiterung | 1a. WENN die Anfrage ungültig ist DANN liefert der Server einen Fehler |
| Priorität | Unverzichtbar |
| Verwendungshäufigkeit | Regelmäßig |

**Erläuterungen und Details**

- Zuerst wird die Gewichtung der Kanten nur mithilfe der Interaktionshäufigkeit bestimmt. Sollte genug Zeit sein, kann dies noch erweitert werden, um komplexere Kantengewichtungen zuzulassen. Die ausgewählte Art der Kantengewichtung kann in diesem Fall als Parameter oder explizit über eine Methode angegeben werden.

### 3.2.3 Auslieferung einer SNA-Maßanalyse

| Use Case Nr. 03 | **Auslieferung einer SNA-Maßanalyse** |
|---|---|
| Erläuterungen | Auf Anfrage wird ein eingegebener Graph hinsichtlich eines ausgewählten SNA-Maßes analysiert und das Ergebnis ausgegeben. |
| Systemgrenzen (Scope) | HW für den Gesamtserver sowie Betriebssystem und gemeinsame Datenbank existieren bereits. Die Software für den Server gehört zum System. |
| Ebene | Hauptebene |
| Vorbedingung | Der Server ist erreichbar. Der eingegebene Graph liegt im korrekten Format vor und beinhaltet gültige Werte. |
| Mindestgarantie | Die Datenbank wird nicht verändert. |
| Erfolgsgarantie | Die korrekten Analysedaten werden im passenden Format ausgeliefert. |
| Stakeholder | - Administrator*in nutzt diese Funktionalität, um das Verhalten der Nutzer*innen aus dem eingegebenem Graphen zu analysieren.<br>- Editor stellt die Avatare der Benutzer*innen ensprechend der Ergebnisse dar.<br>- Halle benötigt die Ergebnisse für eigene Funktionen. |
| Hauptakteur*in | Anfragesteller*in i.d.R. Editor bzw. Halle. |
| Auslöser | Anfragesteller*in stellt die Anfrage |
| Hauptszenario | 1. Anfragesteller*in stellt die Anfrage<br>2. System berechnet das angegebene SNA-Maß des angegebenen Graphen und sendet diesen an den/die Anfragesteller*in |
| Erweiterung | 2a. WENN die Auswahl des SNA-Maß ungültig ist DANN wird der/die Nutzer*in darauf hingewiesen. Der UC endet hier<br>2b. WENN kein Graph angegeben wurde, DANN wird ein Graph durch UC 2 (Auslieferung des Interaktionsgraphen) berechnet und darauf das angegebene SNA-Maß angewandt |
| Priorität | Unverzichtbar |
| Verwendungshäufigkeit | Regelmäßig |

**Erläuterungen und Details**

- Die verfügbaren SNA-Maße sind in Kapitel 1.2.1 erläutert. Im Verlauf der Entwicklung können optional noch weitere SNA-Maße hinzugefügt werden.

### 3.2.4 Interaktionsanalyse per Web- bzw. Adminoberfläche

| Use Case Nr. 04 | **Interaktionsanalyse per Web- bzw. Adminoberfläche** |
|---|---|
| Erläuterungen | Um auch manuell Interaktionsdaten hinzufügen zu können, sich den Graphen anzugucken und die SNA-Maße berechnen zu lassen gibt es eine Web- bzw. Adminoberfläche. |
| Systemgrenzen (Scope) | HW für den Gesamtserver, Browser und gemeinsame Datenbank existieren bereits. Web- bzw. Adminoberfläche ist Teil des Systems. |
| Ebene | Hauptebene |
| Vorbedingung | Der/Die Nutzer*in hat eine Internetverbindung und einen Webbrowser installiert. Der Analyse Server läuft. |
| Mindestgarantie | Die vorhandenen Daten werden nicht beschädigt. |
| Erfolgsgarantie | Die neuen Daten werden in die Datenbank übernommen und die angeforderten Analysedaten werden angezeigt. |
| Stakeholder | - Administrator*innen bzw. Personen, welche die SNA-Werte und Interaktionshäufigkeit einsehen wollen. |
| Hauptakteur*in | Nutzer*in i.d.R. aus der Administration |
| Auslöser | Nutzer*in öffnet die Weboberfläche im Browser |
| Hauptszenario | 1. Nutzer*in öffnet die Weboberfläche im Browser.<br>2. System antwortet und zeigt die Weboberfläche an.<br>3. Nutzer*in gibt zwei Nutzer*in an zwischen denen interagiert wurde und übermittelt die Daten.<br>4. System führt UC 1 (Aktualisierung der Interaktionsdaten) aus und antwortet mit einer Erfolgsmeldung.<br>5. Nutzer*in bestätigt die Option den Graph anzeigen zu lassen.<br>6. System antwortet mit dem Graphen welcher mit UC 2 (Auslieferung des Interaktionsgraphen) generiert wurde.<br>7. Nutzer*in wählt ein SNA Maß startet die Berechnung.<br>8. Ergebnis, welches mit UC 3 (Auslieferung einer SNA-Maßanalyse) berechnet wurde wird angezeigt. |
| Erweiterung | 3a. WENN der/die Nutzer*in keine zusätzlichen Daten eingeben will DANN kann er/sie diesen Schritt überspringen. Weiter mit 5.<br>5a. WENN der/die Nutzer*in den Graphen nicht angezeigt bekommen möchte DANN kann er/sie diesen Schritt überspringen. Weiter mit 7.<br>7a. WENN der/die Nutzer*in keine SNA durchführen möchte DANN kann er/sie diesen Schritt überspringen. Der UC endet hier. |
| Priorität | Optional |
| Verwendungshäufigkeit | Selten |

**Erläuterungen und Details**

- Ein Mockup der Web- bzw. Adminoberfläche befindet sich in Kapitel 5.3.

