
UNIVERSITÄT
LEIPZIG

Institut für Informatik
Fakultät für Mathematik und Informatik
Abteilung Datenbanken

# Visualisierung zeitlicher Graph Metriken

Bachelorarbeit

vorgelegt von:
Julian Nico Pielmaier

Matrikelnummer:
3752800

Betreuer:
Prof. Dr. Erhard Rahm
Christopher Rost

## Kurzfassung

In vielen Bereichen der heutigen Gesellschaft lassen sich Netzwerke erkennen, in denen komplexe Zusammenhänge zwischen Objekten bestehen. Über die Modellierung eines Netzwerkes als Graphen können diese Beziehungen effektiv dargestellt und analysiert werden. Als eine von vielen Graph Metriken, bietet der Knotengrad eine gute Möglichkeit die Zentralität eines Objektes im Netzwerk zu untersuchen. Da die Struktur eines Netzwerkes in der Realität jedoch sehr dynamisch ist, verändert sich der beschreibende Graph und damit auch dessen Metriken. Um einen Einblick in die Entwicklung der Struktur zu erhalten, ist eine zeitliche Betrachtung nötig. Obwohl theoretische Ansätze existieren, gibt es nur wenige Anwendungen, die eine zeitabhängige Analyse von Graph Metriken ermöglichen. Das Ziel dieser Arbeit ist die Entwicklung eines Tools zur Visualisierung zeitlicher Graph Metriken. Die notwendigen Teilschritte für die Umsetzung dieses Vorhabens werden in den folgenden Kapiteln vorgestellt. Der dabei entworfene Temporal Metric Explorer bildet den Knotengrad in einer Zeitserie ab, wodurch Einblicke in dessen Evolution gewonnen werden können. Die Evaluierung des TME zeigt, dass sich damit interessante Zeitpunkte in der Entwicklung identifizieren und Korrelationen sowie Trends hervorheben lassen.



Julian Nico Pielmaier
3752800

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| AJAX | Asynchronous JavaScript and XML |
| API | Application Programming Interface |
| CSS | Cascading Style Sheets |
| DOM | Document Object Model |
| FA | Funktionale Anforderung |
| HDFS | Hadoop Distributed File System |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| JSON | JavaScript Object Notation |
| MSV | Massive Sequence Views |
| NFA | Nichtfunktionale Anforderung |
| POJO | Plain Old Java Object |
| TGE | Temporal Graph Explorer |
| TME | Temporal Metric Explorer |
| TPGM | Temporal Property Graph Model |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| URN | Uniform Resource Name |

# 1. Einleitung

*„[...] die Zukunft gehört der Netzwerkanalyse.“* [1]

## 1.1. Motivation

Etwa 4.6 Milliarden Menschen nutzten 2022 soziale Netzwerke wie Facebook, YouTube oder WhatsApp [2]. Die Tendenz ist weiterhin steigend. Sei es die Interaktion mit anderen Nutzern über soziale Medien, der Kauf eines Produktes im Onlineshop oder die Fahrt von zu Hause zum Arbeitsplatz – viele Bereiche des alltäglichen Lebens sind vernetzt. Um diese Zusammenhänge modellieren zu können, werden Netzwerke in Graphen überführt. Ein Graph ist eine mathematische Struktur, die Objekte über Knoten abbildet und Beziehungen zwischen den Objekten als Kanten repräsentiert. So werden beispielsweise die Nutzer John und Alice des sozialen Netzwerkes Facebook als Knoten dargestellt und deren Freundschaft als Kante zwischen den Knoten. Da Netzwerke in der Praxis weitaus größer sind und deren Graphen schnell mehrere Millionen Knoten und Kanten erreichen, ist es schwer, Informationen aus ihnen zu gewinnen. Die Graphenanalyse ist daher ein wichtiges Werkzeug zur Interpretation von Netzwerken und es gibt viele verschiedene Ansätze zur Erkennung von Mustern oder Gruppen innerhalb statischer Graphen. Der Knotengrad als Zentralitätsmaß ist eine einfache und zugleich aussagekräftige Metrik zur Identifikation des Einflusses eines Knoten auf den Graphen. Anhand der Anzahl angrenzender Kanten kann durch den Knotengrad die Bedeutsamkeit eines Knoten geschätzt werden. Denn sollte ein Objekt innerhalb eines Netzwerkes eine hohe Konnektivität aufweisen, so ist es wahrscheinlich, dass es einen stärkeren Einfluss auf die Struktur hat als ein Objekt mit wenigen Relationen.

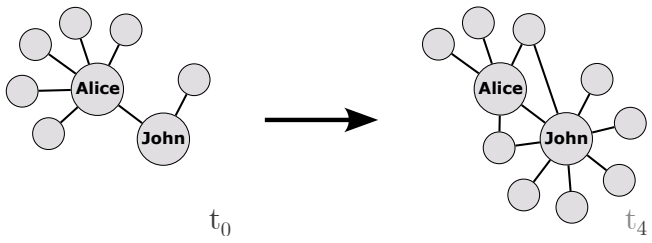



Abb. 1.1.: Entwicklung des sozialen Netzwerkes um John und Alice

Eine statische Betrachtung wird der Schnelllebigkeit der heutigen Gesellschaft und der damit verbundenen hohen Dynamik in vielen Netzwerken nicht gerecht. Über eine zeitabhängige Analyse lässt sich ein Einblick in die Evolution des Graphen gewinnen. Um auf das obige Beispiel zurückzukommen, wird sich der Freundeskreis von John und Alice im Laufe der Zeit voraussichtlich ändern. Es können neue Freundschaften mit anderen Nutzern geschlossen werden und bereits bestehende werden möglicherweise aufgelöst. In Abbildung 1.1 ist diese Entwicklung zu sehen und es lässt sich erkennen, dass John mit der Zeit neue Freundschaften im Netzwerk geschlossen hat. Neben der Veränderung der Graphenstruktur, wandeln sich auch die zugehörigen Metriken. Der Knotengrad zum Beispiel wird dementsprechend nicht mehr nur als skalarer Wert dargestellt, sondern dessen Entwicklung als Zeitserie. Die Visualisierung zeitlicher Graph Metriken deckt im Gegensatz zur

statischen Analyse Korrelationen und Trends auf und enthüllt so einen neuen Blickwinkel auf das Netzwerk. Dadurch lassen sich neue Erkenntnisse gewinnen, mit denen zukunftsorientierte Entscheidungen getroffen, Prozesse optimiert oder Risiken identifiziert werden können. Die Abbildung 1.2 zeigt die Knotengradevolution von John. Im Gegensatz zu einer aggregierten Sicht auf den Graphen lassen sich hierbei genaue Zeitpunkte identifizieren. Dabei ist zu erkennen, dass John zum Zeitpunkt $t_1$ die meisten neuen Freundschaften knüpfte.

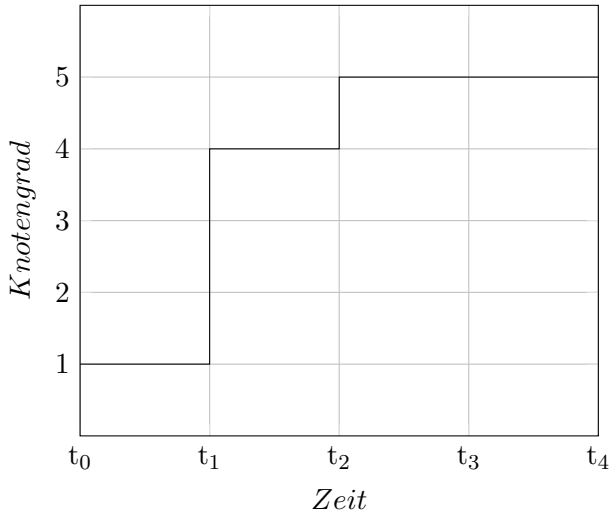



Abb. 1.2.: Knotengradevolution von John als Zeitserie

Trotz der vielfältigen Analysemöglichkeiten, gibt es momentan nur wenige Arbeiten, die die Evolution von Graph Metriken in Zeitserien visualisieren. Um diese Forschungslücke schrittweise zu schließen, wird im Rahmen dieser Arbeit ein Konzept für eine Anwendung zur Visualisierung zeitlicher Graph Metriken entworfen und anschließend ein Prototyp implementiert.

