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": {"a2": 1, "b1": 0.25, "c1": 0.25}, "b2": {"a2": 0.25, "c2": 0.5, "d2": 0.75, "e2": 0.25}, "a2": {"b2": 0.25, "a1": 1, "d2": 0.25, "e2": 0.25, "c2": 0.25}, "d1": {"c1": 0.25}, "e2": {"a2": 0.25, "b2": 0.25, "c2": 0.75, "d2": 0.25}, "c1": {"d1": 0.25, "a1": 0.25, "b1": 0.75}, "d2": {"a2": 0.25, "c2": 0.25, "b2": 0.75, "e2": 0.25}, "b1": {"a1": 0.25, "c1": 0.75}, "c2": {"a2": 0.25, "b2": 0.5, "d2": 0.25, "e2": 0.75} }

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": {"a2": 1, "b1": 0.25, "c1": 0.25}, "b2": {"a2": 0.25, "c2": 0.5, "d2": 0.75, "e2": 0.25}, "a2": {"b2": 0.25, "a1": 1, "d2": 0.25, "e2": 0.25, "c2": 0.25}, "d1": {"c1": 0.25}, "e2": {"a2": 0.25, "b2": 0.25, "c2": 0.75, "d2": 0.25}, "c1": {"d1": 0.25, "a1": 0.25, "b1": 0.75}, "d2": {"a2": 0.25, "c2": 0.25, "b2": 0.75, "e2": 0.25}, "b1": {"a1": 0.25, "c1": 0.75}, "c2": {"a2": 0.25, "b2": 0.5, "d2": 0.25, "e2": 0.75} } an /api/network-analysis geschickt. Die Anfrage bezieht sich auf die Closeness-Centrality-Analyse. Closeness-Centrality steht im Header.	Die Werte { "a1": 0.8648648648648649, "b2": 0.6857142857142856, "a2": 0.8888888888888888, "d1": 0.4085106382978724, "e2": 0.6575342465753425, "c1": 0.6357615894039734, "d2": 0.6575342465753425, "b1": 0.6193548387096774, "c2": 0.6857142857142856 } werden zurückgegeben
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 { "a1": 0.8648648648648649, "b2": 0.6857142857142856, "a2": 0.8888888888888888, "d1": 0.4085106382978724, "e2": 0.6575342465753425, "c1": 0.6357615894039734, "d2": 0.6575342465753425, "b1": 0.6193548387096774, "c2": 0.6857142857142856 } werden zurückgegeben

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": {"a2": 1, "b1": 0.25, "c1": 0.25}, "b2": {"a2": 0.25, "c2": 0.5, "d2": 0.75, "e2": 0.25}, "a2": {"b2": 0.25, "a1": 1, "d2": 0.25, "e2": 0.25, "c2": 0.25}, "d1": {"c1": 0.25}, "e2": {"a2": 0.25, "b2": 0.25, "c2": 0.75, "d2": 0.25}, "c1": {"d1": 0.25, "a1": 0.25, "b1": 0.75}, "d2": {"a2": 0.25, "c2": 0.25, "b2": 0.75, "e2": 0.25}, "b1": {"a1": 0.25, "c1": 0.75}, "c2": {"a2": 0.25, "b2": 0.5, "d2": 0.25, "e2": 0.75} } an /api/network-analysis geschickt. Die Anfrage bezieht sich auf die Betweenness-Centrality-Analyse. Betweenness-Centrality steht im Header.	Die Werte { "a1": 0.5357142857142857, "b2": 0.03571428571428571, "a2": 0.5714285714285714, "d1": 0.0, "e2": 0.0, "c1": 0.25, "d2": 0.0, "b1": 0.0, "c2": 0.03571428571428571 } werden zurückgegeben
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 { "a1": 0.5357142857142857, "b2": 0.03571428571428571, "a2": 0.5714285714285714, "d1": 0.0, "e2": 0.0, "c1": 0.25, "d2": 0.0, "b1": 0.0, "c2": 0.03571428571428571 } werden zurückgegeben

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": {"a2": 1, "b1": 0.25, "c1": 0.25}, "b2": {"a2": 0.25, "c2": 0.5, "d2": 0.75, "e2": 0.25}, "a2": {"b2": 0.25, "a1": 1, "d2": 0.25, "e2": 0.25, "c2": 0.25}, "d1": {"c1": 0.25}, "e2": {"a2": 0.25, "b2": 0.25, "c2": 0.75, "d2": 0.25}, "c1": {"d1": 0.25, "a1": 0.25, "b1": 0.75}, "d2": {"a2": 0.25, "c2": 0.25, "b2": 0.75, "e2": 0.25}, "b1": {"a1": 0.25, "c1": 0.75}, "c2": {"a2": 0.25, "b2": 0.5, "d2": 0.25, "e2": 0.75} } an /api/network-analysis geschickt. Die Anfrage bezieht sich auf die Eigenvector-Centrality-Analyse. Eigenvector-Centrality steht im Header.	Die Werte { "a1": 0.13710442313196577, "b2": 0.4364225935824221, "a2": 0.46350795737155703, "d1": 0.011785494846305042, "e2": 0.4364225935824221, "c1": 0.04786769748674075, "d2": 0.4364225935824221, "b1": 0.04553916449180267, "c2": 0.4364225935824221 } werden zurückgegeben
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 { "a1": 0.13710442313196577, "b2": 0.4364225935824221, "a2": 0.46350795737155703, "d1": 0.011785494846305042, "e2": 0.4364225935824221, "c1": 0.04786769748674075, "d2": 0.4364225935824221, "b1": 0.04553916449180267, "c2": 0.4364225935824221 } werden zurückgegeben

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.