# 4 Qualitätsanforderungen

## 4.1 Qualitätsziele des Projekts

Allgemein gesagt ist das Ziel des Projektes einen HTTP-Server bereitzustellen, welcher per HTTP-Anfrage eine angelegte Datenbank mit personenbezogenen Daten des VirtuHoS befüllt und diese durch eine Soziale-Netzwerkanalyse ausgewertet zurückliefern kann. Die Daten beschreiben allgemein gesagt, welche Personen innerhalb des VirtuHoS wie (viel) miteinander interagieren.
Sicherzustellen ist dabei, dass das Auswerten der Daten in Echtzeit geschehen soll und die Auswertung somit performant implementiert werden muss.
Zudem sollte der HTTP-Server invalide Anfragen bearbeiten könnnen. So zum Beispiel, wenn eine ungültige Anfrage-Methode für die Routen verwendet wird oder unpassende Paramter bei der Anfrage mitgeschickt werden. Dies dient der Robustheit unsere Projektes.

## 4.2 Qualitäts-Prioritäten des Kunden

Die Qualitätsziele sind allgemein wie folgt absteigend priorisiert:

- Einfachheit
- Robustheit

Ein zusätzliches Qualitätsziel wäre Sicherheit. Dabei würde vorallem wichtig sein, dass personenbezogenen Daten sicher sind. Genau so wichtig wäre, dass keine falschen Daten von unautorisierten zur Datenbank hinzugefügt werden und die Netzwerkanalyse-Ergebnisse verfälscht werden. Ersteres, da dies reale datenschutzrechtliche Konsequenzen zur Folge haben könnte und zweiteres, da damit der komplette Zweck unseres Projektes entfallen würde.

## 4.3 Wie Qualitätsziele erreicht werden sollen

Die primären Qualitätsziele (Einfachheit, Robustheit) zu erreichen, geht Hand in Hand, da einfache Systeme tendenziell auch die robusteren sind.
Durch die Sicherstellung, dass unser HTTP-Server alle Anfragen auf seinen verfügbaren Routen beantwortet, kann der/die Benutzer*in unserer API sicher gehen, dass stets eine Antwort vom Server geschickt wird.
Um die Benutzung der API simpel zu halten, stellen wir Beispielcode für die anderen VirtuHoS Komponenten zur Verfügung, welcher zeigt, wie ein Anfrage an unseren Server gestellt wird und wie die erhaltene Anfrage zu bearbeiten ist. Ferner wird dadurch sichergestellt, dass keine fehlerhaften Anfragen an unseren Server geschickt werden.
Die Robustheit wird intern bei der Netzwerkanalyse durch ausführliche Testfälle sichergestellt, sodass der Kunde sicher sein kann, dass die Netzwerkanalyse richtige Ergebnisse liefern wird.

# 5 Hinweise zur Umsetzung

Der Webserver wird mit Java programmiert.