## 1.2. Zielsetzung

Der an der Universität Leipzig entworfene Temporal Graph Explorer (TGE) unterstützt durch die Verwendung von Graph-Operatoren bereits eine Komponente zur Analyse von temporalen Graphen. Der in dieser Arbeit zu entwerfende Temporal Metric Explorer (TME) soll auf dem bestehenden TGE aufbauen und diesen um die Möglichkeit zur Visualisierung zeitlicher Graph Metriken erweitern. Gegenstand dieser Arbeit ist damit die Entwicklung einer Anwendung zur Analyse zeitlicher Graph Metriken. Der Fokus liegt dabei auf einer interaktiven Darstellung, die es erlaubt, Informationen aus visualisierten Daten auf explorative Art und Weise zu gewinnen. Ein Nutzer soll neben der Wahl einer Graph Metrik auf einen eigenen Datensatz auch das zu betrachtende Zeitintervall konfigurieren können und anhand der gewählten Optionen die Metrikevolution visualisiert bekommen. Die Anwendung soll zur Berechnung der zeitlichen Graph Metriken das Framework Gradoop verwenden und den bereits implementierten Knotengrad unterstützen. Neben der Entwicklung eines Prototyps gilt es die Frage zu lösen, welche funktionalen und nichtfunktionalen Anforderungen an eine solche Anwendung bestehen und wie diese Anforderungen geeignet umgesetzt werden können. Die genauen Voraussetzungen an die Implementierung werden in Abschnitt 4.1 erläutert.

## 1.3. Aufbau der Arbeit

Zum Beginn der Arbeit werden im Kapitel *Grundlagen* Begriffe und Prinzipien erklärt, die für das Verständnis der weiteren Abschnitte erforderlich sind. Anschließend erfolgt in *Verwandte Arbeiten* ein kurzer Einblick in die aktuelle Forschungslage, indem bereits existierende Methoden zur Visualisierung temporaler Graphen vorgestellt werden. Hierbei wird der Bedarf an einer Anwendung mit dem Fokus auf der Analyse von Metriken aufgezeigt und der zu entwerfende Temporal Metric Explorer zwischen ähnlichen Arbeiten eingeordnet und abgegrenzt. Im Kapitel *Konzeption* wird ein technisches Konzept des TME entworfen und es werden Anforderungen erhoben, die ein solches Analysetool erfüllen soll. Im nächsten Schritt werden in der *Realisierung* für die Umsetzung geeignete Technologien gewählt und die Implementierung eines Prototyps wird anhand einzelner Komponenten vorgestellt. Schlussendlich erfolgt durch die Auswertung von Beispieldatensätzen und einer Vorstellung der Funktionalitäten die *Evaluierung* des Temporal Metric Explorers.

# 2. Grundlagen

In diesem Kapitel werden die Begrifflichkeiten *temporale Graphen* und *zeitliche Graph Metriken* definiert. Deren Verständnis ist grundlegend für die weiteren Kapitel diese Arbeit.

## 2.1. Temporale Graphen

Nach Diestel [3, S. 2] ist ein Graph $G$ ein geordnetes Paar $G = (V, E)$, bestehend aus einer Menge $V$ von Knoten und einer Menge $E \subseteq V \times V$ von Kanten. Eine Kante $e \in E$ und ein Knoten $v$ heißen inzident, wenn $v \in e$. Zwei über eine Kante verbundene Knoten sind adjazent und werden Nachbarn genannt. Unterschieden werden gerichtete und ungerichtete Graphen. Sollte $E$ geordnete Knotenpaare enthalten, wird von einem gerichteten Graphen [4, S. 8] gesprochen. Wenn $(u, v) \in E$ und $(v, u) \notin E$, steht $u$ in Relation zu $v$, nicht aber $v$ in Relation zu $u$. Die Kanten in gerichteten Graphen werden typischerweise als Pfeile von $u$ nach $v$ dargestellt. Ungerichtete Graphen hingegen zeichnen sich dadurch aus, dass wenn $u$ in Relation zu $v$ steht, $v$ auch in Relation zu $u$ steht. Demnach gilt $(u, v) \in E \rightarrow (v, u) \in E$. Die Kanten in ungerichteten Graphen werden für gewöhnlich als Linien dargestellt. Sollte $G = (V, E, w)$ gelten, handelt es sich um einen gewichteten Graphen [5, S. 11]. Dieser enthält, durch die Funktion $w : E \rightarrow \mathbb{R}_+$, gewichtete Kanten.

Die Struktur eines Graphen eignet sich besonders gut zur Darstellung von Daten, die miteinander in Beziehung stehen. Um praktische Anwendungsfälle zu modellieren, repräsentiert $V$ eine Menge von Objekten und $E$ eine Menge von Relationen zwischen den Objekten. Ein charakteristisches Beispiel hierfür sind soziale Netzwerke wie Facebook. Die Benutzer werden in $V$ abgebildet und die Freundschaften in $E$. Das gesamte Netzwerk bildet demnach einen Graphen $G$. Die Struktur solcher Netzwerke ist in der Regel sehr dynamisch. Dem entgegen steht jedoch die eben genannte statische Beschreibung von $G$. Eine zeitliche Betrachtung des Graphen $G$ trägt dazu bei, die Entwicklung der Struktur des Graphen abzubilden.

Eine Möglichkeit zur zeitlichen Analyse eines Graphen besteht in der Modellierung statischer Graphen als Serie. Die Folge von statischen Graphen $G^t = (V^t, E^t)$ zum jeweiligen Zeitpunkt $t$ mit $t_{min} <= t <= t_{max}$ bildet dabei den temporalen Graphen $G_t = \{G_t^{t_{min}}, ..., G_t^{t_{max}}\}$. Ein Graph $G^{t+1}$ folgt aus dem vorherigen Graphen $G^t$. Dabei sind Änderungen an der Graphenstruktur wie das Hinzufügen oder Löschen von Knoten und Kanten möglich [5, S. 10]. Die Menge $V^t \subseteq V$ enthält die Knoten $v \in V$ zum Zeitpunkt $t$. Dabei ist $V$ die Vereinigung aller Mengen $V^t$ und es gilt $\bigcup_{t_{min}}^{t_{max}} V^t = V$. Die Menge $E^t \subseteq E$ lässt sich ebenso als Menge der Kanten $e \in E$ zum Zeitpunkt $t$ auffassen, wobei $\bigcup_{t_{min}}^{t_{max}} E^t = E$ [6, S. 6f]. Eine weitere Möglichkeit temporale Graphen zu modellieren besteht darin, nicht nur den Graphen, sondern auch seine einzelnen Komponenten mit einer Zeitdimension zu erweitern [7, S. 5]. Hierbei führen der temporale Graph selbst sowie dessen Knoten und Kanten jeweils ein Zeitintervall als Attribut. Über das Attribut kann definiert werden, in welchem Zeitraum die Elemente valide sind.

Auf das eben genannte Beispiel bezogen lässt sich mittels eines temporalen Graphen somit nicht nur das Netzwerk zu einem spezifischen Zeitpunkt abbilden, sondern dessen Entwicklung über ein Zeitintervall. In Abbildung 2.1 wurde dies veranschaulicht. Hier ist zu erkennen, dass der dargestellte temporale Graph zum Zeitpunkt $t_0$ eine andere Struktur aufweist als zum Zeitpunkt $t_4$. Wie in echten Netzwerken üblich kamen mit der Zeit Knoten und Kanten hinzu, während andere wiederum verschwanden. Aus einer statischen Sicht auf den Graphen wäre diese Evolution nicht erkennbar.

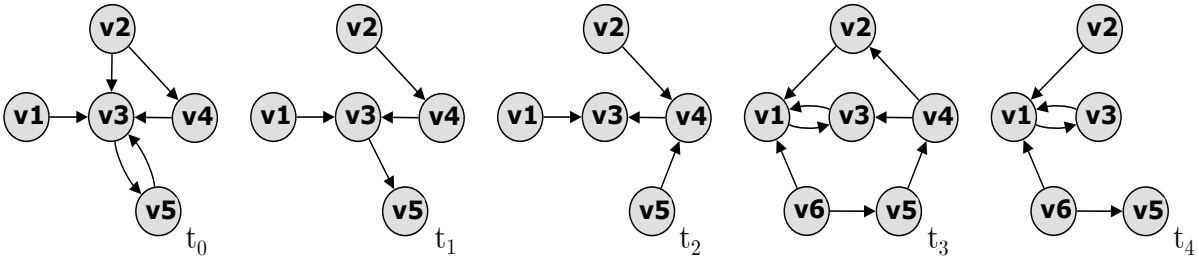



Abb. 2.1.: Evolution eines temporalen Graphen

## 2.2. Zeitliche Graph Metriken

Graph Metriken geben Aufschluss über die Struktur eines Graphen. Es gibt verschiedene Arten von Metriken in der Graphenanalyse, die entweder den gesamten Graph oder einzelne Knoten betrachten. Eine einfache und zugleich aussagekräftige Metrik, ist der *Degree* (dt. Knotengrad). Der Knotengrad bestimmt die Konnektivität eines Knotens im Graphen. Je höher der Knotengrad eines Knotens $v$ ist, desto stärker ist $v$ vernetzt.