## 5.1 Beispiel für grafische Ausgabe

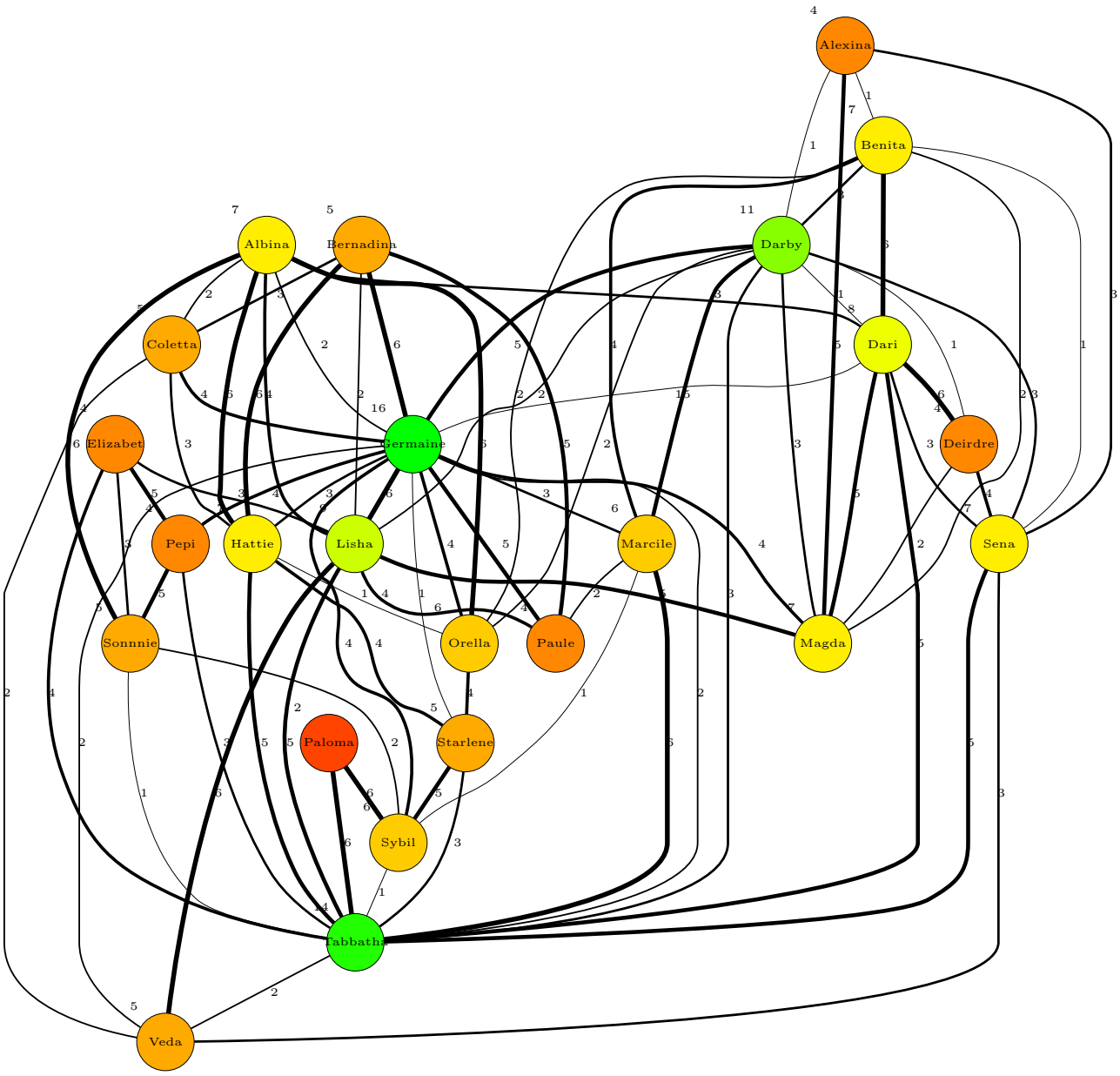



Abbildung 1: Beispiel für visuelle Repräsentation der Interaktionsdaten

Wird ein Graph von der Web- bzw. Adminoberfläche angefordert, so kann dieser wie in Abbildung 1 dargestellt werden. Die Nutzer*innen des VirtuHoS sind als Kreise dargestellt. Sind zwei Nutzer*innen durch eine Linie verbunden, so haben sie mindestens einmal kommuniziert. Die Kantendicke repräsentiert die Interaktionshäufigkeit. Je dicker die Kante, desto öfter wurde interagiert. Die Farbe der Nutzer repräsentiert den errechneten Wert des SNA Maßes. Ein kleiner Wert wird Rot, ein großer Grün angezeigt. Alle Werte dazwischen werden in einem Farbverlauf zwischen Rot über Gelb nach Grün dargestellt. In diesem Beispielbild wurde als SNA Maß "Gradzentralität"verwendet. Auch wenn dieses nicht

implementiert werden soll, dient es hier als anschaulicheres Beispiel als bspw. die “Eigenvektorzentralität".
Um das Prinzip des Graphen noch anschaulicher zu machen betrachten wir nun die Nutzerin Veda (ganz unten Links). Man sieht, dass dieser Nutzerin ein Wert von 5 durch die SNA zugewiesen wurde. Dieser ist eher niedrig im Vergleich zu anderen Nutzer*innen (min: 2 max: 14), daher der Farbton Orange. Orange ist ein Farbton zwischen Gelb und Rot. Die Kante zu Lisha ist sehr dick, da diese Nutzerinnen bereits oft interagiert haben. Dagegen ist die Kante zu Coletta dünner, da sie nur zwei Mal interagiert haben. Zusätzlich zu Lisha und Coletta hat Veda bereits mit Germaine, Tabbatha und Sena ineragiert.

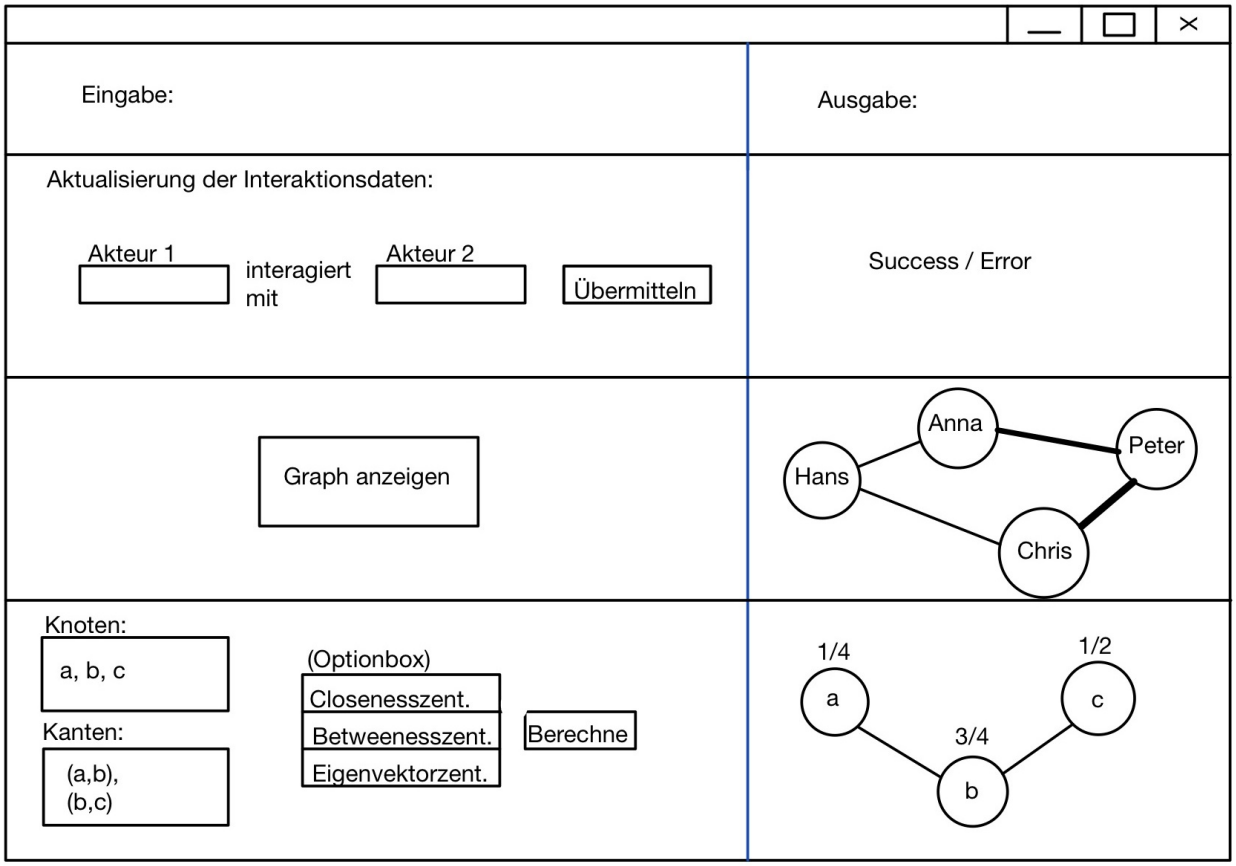
## 5.2 Mögliche Anfragen an den HTTP-Server

Es sei anzumerken, dass die Routen hier exemplarisch gewählt sind und nicht den finalen Routen entsprechen müssen. Parameter werden zunächst wie in den Use-Cases angegeben implementiert. Die hier zusätzlich erwähnten Parameter sind als optionale Erweiterungen zu betrachten.

| **Anfragetyp** | **Route** | **Parameter** | **Erfolgsrückgabe** | **Fehlerrückgabe** |
|---|---|---|---|---|
| POST | /api/interaction | Ein oder mehrere Tupel, die die Interaktion zwischen zwei Personen beschreibt | Status 200, Die Daten wurden in Datenbank geschrieben | Status 500, Intern ist ein Fehler passiert und die Daten konnten nicht in die Datenbank hinzugefügt |
| GET | /api/graph | Das Format in welchem der Graph zurückgegeben werden soll (z.B. als SVG oder JSON). Zusätzliche Parameter könnten optional sagen, aus welchen Interaktionsdaten der Graph zusammengestellt werden soll | Status 200, Es wird ein Graph in entsprechender Codierung zurückgegeben. | Status 500 Es wird kein Graph zurückgegeben, da intern ein Fehler passiert ist. |
| GET | /api/network-analysis | Der Analyse-Typ gibt an, auf welches Maß analysiert werden soll. Zudem kann ein Graph mitgeschickt werden. Wird ein Graph (entsprechend codiert) mitgeschickt, so wird diese statt den Daten in der Datenbank analysiert. | Status 200, die Analyse liefert die korrekten Werte für alle Knoten im Netzwerk zurück. | Status 500, es werden keine SNA-Werte zurückgegeben, da ein interner Fehler passiert ist. |

## 5.3 Mockup Web- bzw. Adminoberfläche

Im Folgenden ist ein Mockup der Adminoberfläche im Browser zu sehen, über welches manuell die Analyse gesteuert werden kann. Die Boxen "Übermitteln", "Graph anzeigen“ und "Berechne“ sind Buttons und alle restlichen zu sehenden Boxen (wenn nicht anders gekennzeichnet) sind Eingabefelder. Auf der rechten Seite ist nur die visuelle Ausgabe zu sehen, hiermit ist keine Interaktion vorgesehen.




## 5.4 Zu implementierende Analyse Verfahren

Um deutlich festzulegen, welche Analysen implementiert werden, sind im Folgenden die zu implementierenden Maße eindeutig definiert.

Die **Interaktionshäufigkeit** definiert das Gewicht der Kanten innerhalb des Interaktionsgraphen. Die Centralities werden im folgenden Abschnitt genau definiert.

Die folgenden drei Formeln beziehen sich alle auf einen ungerichteten, ungewichteten Graphen $G = (V, E)$.

**Closeness Centrality:**
Die Closeness centrality eines Knoten $v \in V$ ist definiert als

$$\frac{|V| - 1}{\sum_{s \in V} d(s, v)}$$

**Betweenness Centrality:**
Um die Formel mathematisch präzise formulieren zu können, werden hier zunächst ein paar Hilfsdefinitionen eingeführt.

$paths(s,t) := \{(a_0, ..., a_n) \mid a_0 = s \wedge a_n = t \wedge n \in \mathbb{N}_0 \wedge \forall i \in \{0, ..., n-1\} : (a_i, a_{i+1}) \in E\}$
$length(a_0, ..., a_{n-1}) := n$, wobei $(a_0, ..., a_{n-1}) \in paths(a_0, a_{n-1})$
$:= mpl(s,t) := min\{length(p) \mid p \in paths(s,t)\}$
$M(s,t) := \{(s,t) \mid s \neq v \wedge t \neq v\}$

$\sigma_{st} := |\{p \in paths(s,t) \mid length(p) = mpl(s,t)\}|$
$\sigma_{st}(v) := |\{p = (a_0, ..., a_n) \in paths(s,t) \mid length(p) = mpl(s,t) \wedge \exists k \in \{0, ..., n\} : v = a_k\}|$

SDie Betweenness Centrality eines Knoten $v \in V$ ist nun definiert als

$$\sum_{(s,t) \in M(s,t)} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

**Eigenvector Centrality:**

Sei die Knotenmenge $V = \{v_0, ..., v_n\}$ und damit $A \in \mathbb{N}_0^{n \times n}$, mit $A_{ij} = 1$, wenn $(v_i, v_j) \in E$ und 0 sonst, die Adjazenzmatrix von G.
Außerdem sei X die Menge der Eigenvektoren der Eigenwertgleichungen $Ax = \lambda x$, wobei $\lambda \in Spec(A)$.
Sei $X_+ := \{x = (x_0, ..., x_n) \in X \mid \forall i \in \{0, ..., n\} : 0 \leq x_n\}$
Aus dem Perron–Frobenius Theorem lässt sich folgern, dass gilt $X_+ \neq \emptyset \wedge \forall x, y \in X_+ : \frac{x}{|x|} = \frac{y}{|y|}$
Daher ist die Eigenvector Centrality des Knoten $v_i \in V$ dann wohldefiniert als

$$(\frac{x}{|x|})_i$$

, wobei $x \in X_+$ beliebig gewählt werden kann.