Nach der Definition von Gross et al. ist der Knotengrad $deg(v)$ eines Graphen $G$ die Anzahl der zu $v$ inzidenten Kanten [8, S. 8]. In ungerichteten Graphen ohne mehreren Kanten zwischen zwei gleichen Knoten entspricht dies der Anzahl der Nachbarn von $v$. Hierbei sei anzumerken, dass Kanten der Form $e = (u, u)$ doppelt zählen. Solche Kanten werden als Schleifen betitelt. Der Knotengrad lässt sich formal ausdrücken als:

$$deg(v) = |\{u \in V \mid u \text{ und } v \text{ sind adjazent}\}|$$

In gerichteten Graphen lässt sich der Knotengrad noch weiter unterteilen in den Eingangsknotengrad $deg^-(v)$ und den Ausgangsknotengrad $deg^+(v)$. Der Eingangsknotengrad gibt die Anzahl der nach $v$ gerichteten Kanten an. Äquivalent dazu wird der Ausgangsgrad durch die Anzahl der von $v$ ausgehenden Kanten bestimmt. Die formale Beschreibung lautet wie folgt:

$$deg^-(v) = |\{u \in V \mid (u, v) \in E\}|$$
$$deg^+(v) = |\{u \in V \mid (v, u) \in E\}|$$

Der Knotengrad samt seinen unterschiedlichen Ausführungen kann ein Maß für die Vernetzungsdichte eines Knotens im Graphen sein. Er findet daher auch als Zentralitätsmaß unter der Bezeichnung *Degree Centrality* Verwendung [9]. Neben der Degree Centrality gibt es noch weitere, komplexere Zentralitätsmaße wie die *Closeness*, *Betweenness* und *Eigenvector Centrality* [10]. Diese Metriken beziehen sich allesamt lediglich auf einen spezifischen Knoten. Daraus lässt sich per se jedoch nicht auf die Graphenstruktur als Ganzes schließen. Daher gibt es auch aggregierte Metriken, die den gesamten Graphen in Betracht ziehen. Beispiele für solche Graph Metriken sind der *Minimun-*, *Maximum-* oder *Durchschnittsknotengrad*. Der Minimumknotengrad gibt den kleinsten Knotengrad $deg_{min}(G)$ aller Knoten im Graphen $G$ an. Der Maximumknotengrad $deg_{max}(G)$ wiederum den größten Knotengrad. Der Durchschnittsknotengrad $deg_{avg}(G)$ bildet einen gewichteten Durchschnittswert aller Knotengrade in $G$. Formal lassen sich diese Graph Metriken folgendermaßen ausdrücken [3, 8]:

$$deg_{min}(G) = min\{deg(v) \mid v \in V\}$$
$$deg_{max}(G) = max\{deg(v) \mid v \in V\}$$
$$deg_{avg}(G) = \frac{1}{|V|} \sum_{v \in V} deg(v)$$

Die aktuelle Betrachtung der Graph Metriken basiert auf statischen Graphen. Wie bei der Umwandlung von statischen zu temporalen Graphen kann die Zeit als weitere Dimension ergänzt werden, um von zeitlichen Graph Metriken sprechen zu können.

Der zeitliche Knotengrad [6] $degt(v,t)$ gibt den Knotengrad des Knotens $v$ zum Zeitpunkt $t$ an. Sollte $v$ nicht in $V^t$ liegen, dann ist der zeitliche Knotengrad undefiniert. Die Erweiterung des Ein- und Ausgangsknotengrades erfolgt identisch. Auch der Minimum-, Maximum- und Durchschnittsknotengrad lässt sich problemlos erweitern. Der zeitliche Minimumknotengrad $degt_{min}(G,t)$ eines temporalen Graphen $G$ ergibt sich aus dem Minimum der Menge aller Knotengrade von $G$ zum Zeitpunkt $t$. Auf analoge Weise lassen sich der Maximum- und der Durchschnittsknotengrad konstruieren. In Abbildung 2.2 sind zwei Zustände eines temporalen Graphen $G_t$ abgebildet. Um die genannten zeitlichen Graph Metriken zu veranschaulichen, sind hier Beispiele gegeben:

$$degt(v3,t_0)=5 \qquad degt^-(v3,t_0)=4 \qquad degt^+(v3,t_0)=1$$
$$degt_{min}(G_t,t_0)=1 \qquad degt_{max}(G_t,t_0)=5 \qquad degt_{avg}(G_t,t_0)=2.4$$

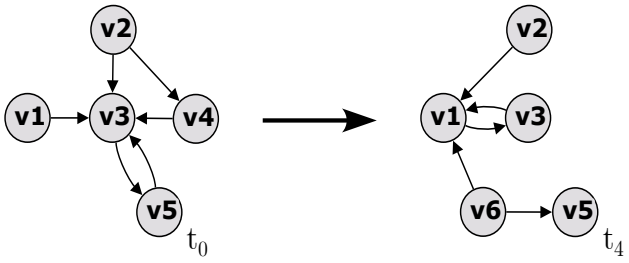



Abb. 2.2.: Ein temporaler Graph zu zwei Zeitpunkten

Der zeitliche Knotengrad bezieht sich zwar auf einen temporalen Graphen, betrachtet jedoch nur einen spezifischen Zeitpunkt. Die Sicht auf den Graphen oder auf einen Knotens ist demnach weiterhin statisch und gibt keine Auskunft über die zeitliche Entwicklung. In Abbildung 2.2 ist zu sehen, dass der Knoten $v1$ zum Zeitpunkt $t_0$ einen zeitlichen Knotengrad von 1 besitzt, wohingegen bei $t4$ ein Wert von 4 vorliegt. Wenn $t_0 < t_4$, dann hat $v1$ an Zentralität gewonnen. Im Vergleich des zeitlichen Minimumknotengrades von $degt_{min}(G_t,t_0)=1$ mit dem Maximumknotengrad $degt_{max}(G_t,t_4)=4$ ist zu erkennen, dass sich Knoten $v1$ vom unbedeutendsten zum einflussreichsten Knoten entwickelt hat.

Diese Entwicklung lässt sich über die *Knotengradevolution degev* ausdrücken. Über ein Intervall $\tau$, werden die skalaren Werte des zeitlichen Knotengrades in einer Zeitserie abgebildet. Die Knotengradevolution besteht dann aus Tupeln von Knotengraden und ihren entsprechenden Zeitpunkten. Nach Rost et al. ergibt sich daraus folgende Definition [6, S. 8]:

$$degev(v,\tau)=\{\langle degt(v,t),t\rangle \mid t\in\tau\} \text{ wobei } \tau=[t_{start},t_{end})$$

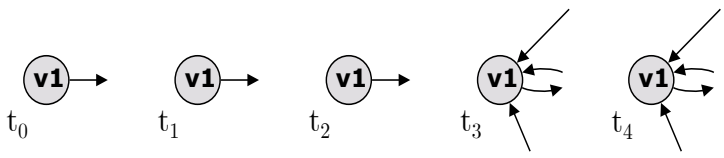



Abb. 2.3.: Evolution eines Knotens

Wird in Abbildung 2.3 angenommen, dass $t_i < t_{i+1}$ mit $i\in\mathbb{N}_0$, so ist die Knotengradevolution des Knotens $v1$ gleich $degev(v1,[t_0,t_5))=\{\langle 1,t_0\rangle,\langle 1,t_1\rangle,\langle 1,t_2\rangle,\langle 4,t_3\rangle,\langle 4,t_4\rangle\}$.