# 6 Probleme und Risiken

Im Folgenden die wichtigsten Probleme und Risiken des Projektes, geordnet nach Risk Exposure. Die Abhilfe zu Risiko i ist in Punkt i der Abhilfe, einen Absatz tiefer beschrieben.

1. WENN die Schnittstellen zum Abrufen der Analysedaten und zum Anfragen der Interaktionen nicht funktionieren oder unzureichend dokumentiert sind
   DANN kann das Projekt nicht mit den anderen Projekten des VirtuHoS interagieren.
   KONSEQUENZ: Wir können nicht mit den anderen Modulen interagieren und müssen auf unsere (Not)-Oberfläche zurückgreifen.
   Wahrscheinlichkeit: L, Schaden: M

2. WENN die Analysealgorithmen nicht ausreichend getestet werden
   DANN liefert die Analyse möglicherweise falsche Ergebnisse.
   KONSEQUENZ: Irreführende visuelle Darstellung und falsche Paarungen von Personen.
   Wahrscheinlichkeit: M, Schaden: M

3. WENN bei den Anfragen auf die Datenbank nicht auf Sicherheit geachtet wird
   DANN könnten Unbefugte Daten lesen oder auf die Datenbank schreiben und dort einen Totalschaden anrichten.
   KONSEQUENZ: Die anderen Komponenten des VirtuHoS können nicht mehr auf die Analyse zugreifen.
   Wahrscheinlichkeit: L, Schaden: M

4. WENN die Analysealgorithmen ineffizient implementiert werden
   DANN kann es zu Laufzeitproblemen bei größeren Personenmengen kommen.
   KONSEQUENZ: Das Liefern der Analyseergebnisse verzögert sich und damit auch die Darstellung dieser.
   Wahrscheinlichkeit: H, Schaden: L

5. WENN die Analysealgorithmen unzureichend dokumentiert und unflexibel sind
   DANN ist es schwierig bei neuen Kundenwünschen das Programm um neue Formeln zu erweitern.
   KONSEQUENZ: Der Kunde ist unzufrieden.
   Wahrscheinlichkeit: L, Schaden: L

**Abhilfe:**

1. Früh und intensiv mit den Programmierenden der anderen Komponenten des VirtuHoS auseinandersetzen und über die Schnittstellen reden. Außerdem Fokus auf die Dokumentation dieser legen.

2. Ständiges Testen und Unit Testing.

3. Anfragen auf Datenbank verschlüsseln und wenn möglich Prepared Statements benutzen. Mögliche Angreifer*innen auf rechtliche Folgen aufmerksam machen.

4. Mentalität für Effizienz schaffen und theoretische Laufzeitanalysen vor der Implementierung durchführen. Zur Not Laufzeitbegrenzungen einführen und eine heuristische Lösung liefern.

5. Großen Wert auf Codedokumentation legen und ständiges Überprüfen, ob dies auch durchgeführt wird. Außerdem vor der Implementierung Gedanken über eine flexible Softwarearchitektur machen.

# 7 Optionen zur Aufwandsreduktion

## 7.1 Mögliche Abstriche

- Anstatt das Netzwerk nach 3 Kriterien zu analysieren, nur nach Closeness und Betweenness analysieren.
- Auf Funktionalität setzten, statt sich lange mit effizienter Implementierung aufzuhalten.
- Weniger Dokumentation bei nicht kritischem Code.
- Web- bzw. Adminoberfläche nicht interaktiv gestalten und auf Ästhetik verzichten.

## 7.2 Inkrementelle Arbeit

Am wichtigsten ist das Aufsetzen des Webservice, so dass die anderen Komponenten bereits unsere Schnittstelle benutzen können.
Danach werden geordnet nach Priorität die Algorithmen zur Netzwerkanalyse implementiert. Nacheinander, da der Kunde lieber einen funktionierenden Algorithmus hat, anstatt 3, die nur zur Hälfte fertig sind und gar nicht funktionieren.
Zum Schluss können dann noch zusätzliche Wünsche des Kunden implementiert werden, wie zum Beispiel weitere Analysealgorithmen oder, dass die personenbezogenen Daten vor Angreifer*innen besser geschützt werden.

# 8 Glossar

**Graph** Mathematisches Modell eines Netzwerks.

**Soziale Netzwerkanalyse (SNA)** Eine Sammlung von Verfahren zum Analysieren sozialer Netzwerke mithilfe der Graphentheorie.

**Zentralität** Eine Familie von Maßen für die Wichtigkeit eines Knotens in einem Graphen. Verschiedene Zentralitätsmaße definieren die Wichtigkeit auf unterschiedliche Weise.

- **Eigenvektor-Zentralität (Eigenvector Centrality)** Ein Zentralitätsmaß, welches die Wichtigkeit anhand von Verbindungen zu anderen wichtigen Knoten definiert.
- **Nähezentralität (Closeness Centrality)** Ein Zentralitätsmaß, welches die Wichtigkeit anhand der Nähe zu anderen Knoten im Netzwerk definiert.
- **Zwischenzentralität (Betweenness Centrality)** Ein Zentralitätsmaß, welches die Wichtigkeit anhand der Anzahl der kürzesten Pfade im Netzwerk, die durch diesen Knoten gehen, definiert.

**Zentralisierung** Ein Maß für die Zentralität des zentralsten Knotens im Vergleich zu den anderen Knoten in einem Netzwerk. Die Zentralisierung hängt vom verwendeten Zentralitätsmaß ab.

# 9 Abnahme-Testfälle

Hinweis: Der folgende Graph wird in den Abnahmetestfällen referenziert.

## 9.1 Setup

Zum Beginnen der Abnahme-Testfälle muss die Web- bzw. Adminoberfläche unserer API im Browser geöffnet werden.
Damit sich die Web- bzw. Adminoberfläche öffnen lässt, muss vorher der HTTP-Server und somit unsere API gestartet worden sein. Dabei muss die Datenbank leer sein. Diese wird im Anschluss im ersten Test mit Interaktionsdaten befüllt. Auf der nun gefüllten Datenbank werden nun die weiteren Testfälle ausgeführt.
Die Testfälle werden als erfolgreich annerkannt, falls die berechneten Maße mit einer Genauigkeit von $\pm 0.1\%$ berechnet wurden. Es kann kleine Abweichungen geben, da nicht alle Fließkommazahlen mit beliebiger Genauigkeit von dem Computer dargestellt werden können.
Der folgende ungerichtete, gewichtete Graph $G = (V, E)$ mit
$V = \{a1, b1, c1, d1, a2, b2, c2, d2, e2\}$ und E wie in Abbildung 2 wird für die Abnahmetestfälle verwendet. Dabei werden zunächst die zugrundeliegenden Interaktionsdaten zur Datenbank hinzugefügt und der resultierende Graph auf SNA-Maße analysiert.

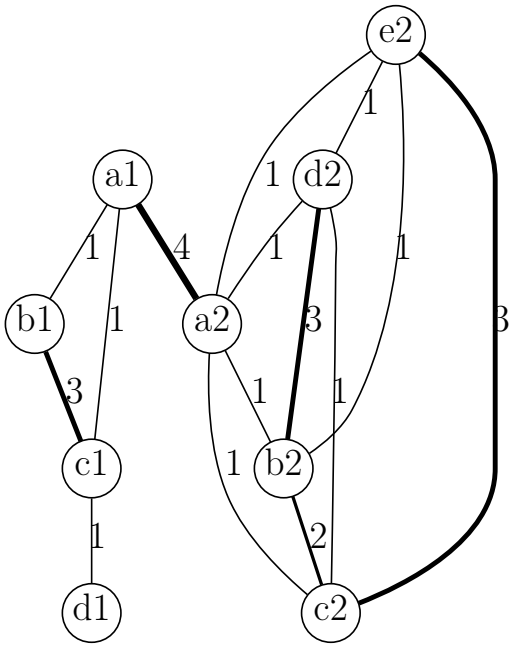



Abbildung 2: Test Graph G

## 9.2 Testfälle

### 9.2.1 Interaktionsdaten eingeben und Graph generieren lassen

Grundlage für diesen Teiltestfall ist der in 9.1 gezeigte Graph. Dabei werden zunächst die zugrundeliegenden Interaktionsdaten zur Datenbank hinzugefügt und der resultierende Graph generiert. Davor wurden noch keine Interaktionen an den Server gemeldet.

| **Eingabe** | **Soll-Ausgabe** |
|---|---|
| Interaktionskanten hinzufügen | |
| Melde an Server: a1 interagiert mit b1 | Bestätigung vom Server |
| Melde an Server: b1 interagiert mit c1 | Bestätigung vom Server |
| Melde an Server: a1 interagiert mit a2 | Bestätigung vom Server |
| Melde an Server: a1 interagiert mit c1 | Bestätigung vom Server |
| Melde an Server: a2 interagiert mit b2 | Bestätigung vom Server |
| Melde an Server: a2 interagiert mit c2 | Bestätigung vom Server |
| Melde an Server: d2 interagiert mit b2 | Bestätigung vom Server |
| Melde an Server: e2 interagiert mit c2 | Bestätigung vom Server |
| Melde an Server: b2 interagiert mit c2 | Bestätigung vom Server |
| Melde an Server: a1 interagiert mit a2 | Bestätigung vom Server |
| Melde an Server: c1 interagiert mit d1 | Bestätigung vom Server |
| Melde an Server: b1 interagiert mit c1 | Bestätigung vom Server |
| Melde an Server: d2 interagiert mit b2 | Bestätigung vom Server |
| Melde an Server: e2 interagiert mit c2 | Bestätigung vom Server |
| Melde an Server: a1 interagiert mit a2 | Bestätigung vom Server |
| Melde an Server: d2 interagiert mit a2 | Bestätigung vom Server |
| Melde an Server: b2 interagiert mit c2 | Bestätigung vom Server |
| Melde an Server: d2 interagiert mit c2 | Bestätigung vom Server |
| Melde an Server: b1 interagiert mit c1 | Bestätigung vom Server |
| Melde an Server: e2 interagiert mit a2 | Bestätigung vom Server |
| Melde an Server: e2 interagiert mit b2 | Bestätigung vom Server |
| Melde an Server: a1 interagiert mit a2 | Bestätigung vom Server |
| Melde an Server: d2 interagiert mit b2 | Bestätigung vom Server |
| Melde an Server: e2 interagiert mit c2 | Bestätigung vom Server |
| Melde an Server: e2 interagiert mit d2 | Bestätigung vom Server |
| Es wird der Graph vom Server unter /api/graph angefordert | Graph G wird entsprechend codiert als gewichteter Graph zurück gegeben: {"a1": [{"id": "a2", "weight": 4}, {"id": "b1", "weight": 1}, {"id": "c1", "weight": 1}], "b2": [{"id": "a2", "weight": 1}, {"id": "c2", "weight": 2}, {"id": "d2", "weight": 3}, {"id": "e2", "weight": 1}], "a2": [{"id": "b2", "weight": 1}, {"id": "a1", "weight": 4}, {"id": "d2", "weight": 1}, {"id": "e2", "weight": 1}, {"id": "c2", "weight": 1}], "d1": [{"id": "c1", "weight": 1}], "e2": [{"id": "a2", "weight": 1}, {"id": "b2", "weight": 1}, {"id": "c2", "weight": 3}, {"id": "d2", "weight": 1}], "c1": [{"id": "d1", "weight": 1}, {"id": "a1", "weight": 1}, {"id": "b1", "weight": 3}], "d2": [{"id": "a2", "weight": 1}, {"id": "c2", "weight": 1}, {"id": "b2", "weight": 3}, {"id": "e2", "weight": 1}], "b1": [{"id": "a1", "weight": 1}, {"id": "c1", "weight": 3}], "c2": [{"id": "a2", "weight": 1}, {"id": "b2", "weight": 2}, {"id": "d2", "weight": 1}, {"id": "e2", "weight": 3}]} |