# 3. Verwandte Arbeiten

Graphen haben sich als nützliches Werkzeug zur Modellierung von in Beziehung stehenden Daten bewährt, doch deren Visualisierung stellt selbst erfahrene Experten vor große Herausforderungen. Da komplexe Graphen sehr groß werden, kann die Anzahl der Datenpunkte die Darstellungsfläche übertreffen oder zu einer Informationsüberflutung führen. Selbst wenn sich alle Daten abbilden lassen, steht die Frage im Raum, wie der Graph geeignet in einer Ebene dargestellt werden kann. Denn auch die Art und Weise wie Verbindungen zwischen Knoten gezogen werden, trägt maßgeblich dazu bei, wie gut der Graph interpretiert werden kann. Daher haben sich viele verschiedene Ansätze zur Visualisierung von Graphen entwickelt, die jeweils Vor- und Nachteile haben. Eine gute Vorstellung unterschiedlicher Darstellungsmethoden liefern Herman et al. in ihrer Studie [11].

Eine Analyse von statischen Netzwerken ist durch die zunehmende Dynamik in vielen Anwendungsgebieten heute jedoch unzureichend. Daher wird die Visualisierung von zeitlichen Graphen immer bedeutsamer. Meist werden für die zeitliche Abbildung Animationen verwendet oder es werden kleine Mehrfachdiagramme eingesetzt. Bei den kleinen Mehrfachdiagrammen wird ein zeitlicher Graph in mehreren nebeneinander stehenden Diagrammen zu unterschiedlichen Zeitpunkten dargestellt. Das Problem hierbei ist die Wahl der optimalen Anzahl an darzustellenden Graphausschnitten. Zu viele Diagramme würden den Fokus auf interessante Eigenschaften erschweren und durch zu wenige Diagramme könnten wichtige Entwicklungen unentdeckt bleiben. Bei Animationen ist es kompliziert, sich auf mehrere parallele Änderungen zu konzentrieren sowie sich zeitliche Entwicklungen zu merken. Eine Visualisierungstechnik, die diese Probleme umgeht, ist die Darstellung in *Massive Sequence Views* (MSV) [12]. Die Knoten-Kanten-Diagramme von Graphen werden hierbei durch Zeitreihen ersetzt. Diese Zeitreihen repräsentieren auf einer Achse die Zeitdimension und auf einer anderen Achse die Knotenmenge des zeitlichen Graphen. Die Knoten werden dabei entsprechend der Graphenstruktur über Linien miteinander verbunden. Eine Linie zwischen zwei Knoten stellt dabei eine Kante im Graphen dar. MSV werden standardmäßig in zwei orthogonalen Achsen abgebildet, doch es existiert auch die Verwendung einer kreisförmigen Darstellung. Hierbei gibt die Nähe zum Kreismittelpunkt die zeitliche Dimension an und die Kreissektoren beinhalten die Knoten. Die Visualisierungstechnik hat sich in Kombination mit Filtermöglichkeiten als gut skalierbar herausgestellt. Doch die Analyse von MSV ist weniger intuitiv und hängt stark davon ab, wie die Knoten auf der Achse angeordnet sind.

Die meisten Verfahren zur zeitlichen Graphenvisualisierung beschränken sich auf die Graphenstruktur als Ganzes, geben jedoch ungenaue Auskunft über spezifische Graph Metriken. Es gibt zwar Ansätze [13], die den Grad eines Knotens beispielsweise über seine Größe im Graphen oder dessen Farbe definieren, doch diese visuellen Eigenschaften erlauben keine gezielte Analyse. Pohl et al. haben eine Anwendung entwickelt, die neben der Visualisierung zeitlicher Graphen auch auf zeitliche Metriken eingeht [14]. Das von ihnen entworfene DGD-Tool unterstützt bereits eine Ansicht für die zeitliche Auswertung der Gradzentralität eines Knotens.

Beim DGD-Tool ist jedoch zu sehen, dass der Fokus auf der Graphenvisualisierung liegt und die Knotengradmetrik nur als Zusatzfunktion zu betrachten ist. Denn neben der reinen Visualisierung gibt es nur wenige Konfigurationsmöglichkeiten für Graph Metriken.

Der zu entwickelnde Temporal Metric Explorer hingegen wird sich auf die Untersuchung von Graph Metriken spezialisieren und soll in der Kombination mit dem Temporal Graph Explorer vielfältige Funktionalitäten für eine umfangreiche Analyse ermöglichen. Dazu wird er den TGE um die Möglichkeit zur Exploration zeitlicher Graph Metriken erweitern und erlaubt damit tiefere Einblicke in die Dynamiken von Graphen.

Als Open Source Framework ermöglicht der *Temporal Graph Explorer* [15] die zeitabhängigen Exploration von Graphen sowie die Analyse von großen Echtweltnetzwerken. Mit Hilfe einer webbasierten Benutzeroberfläche kann der Nutzer einen von ihm gewählten zeitlichen Graphen nach eigenen Konfigurationen untersuchen. Dazu stehen ihm die Operatoren *Snapshot*, $Difference$ und *Grouping* zur Verfügung. Snapshot ermöglicht die Visualisierung eines zeitlichen Graphen zu einem bestimmten Zeitpunkt oder Intervall. Difference gibt einen Einblick darüber, wie sich der Graph über einen Zeitraum verändert hat. Eine aggregierte und damit abstraktere Sicht auf den Graphen lässt sich mittels Grouping visualisieren.

Die durch den Temporal Graph Explorer zu untersuchenden zeitlichen Graphen werden intern durch das *Temporal Property Graph Model* (TPGM) [7] modelliert. Die Verwendung des TPGM wird durch Gradoop [6, 7, 15, 16, 17] ermöglicht. Als Open Source Datenverarbeitungssystem zur Analytik zeitlicher Graphen bildet Gradoop somit den Kern des TGE. Die integrierte und anwendungsspezifische Sprache *GraLa* bietet einen deklarativen Umgang mit dem TPGM. Die im TGE verwendeten Operatoren Snapshot, Difference und Grouping lassen sich demzufolge über GraLa auf das TPGM anwenden. Da zeitliche Graphen die Speicherung historischer Daten erfordern, können die Graph-Datensätze schnell sehr groß werden. Ein effektiver Umgang mit solchen großen Datenmengen erfordert eine skalierbare Berechnung. Der Begriff Gradoop, der aus den Wörtern Graph und Hadoop zusammengesetzt ist, deutet bereits auf eine verteilte Arbeitsweise hin.

Apache Hadoop bietet mit dem Hadoop Distributed File System (HDFS) [18, S. 1] ein verteiltes Dateisystem. Mit diesem ist es möglich, sehr große Datenmengen skalierbar in mehrere Einheiten zu verteilen und dabei den Schein eines einzigen Dateisystems zu wahren. Hadoop sorgt anschließend zuverlässig für die Verteilung und Verfügbarkeit der Daten, indem es die Daten selbständig repliziert und Ausfälle automatisch erkennt und behebt. Bereits im Jahr 2010 konnten 25 Petabyte an Daten, die über Yahoo! angefallen sind, mit HDFS gespeichert und verarbeitet werden, was die Leistungsfähigkeit von HDFS unter Beweis stellt.

Gradoop nutzt zum HDFS zusätzlich Apache Flink [19] als Open Source Datenfluss-Engine. Flink kann verteilte Batch- und Stream-Daten verarbeiten und ist im Zusammenspiel mit HDFS für eine verteilte Berechnung geeignet. Das HDFS sorgt für die Verfügbarkeit der Daten, sowie deren Verteilung und Flink kümmert sich um die Verarbeitung und Analyse.

# 4. Konzeption des Temporal Metric Explorer

Bevor mit der Entwicklung des Temporal Metric Explorer begonnen wird, ist es wichtig ein genaues Konzept zu erstellen. Diese Konzeption umfasst unter anderem eine Anforderungsanalyse, durch die konkrete Ziele definiert werden können. Des Weiteren trägt ein Entwurf der Softwarearchitektur und des Designs maßgeblich zu einer erfolgreichen Entwicklung bei, indem bereits vor der Implementierung Fehler entdeckt und behoben werden können.

## 4.1. Anforderungsanalyse

In diesem Kapitel wird eine Anforderungsanalyse erhoben. Dabei wird zur besseren Übersicht in funktionale und nichtfunktionale Anforderungen unterschieden.

### 4.1.1. Funktionale Anforderungen

Die Analyse von zeitlichen Graph Metriken steht bei der Entwicklung des TME im Fokus. Um dieses Ziel zu erreichen, müssen zu Beginn die dafür benötigten Funktionalitäten festgelegt werden. Im folgenden Abschnitt werden diese zusammengefasst.