### 9.2.2 Closeness-Centrality berechnen

Grundlage für diesen Teiltestfall ist der
in 9.1 gezeigte Graph. Dabei wurden bereits in 9.2.1 die zugrunde liegenden Daten hinzugefügt.

| **Eingabe** | **Soll-Ausgabe** |
|---|---|
| Closeness-Centrality berechnen | |
| Der Graph G wird entsprechend codiert: {"a1": [{"id": "a2", "weight": 4}, {"id": "b1", "weight": 1}, {"id": "c1", "weight": 1}], "b2": [{"id": "a2", "weight": 1}, {"id": "c2", "weight": 2}, {"id": "d2", "weight": 3}, {"id": "e2", "weight": 1}], "a2": [{"id": "b2", "weight": 1}, {"id": "a1", "weight": 4}, {"id": "d2", "weight": 1}, {"id": "e2", "weight": 1}, {"id": "c2", "weight": 1}], "d1": [{"id": "c1", "weight": 1}], "e2": [{"id": "a2", "weight": 1}, {"id": "b2", "weight": 1}, {"id": "c2", "weight": 3}, {"id": "d2", "weight": 1}], "c1": [{"id": "d1", "weight": 1}, {"id": "a1", "weight": 1}, {"id": "b1", "weight": 3}], "d2": [{"id": "a2", "weight": 1}, {"id": "c2", "weight": 1}, {"id": "b2", "weight": 3}, {"id": "e2", "weight": 1}], "b1": [{"id": "a1", "weight": 1}, {"id": "c1", "weight": 3}], "c2": [{"id": "a2", "weight": 1}, {"id": "b2", "weight": 2}, {"id": "d2", "weight": 1}, {"id": "e2", "weight": 3}]}an /api/network-analysis geschickt. Die Anfrage bezieht sich auf die Closeness-Centrality-Analyse. Closeness-Centrality steht im Header. | Die Werte $b2 = c2 = d2 = e2 = \frac{1}{2}$, $a1 = \frac{8}{13}$, $a2 = \frac{8}{12}$, $b1 = \frac{8}{18}$, $c1 = \frac{8}{17}$ und $d1 = \frac{8}{24}$ werden entsprechend codiert zurück gegeben: {"a1": 0.615385, "a2": 0.666667, "b1": 0.444444, "b2": 0.5, "c1": 0.470588, "c2": 0.5, "d2": 0.5, "e2": 0.5} |
| Der Graph G befindet sich als Rohdaten in der Datenbank (Durch Testfall Interaktionsdaten hinzufügen). Die Anfrage bezieht sich auf die Closeness-Centrality-Analyse. Closeness-Centrality steht im Header. | Die Werte $b2 = c2 = d2 = e2 = \frac{1}{2}$, $a1 = \frac{8}{13}$, $a2 = \frac{8}{12}$, $b1 = \frac{8}{18}$, $c1 = \frac{8}{17}$ und $d1 = \frac{8}{24}$ werden entsprechend codiert zurück gegeben: {"a1": 0.615385, "a2": 0.666667, "b1": 0.444444, "b2": 0.5, "c1": 0.470588, "c2": 0.5, "d2": 0.5, "e2": 0.5} |

### 9.2.3 Betweenness-Centrality berechnen

Grundlage für diesen Teiltestfall ist der in 9.1 gezeigte Graph. Dabei wurden bereits in 9.2.1 die zugrunde liegenden Daten hinzugefügt.