Innerhalb des TME und TGE wird es die Möglichkeiten geben, zur jeweils anderen Anwendung zu gelangen. Zudem gibt es durch das Wählen eines Knotens im TGE die Funktion, dessen Metrik im TME direkt anzeigen zu lassen. Im TME wird es über einen Button in der Seitenleiste möglich sein, eine zeitliche Graph Metrik hinzuzufügen. Der für die Metrik zugrundeliegende temporale Graph soll durch den Benutzer frei wählbar und konfigurierbar sein. Dabei kann der Graph auf ein spezifisches Zeitfenster angepasst oder nach Knoten gefiltert werden. Die darauf zu berechnende Metrikevolution wird von Gradoop berechnet und anschließend in einem Diagramm visualisiert. Bereits errechnete Metriken werden zusätzlich zwischengespeichert. Zudem können mehrere Metriken gleichzeitig dargestellt werden und über Informationen in der Seitenleiste lassen sich die gewählten Graph-Konfigurationen den Diagrammen zuordnen. Die Seitenleiste wird zur Vergrößerung der Darstellungsfläche einklappbar sein. Linien repräsentieren innerhalb der Diagramme einen spezifischen Knoten und dessen zeitlichen Verlauf der zuvor konfigurierten Metrik. Der Benutzer kann diese Knoten frei wählen. Ein Diagramm kann mehrere Knoten abbilden und farbige Bezeichner werden die Zuordnung von Knoten zu Linien ermöglichen. Das Klicken auf einen Bezeichner wird den dazugehörigen Knoten ausblenden. Für die Diagramme werden weitere Optionen geboten. So wird der Benutzer das zu betrachtende Zeitintervall einstellen können und Knoteninformationen beim Hovern erhalten. Am Rand und innerhalb des Diagramms werden die Minimum-, Maximum- und Durchschnittsmetrik visualisiert. Der Moving Average wird ebenfalls unterstützt und dessen Fenstergröße wird anpassbar sein. Auch eine Textansicht auf die visualisierten Daten wird vorhanden sein und das Diagramm wird sich als Bild herunterladen lassen.

Die eben genannte Zusammenfassung ist zum besseren Verständnis teilweise grob formuliert. Da diese Beschreibung jedoch Interpretationsraum offen lässt und sich somit die Anforderungen nicht objektiv validieren lassen, bedarf es einer eindeutigen Definition. In Tabelle 4.1 werden daher die funktionalen Anforderungen an den TME tabellarisch aufgeführt und präzise beschrieben.

| ID | Funktionale Anforderung | Beschreibung |
|---|---|---|
| FA1 | TGE-Verknüpfung | Es gibt jeweils einen Hyperlink vom TGE zum TME und umgekehrt. |
| | | Über einen gewählten Knoten im TGE kann die entsprechende Knotenmetrik im TME visualisiert werden. |
| FA2 | Metrikverwaltung | Es gibt einen Button zum Hinzufügen einer neuen Metrik. |
| | | Es gibt einen Button zum Löschen einer vorhandenen Metrik. |
| | | Eine bereits visualisierte Metrik lässt sich einer Graph-Konfiguration zuordnen. |
| FA3 | Graph-Konfiguration | Der Graph-Datensatz kann gewählt werden. |
| | | Es kann zwischen Eingangs-, Ausgangs- und einfachen Knotengrad gewählt werden. |
| | | Es kann gewählt werden, ob der ganze Zeitraum berechnet werden soll, ein spezifischer Zeitpunkt $t_1$, das Intervall von $t_1$ bis zu einem zweiten Zeitpunkt $t_2$ oder der Zeitraum zwischen $t_1$ und $t_2$. |
| | | Die Zeitpunkte $t_1$ und $t_2$ sind bei Bedarf veränderbar. |
| | | Es können Knoten als Filter angegeben werden. |
| FA4 | Metrikberechnung | Die Metrik wird durch Gradoop berechnet. |
| | | Die Metrik wird auf Grundlage der gewählten Konfiguration berechnet. |
| FA5 | Metrikvisualisierung | Die zeitliche Metrik eines Knoten wird gemäß der Konfiguration in einem Stufendiagramm dargestellt. |
| | | Es lassen sich in einem Diagramm bis zu zwei Knoten gleichzeitig abbilden. |
| | | Es können bis zu vier Metriken gleichzeitig visualisiert werden. |
| | | Minimum-, Maximum- und Durchschnittsknotengrad werden dargestellt. |
| | | Die Darstellungsfläche lässt sich durch das Zusammenklappen der Seitenleiste vergrößern. |
| | | Das Hovern über eine Linie gibt Informationen über Knoten, Metrikwert und Zeitpunkt. |
| FA6 | Diagramm-Konfiguration | Die im Diagramm dargestellten Knoten können ausgewählt werden. |
| | | Das zu betrachtende Zeitfenster lässt sich verkleinern, vergrößern und verschieben. |
| | | Der Moving Average kann verwendet werden und dessen Fenstergröße ist anpassbar. |
| | | Die visualisierten Knoten lassen sich ausblenden. |
| | | Es gibt eine Datenansicht des Diagramms in Textform. |
| | | Das Diagramm lässt sich als PNG herunterladen. |

Tab. 4.1.: Funktionale Anforderungen an den Temporal Metric Explorer

### 4.1.2. Nichtfunktionale Anforderungen

Für die Qualität einer Software sind die nichtfunktionalen Anforderungen ebenso entscheidend, wie funktionale Anforderungen. Die Definition ist unter Autoren nicht eindeutig und wird meist nur sehr informal beschrieben. Dieser Abschnitt bezieht sich auf die Definition von Chung et al., in der eine nichtfunktionale Anforderung eine „[...] Software Anforderung [ist], die nicht beschreibt was die Software tun wird, sondern wie die Software es tun wird [...]“ [20, S. 6]. Schemata wie NFR Framework, Quality Tree, FURPS oder ISO/IEC 9126 [21] teilen verschiedene Anforderungen in Qualitätsklassen ein. Da die Schemata aber teilweise unterschiedliche Qualitätsmerkmale betrachten, sind sie leider inkonsistent zueinander. Es ist daher nötig, einen Überblick über mehrere nichtfunktionale Anforderungen zu haben und ein Schema auf die zu entwickelnde Anwendung anzupassen.

| ID | Nichtfunktionale Anforderung | Beschreibung |
|---|---|---|
| NFA1 | Erweiterbarkeit | Es soll unkompliziert sein, den TME um neue zeitliche Metriken oder Funktionalitäten zu erweitern. |
| NFA2 | Performanz | Die Reaktionszeit des TME soll schnell sein. |
| NFA3 | Benutzerfreundlichkeit | Ein Nutzer soll den TME intuitiv nutzen können. Die Benutzeroberfläche soll einfach und verständlich gestaltet werden. |
| NFA4 | Kompatibilität | Der TME soll mit verschiedenen Browsern und Betriebssystemen benutzt werden können. |
| NFA5 | Konfigurierbarkeit | Es sollen sich viele Einstellungen treffen lassen, um dem Nutzer eine individuell zugeschnittene Analyse zu ermöglichen. |
| NFA6 | Robustheit | Fehlerhafte Benutzereingaben sollen abgefangen und behandelt werden. |
| NFA7 | Konsistenz | Die Ergebnisse von gleichen Eingaben sollen sich nicht voneinander unterscheiden. |
| NFA8 | Effizienz | Der TME soll sparsam mit Hardwareressourcen umgehen. |
| NFA9 | Ästhetik | Die Benutzeroberfläche des TME soll ansprechend wirken. |
| NFA10 | Verfügbarkeit | Ein Nutzer soll die Anwendung von überall aus verwenden können. |

Tab. 4.2.: Nichtfunktionale Anforderungen an den Temporal Metric Explorer

Der Tabelle 4.2 ist zu entnehmen, welche nichtfunktionale Anforderungen bei der Entwicklung des Temporal Metric Explorer berücksichtigt werden sollen. Da diese Eigenschaften schwer zu messen und teils subjektiv zu bewerten sind, wird im Abschnitt 6.1 näher darauf eingegangen, welche Vorkehrungen getroffen wurden, um den Anforderungen gerecht zu werden.