| **Eingabe** | **Soll-Ausgabe** |
|---|---|
| Betweenness-Centrality berechnen | |
| Der Graph G wird entsprechend codiert: {"a1": [{"id": "a2", "weight": 4}, {"id": "b1", "weight": 1}, {"id": "c1", "weight": 1}], "b2": [{"id": "a2", "weight": 1}, {"id": "c2", "weight": 2}, {"id": "d2", "weight": 3}, {"id": "e2", "weight": 1}], "a2": [{"id": "b2", "weight": 1}, {"id": "a1", "weight": 4}, {"id": "d2", "weight": 1}, {"id": "e2", "weight": 1}, {"id": "c2", "weight": 1}], "d1": [{"id": "c1", "weight": 1}], "e2": [{"id": "a2", "weight": 1}, {"id": "b2", "weight": 1}, {"id": "c2", "weight": 3}, {"id": "d2", "weight": 1}], "c1": [{"id": "d1", "weight": 1}, {"id": "a1", "weight": 1}, {"id": "b1", "weight": 3}], "d2": [{"id": "a2", "weight": 1}, {"id": "c2", "weight": 1}, {"id": "b2", "weight": 3}, {"id": "e2", "weight": 1}], "b1": [{"id": "a1", "weight": 1}, {"id": "c1", "weight": 3}], "c2": [{"id": "a2", "weight": 1}, {"id": "b2", "weight": 2}, {"id": "d2", "weight": 1}, {"id": "e2", "weight": 3}]}an /api/network-analysis geschickt. Die Anfrage bezieht sich auf die Betweenness-Centrality-Analyse. Betweenness-Centrality steht im Header. | Die Werte $b1 = b2 = c2 = d1 = d2 = e2 = 0$, $a1 = 30$, $a2 = 32$ und $c1 = 14$ werden entsprechend codiert zurück gegeben: {"a1": 30, "a2": 32, "b1": 0, "b2": 0, "c1": 14, "c2": 0, "d2": 0, "e2": 0} |
| Der Graph G befindet sich als Rohdaten in der Datenbank. Die Anfrage bezieht sich auf die Betweenness-Centrality-Analyse. Betweenness-Centrality steht im Header. | Die Werte $b1 = b2 = c2 = d1 = d2 = e2 = 0$, $a1 = 30$, $a2 = 32$ und $c1 = 14$ werden entsprechend codiert zurück gegeben: {"a1": 30, "a2": 32, "b1": 0, "b2": 0, "c1": 14, "c2": 0, "d2": 0, "e2": 0} |

### 9.2.4 Eigenvector-Centrality berechnen

Grundlage für diesen Teiltestfall ist der in 9.1 gezeigte Graph. Dabei wurden bereits in 9.2.1 die zugrunde liegenden Daten hinzugefügt.

| **Eingabe** | **Soll-Ausgabe** |
|---|---|
| Eigenvector-Centrality berechnen | |
| Der Graph G wird entsprechend codiert: {"a1": [{"id": "a2", "weight": 4}, {"id": "b1", "weight": 1}, {"id": "c1", "weight": 1}], "b2": [{"id": "a2", "weight": 1}, {"id": "c2", "weight": 2}, {"id": "d2", "weight": 3}, {"id": "e2", "weight": 1}], "a2": [{"id": "b2", "weight": 1}, {"id": "a1", "weight": 4}, {"id": "d2", "weight": 1}, {"id": "e2", "weight": 1}, {"id": "c2", "weight": 1}], "d1": [{"id": "c1", "weight": 1}], "e2": [{"id": "a2", "weight": 1}, {"id": "b2", "weight": 1}, {"id": "c2", "weight": 3}, {"id": "d2", "weight": 1}], "c1": [{"id": "d1", "weight": 1}, {"id": "a1", "weight": 1}, {"id": "b1", "weight": 3}], "d2": [{"id": "a2", "weight": 1}, {"id": "c2", "weight": 1}, {"id": "b2", "weight": 3}, {"id": "e2", "weight": 1}], "b1": [{"id": "a1", "weight": 1}, {"id": "c1", "weight": 3}], "c2": [{"id": "a2", "weight": 1}, {"id": "b2", "weight": 2}, {"id": "d2", "weight": 1}, {"id": "e2", "weight": 3}]}an /api/network-analysis geschickt. Die Anfrage bezieht sich auf die Eigenvector-Centrality-Analyse. Eigenvector-Centrality steht im Header. | Die Werte $b2 = c2 = d2 = e2 \approx 0.436423$, $a1 \approx 0.137098$, $a2 \approx 0.463508$, $b1 \approx 0.0455334$, $c1 \approx 0.0478608$ und $d1 \approx 0.0117825$ werden entsprechend codiert zurück gegeben: {"a1": 0.137098, "a2": 0.463508, "b1": 0.0455334, "b2": 0.436423, "c1": 0.0478608, "c2": 0.436423, "d2": 0.436423, "e2": 0.436423} |
| Der Graph G befindet sich als Rohdaten bereits in der Datenbank (Durch Testfall Interaktionsdaten hinzufügen). Die Anfrage bezieht sich auf die Eigenvector-Centrality-Analyse. Eigenvector-Centrality steht im Header. | Die Werte $b2 = c2 = d2 = e2 \approx 0.436423$, $a1 \approx 0.137098$, $a2 \approx 0.463508$, $b1 \approx 0.0455334$, $c1 \approx 0.0478608$ und $d1 \approx 0.0117825$ werden entsprechend codiert zurück gegeben: {"a1": 0.137098, "a2": 0.463508, "b1": 0.0455334, "b2": 0.436423, "c1": 0.0478608, "c2": 0.436423, "d2": 0.436423, "e2": 0.436423} |

### 9.2.5 Falsch codierter Graph

Der Server ist online und bereit, Anfragen zu beantworten.

| **Eingabe** | **Soll-Ausgabe** |
|---|---|
| Falsch codierter Graph | |
| Eine HTTP-Anfrage an /api/network-analysis mit dem Text {{ im Body wird geschickt. | Eine Fehlermeldung mit dem Status 400 wird zurück gesendet. |

### 9.2.6 Falsch codierte Interaktionsdaten

Der Server ist online und bereit anfragen zu beantworten.

| **Eingabe** | **Soll-Ausgabe** |
|---|---|
| Falsch codierte Interaktionsdaten | |
| Eine HTTP-Anfrage an /api/interaction mit dem Text {"*Jakob*" : 7} im Body wird geschickt. | Eine Fehlermeldung mit dem Status 400 wird zurück gesendet. |