## 4.2. Softwarearchitektur

Die im vorhergehenden Abschnitt genannten Anforderungen an die Flexibilität des Temporal Metric Explorers, lassen die Wahl auf eine Web-Anwendung [22, S. 1-4] fallen. Aufgrund der Tatsache, dass der TGE ebenfalls als Web-Applikation entwickelt wurde, wird diese Entscheidung bestärkt. Die zentrale Bereitstellung einer solchen Anwendung ermöglicht nicht nur eine einfache Wartung und Erweiterbarkeit des Programmcode, sondern auch eine ständige Verfügbarkeit unabhängig vom verwendeten Endgerät (NFA4, NFA10).

Um mögliche Herausforderungen und Probleme bei der Implementierung frühzeitig zu erkennen, ist es wichtig, eine detaillierte Softwarearchitektur zu entwerfen. Hierbei werden grundlegende Komponenten, deren Verantwortlichkeiten und Organisation identifiziert. In Abbildung 4.1 ist die Konzipierung eines Client-/Server-Modells [23] zu erkennen. Das Client-/Server-Modell zeichnet sich durch ein einfaches Request-Response-Schema aus, indem der Client für gewöhnlich Daten vom Server anfordert und der Server nach Verarbeitung des Request die Daten zurückliefert. Dieses Modell eignet sich für die Umsetzung des TME, da dieser dem Request-Response-Schema folgt. Dabei wird eine zeitliche Metrik auf Grundlage einer Konfiguration erfragt und das berechnete Ergebnis wird zurückgeliefert.

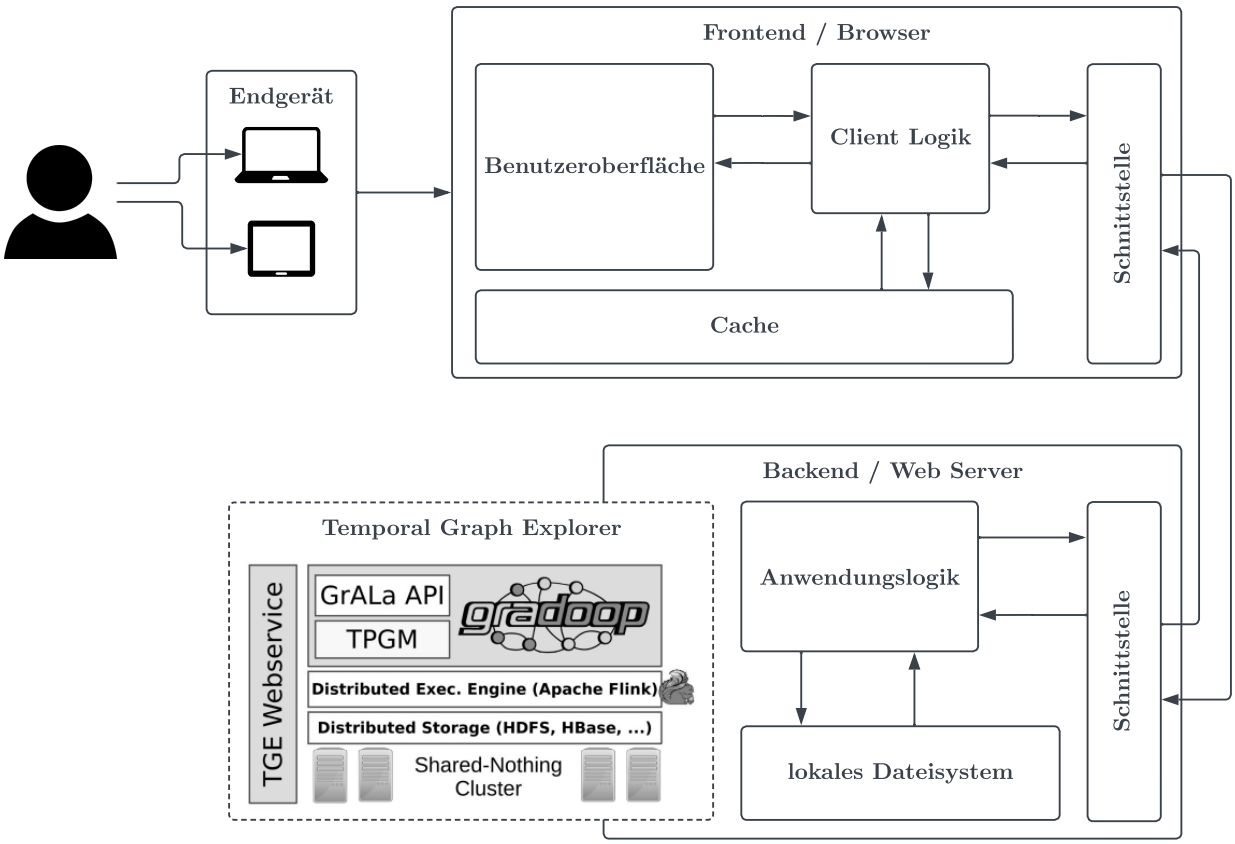



Abb. 4.1.: Softwarearchitektur des Temporal Metric Explorer

Genauer gesagt handelt es sich um ein Client-Server-Modell mit 3-Tier-Architektur [23, S. 225]. Der Client ist das Frontend, beziehungsweise der Browser. Dieser ist für die Interaktion mit dem Nutzer und die Präsentation der Metrik verantwortlich. Der Web-Server bildet zusammen mit Gradoop, Apache Flink und einem Dateisystem wie HDFS das Backend. Zusätzlich fungiert der Web-Server im Zusammenspiel mit Gradoop und Flink als Middleware, die die Anwendungslogik zur Verarbeitung und Übertragung der Daten zur Verfügung stellt. Das darunterliegende Dateisystem ist schlussendlich für die Datenspeicherung und Datenverwaltung zuständig. Diese drei Schichten bilden gemeinsam die 3-Tier-Architektur des TME.

Das Frontend umfasst alle Komponenten, die für die Visualisierung und die Interaktion mit dem Nutzer notwendig sind. Über einen Browser ist das Frontend für ein Endgerät zugänglich. Dem Nutzer wird eine grafische Benutzeroberfläche geboten, die Eingaben und Interaktionen entgegennimmt und sie an die Client-Logik weiterleitet. Da die Benutzeroberfläche weitgehend responsiv gestaltet wird, ist es möglich, den TME in unterschiedlichen Bildschirmgrößen zu nutzen. Hierbei sei jedoch zu erwähnen, dass die Anzeige erst ab mittleren Bildschirmgrößen, wie bei Tablets, optimiert wird, da die Analyse von Diagrammen auf kleineren Geräten nicht mehr zielführend ist. Die Client-Logik sorgt für eine korrekte Darstellung der Metrik in der Benutzeroberfläche und umfasst die Verarbeitung, Kontrolle und Übertragung der Daten. Da die aufkommenden Datenmengen groß sein könnten, ist es nicht effizient, ständig alle Daten vom Backend abzufragen und über das Netzwerk zu verschicken. Daher soll der Browser Cache als Speicher genutzt werden, um Metriken zwischenzuspeichern. Eine Schnittstelle zur Verbindung mit dem Backend wird ebenfalls benötigt, um zwischen den unterschiedlichen Softwaresystemen zu kommunizieren.

Das Backend muss ebenfalls mit einer Schnittstelle erweitert werden, die eine Vorschrift zur Kommunikation mit dem Frontend vereinbart. Die Verarbeitung der vom Frontend erhaltenden Daten übernimmt die Anwendungslogik. Diese interagiert ebenfalls mit dem lokalen Dateisystem, um Metadaten über die vorliegenden Graph-Datensätze zu erfahren und ruft Gradoop zur Berechnung einer Metrik auf. Zusätzlich formuliert die Anwendungslogik aus den errechneten Daten eine geeignete Response für das Frontend.

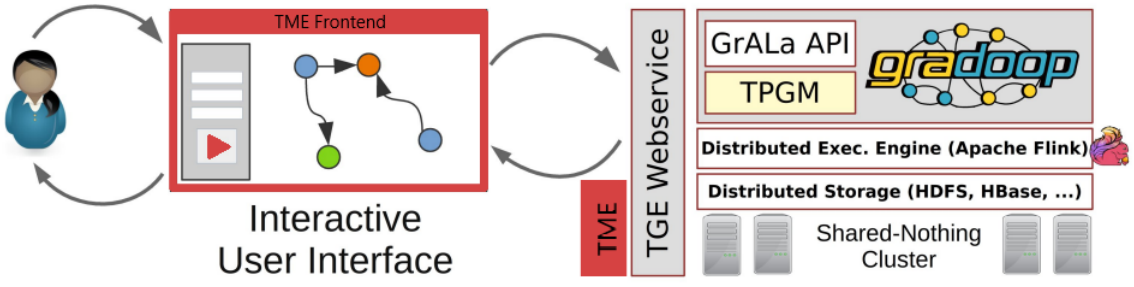
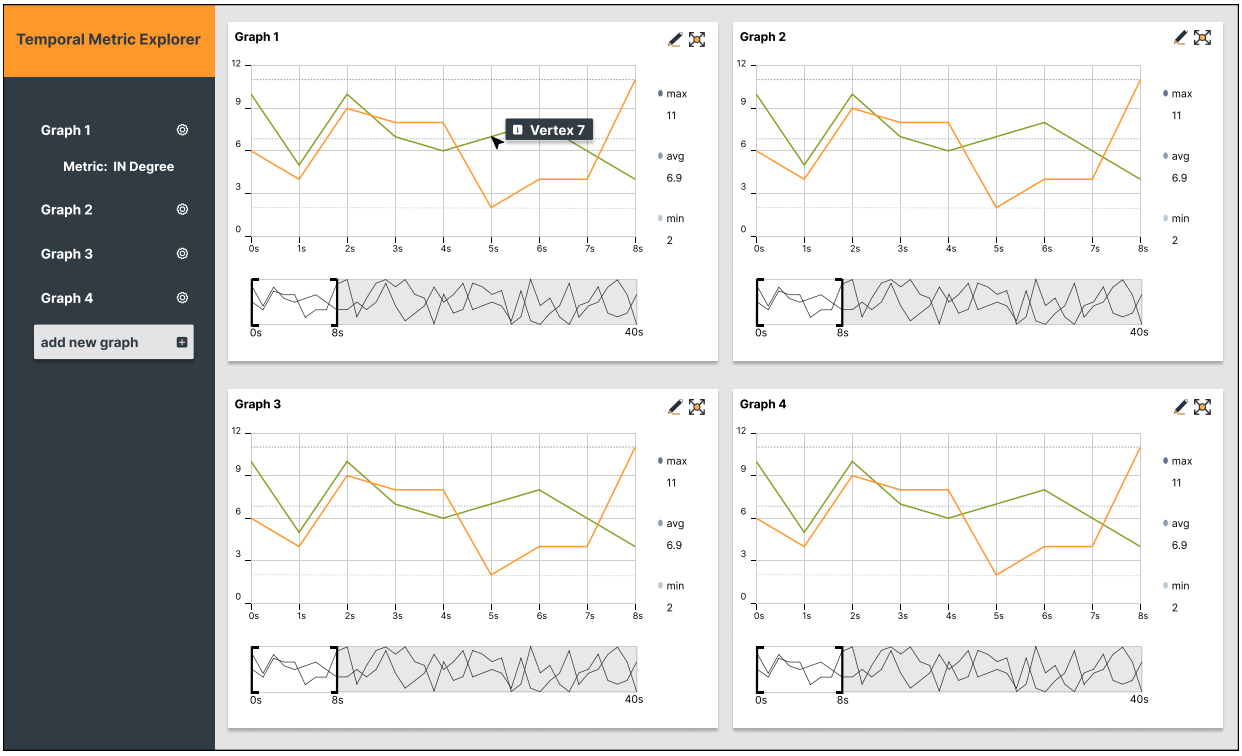



Abb. 4.2.: Systemarchitektur des Temporal Metric Explorer in Anlehnung an [15, S. 3]

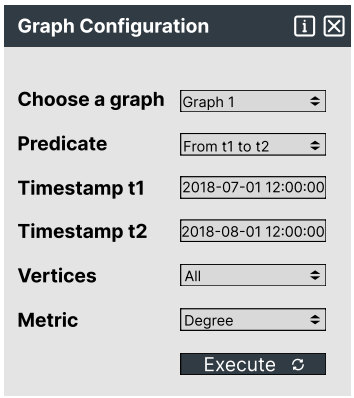
Wie Abbildung 4.2 zu entnehmen ist, soll der Temporal Metric Explorer eine Erweiterung des bereits bestehenden Temporel Graph Explorers darstellen. Der TME baut auf dem TGE-Webservice auf und bietet eine vollständig neue Benutzeroberfläche zur Analyse. Gradoop und Apache Flink sollen dabei weiterhin den Grundbaustein für die verteilte Berechnung von zeitlichen Graph Metriken bilden.

## 4.3. Design-Entwurf der Benutzeroberfläche

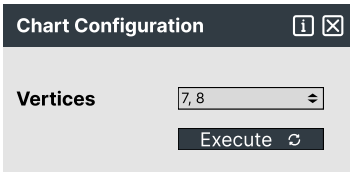
Neben der Erhebung technischer Anforderungen, ist der visuelle Entwurf ein wichtiger Schritt in der Konzeption einer Softwareanwendung. Bereits im Design-Entwurf können mögliche Schwierigkeiten im Umgang mit der Benutzeroberfläche identifiziert und behoben werden. Abbildung 4.3 zeigt die prototypische Gestaltung des TME und vermittelt damit eine Vorstellung von der zukünftigen Benutzeroberfläche.




(a) Landing Page




(b) Graph-Konfigurationsfenster




(c) Diagramm-Konfigurationsfenster

Abb. 4.3.: Erster Design-Entwurf der graphischen Benutzeroberfläche des TME

Dieses Design bietet eine Vorlage zur visuellen Gestaltung der Benutzeroberfläche. Da der Entwicklungsprozess jedoch sehr dynamisch verläuft, ist es nicht ungewöhnlich, dass an dem ersten Entwurf noch Änderungen vorgenommen werden. Dementsprechend kann die finale Web-Anwendung von diesem Design abweichen.

# 5. Realisierung des Temporal Metric Explorer

Dieses Kapitel beschreibt, wie das in Kapitel 4 entworfene Konzept in Form eines Prototyps umgesetzt werden konnte. Dafür werden zunächst die verwendeten Technologien vorgestellt, bevor anschließend erläutert wird, wie deren Zusammenspiel eine Realisierung des Temporal Metric Explorers ermöglichte. Hierbei wird zur besseren Übersicht in die einzelnen Softwarekomponenten unterteilt.

## 5.1. Technologieauswahl

Bevor mit der Implementierung begonnen werden konnte, mussten für den Entwurf geeignete Technologien gewählt werden. Dieser Abschnitt stellt die verwendeten Technologien vor, die eine Realisierung der Web-Anwendung ermöglicht haben.

### 5.1.1. Benutzeroberfläche

Um die Benutzeroberfläche modern und interaktiv zu gestalten, wird HTML, CSS und Bootstrap[1] verwendet. Diese Technologien werden durch die gängigsten Browser unterstützt und sind damit plattformunabhängig (NFA4).

Den Grundbaustein einer Webseite bildet eine Beschreibungssprache. Der De-facto-Standard ist die Hypertext Markup Language, kurz HTML [24]. HTML spezifiziert die Struktur einer Webseite und formuliert Regeln für die Darstellung von Inhalten. Die Entwicklung begann im Jahr 1991 und wird durch wachsende Anforderungen stetig vorangetrieben. Die aktuelle Version HTML5 [25] bietet im Vergleich zu ihren Vorgängern unter anderem eine Erweiterung der Syntax durch semantische Elemente, eine bessere Performanz und Multimedia-Unterstützung. Die Struktur einer Webseite wird durch HTML in einzelne Elemente untergliedert. Jedem Element ist eine spezifische Aufgabe zugeordnet. Dies kann die Darstellung einer Überschrift sein oder ein Textfeld für eine Benutzereingabe. Alle Elemente werden zusammen im Document Object Model als Baumstruktur gespeichert.

Obwohl es möglich ist, die visuelle Darstellung einer Webseite über HTML zu beschreiben, sollten Inhalt und Darstellung voneinander getrennt werden. Dies folgt dem Softwareentwicklungsprinzip “Trennung von Belangen“ [26] und sorgt für eine verbesserte Wartbarkeit und Anpassbarkeit des Programmcodes. Zur Darstellung eignen sich Cascading Style Sheets (CSS) [24]. CSS übernimmt die optische Gestaltung der Webseite, indem über Selektoren einzelne Elemente oder ganze Elementgruppen in Bezug auf Größe, Farbe, Form, Layout usw. formatiert werden können. Auch hier fällt die Wahl auf die aktuellste Version CSS3.

[1] https://getbootstrap.com/ (abgerufen am 01.03.2023)

Neben der Verwendung von HTML und CSS ist es sinnvoll, ein Frontend-Framework wie Bootstrap [27] zu nutzen. Bootstrap ist Open Source und erleichtert durch vorgefertigte Komponente die Entwicklung und sorgt damit zeitgleich für eine konsistente Struktur. Das enthaltene Grid-System erlaubt eine flexible Anordnung der HTML-Elemente in Reihen und Spalten und eignet sich zur Umsetzung einer responsiven Anwendung. Da der TME auch auf unterschiedlichen Bildschirmgrößen korrekt angezeigt werden soll, ist dies ein nützlicher Vorteil. Zusätzlich bietet Bootstrap Buttons, Modale, Pop-Overs und Spinner. Dabei sind die Komponenten bereits in Funktion und Optik vorgefertigt und müssen lediglich über HTML eingebunden werden. Ein Modal legt sich über die aktuelle Ansicht und fordert eine Eingabe, was bei der Graph und Diagramm-Konfiguration helfen kann. Ein Spinner eignet sich zur Visualisierung von Ladevorgängen. Zur Implementierung wird die aktuelle Version 5 von Bootstrap verwendet, da hier der größte Funktionsumfang geboten wird.

### 5.1.2. Client-Logik

Die clientseitige Programmlogik wird in JavaScript geschrieben und nutzt Bibliotheken wie jQuery[2] und Apache ECharts[3], um die Entwicklung effizienter und die Funktionalität vielseitiger zu gestalten. Des Weiteren wird für leistungsfähige Dropdowns das Plugin VirtualSelect[4] verwendet. Auch hier unterstützen die gängigsten Browser die Software (NFA4).

JavaScript [28] wurde 1995 von dem Softwareunternehmen Netscape mit der Absicht entwickelt, Webseiten dynamischer zu gestalten. Über das Einbinden von Skripten sollten die sonst statischen HTML-Ansichten interaktiv gestaltbar werden. Dabei sollte es auch Personen, die keine tieferen Programmierkenntnisse haben ermöglicht werden, unkompliziert Skripte zu schreiben. Unter anderem durch seine simple Syntax konnte sich JavaScript gegen komplexere Sprachen wie Perl, Python und Java in der Webentwicklung durchsetzen und wurde in das standardisierte Softwarepaket von Browsern integriert. Durch das Softwarepaket, das die Interoperabilität verschiedener Browser definiert, ist in jedem modernen Browser eine JavaScript-Engine installiert, die es erlaubt JavaScript Code zur Laufzeit in ausführbaren Maschinencode zu übersetzen. Ursprünglich unter dem Namen LiveScript vorgestellt, folgte frühzeitig die Umbenennung in JavaScript, da aus Marketingzwecken und aufgrund medialer Aufmerksamkeit eine enge Verbindung zur beliebten Programmiersprache Java aufgebaut wurde. Doch neben einer ähnlichen Syntax und einem objektorientierten Programmierparadigma sind beide Sprachen technisch grundlegend verschieden.

JavaScript Code [29] kann über das HTML *script*-Element oder über Elementattribute direkt in das HTML-Dokument eingebunden werden. Sollten die JavaScript-Methoden komplexer sein, ist es auch möglich eine separate JS-Datei zu erstellen, die über das *src*-Attribut im *script*-Element referenziert werden kann. Es handelt sich um eine dynamisch typisierte Sprache, was bedeutet, dass Variablen nicht mit einem Datentyp deklariert werden müssen, sondern dieser automatisch zur Laufzeit bestimmt und zugewiesen wird. Ebenso übernimmt JavaScript die Umwandlung, denn Daten können, wenn nötig, von einem Datentyp in einen anderen überführt werden. Zur Verfügung stehen primitive Datentypen, bestehend aus Number, String und Boolean, sowie die beiden

[3] https://echarts.apache.org/ (abgerufen am 01.03.2023)
[4] https://sa-si-dev.github.io/virtual-select/ (abgerufen am 01.03.2023)

trivialen Datentypen null und undefined. Des Weiteren gibt es den komplexen Datentyp Object. Dieser besteht aus einer Komposition von Datentypen wie einer Sammlung von primitiven Daten oder wiederum anderen Objects. Ein Sonderfall von Objects stellen die Arrays und Functions dar. Obwohl sie fundamental das gleiche ausdrücken, werden sie aufgrund ihres unterschiedlichen Verhaltens meist als eigenständige Datentypen angesehen. Arrays bezeichnen eine Liste von geordneten und nummerierten Daten, wohingegen Objects aus ungeordneten Daten bestehen, denen jeweils ein Bezeichner zugewiesen ist. Objects, die mit einem ausführbaren Code ausgestattet sind und über ihren Aufruf eine Operation durchführen, werden Functions genannt. Um den Umgang mit Functions zu erleichtern, liefert JavaScript hierfür eine spezielle Syntax. Die Verwendung von Objects ermöglicht eine objektorientierte Programmierung, die sich jedoch von anderen Sprachen wie Java unterscheidet. Denn Objects werden nicht aus Klassen instantiiert, sondern durch das Klonen von bereits existierenden Prototypen erstellt. Jedes Object hat einen identischen Prototypen, der als Grundlage für die Erstellung weiterer Objects mit ähnlichen Eigenschaften dient.

Das bereits angesprochene Document Object Model (DOM) [29] wird durch den Browser erstellt und repräsentiert die Struktur des HTML-Dokuments. Die einzelnen HTML-Elemente werden im DOM in ihrer hierarchischen Anordnung als Baumstruktur gespeichert und sind über eine Programmierschnittstelle zugänglich. Mittels JavaScript können somit HTML-Elemente über das DOM angesprochen und manipuliert werden. Durch das Hinzufügen, Löschen und Verändern einzelner HTML-Elemente im DOM wird somit eine dynamische Anpassung des HTML-Dokuments ermöglicht.

Die wachsende Beliebtheit von JavaScript brachte die Entwicklung neuer Bibliotheken mit sich. JQuery [30] ist eine solche JavaScript Bibliothek, die das Ziel verfolgt, schneller und einfacher programmieren zu können. Durch eine kompakte und intuitive Syntax erleichtert jQuery das Schreiben von umfangreichen JavaScript-Anweisungen und sorgt zeitgleich für eine zeitsparende Entwicklung. Funktionalitäten wie DOM-Manipulation, Event-Handling und AJAX lassen sich in jQuery mit wenigen Zeilen Code ausdrücken, wohingegen JavaScript teilweise weitaus mehr Zeilen benötigt. Ein anschauliches Beispiel ist durch Bibeault et al. in der folgenden DOM-Traversierung gegeben [30]:

```
1  var checkedValue;
2  var elements = document.getElementsByTagName('input');
3  for (var i = 0; i < elements.length; i++) {
4    if (elements[i].type === 'radio' &&
5        elements[i].name === 'some-radio-group' &&
6        elements[i].checked) {
7      checkedValue = elements[i].value;
8      break;
9    }
10 }
```

Abb. 5.1.: JavaScript Code ohne jQuery. [30]

In Abbildung 5.1 wird mittels JavaScript das DOM nach einem ausgewählten Radio-Button durchsucht. Um zwischen anderen Radio-Buttons zu unterscheiden, soll nur jener Button adressiert werden, der der Gruppe `some-radio-group` angehört. Von diesem Button wird anschließend der zugehörige Wert erfragt, um letztendlich die ausgewählte Benutzereingabe in der Variable `checkedValue`

zu speichern. Dieses Vorgehen lässt sich mit jQuery auch wie folgt ausdrücken:

```
1  var checkedValue =
2    $('input:radio[name="some-radio-group"]:checked').val();
```

Abb. 5.2.: Zu 5.1 äquivalenter JavaScript Code mit jQuery. [30]

Ohne hierbei genauer auf die Syntax einzugehen, ist in Abbildung 5.2 dennoch klar ersichtlich, dass jQuery dabei hilft, kompakteren Code zu schreiben. Aus diesem Grund wird jQuery bei der Entwicklung des TME als ergänzende Bibliothek zu JavaScript verwendet. Dabei wird die Version 1.11. gewählt, da der TGE ebenfalls diese nutzt und Fehler aufgrund unterschiedlicher Versionen vermieden werden sollen.

Um die zeitlichen Graph Metriken zu visualisieren, wird das Open Source JavaScript Framework Apache ECharts [31] verwendet. Die Motivation hinter der Entwicklung von ECharts deckt sich mit den in Abschnitt 4.1 genannten Anforderungen, denn ECharts bietet eine benutzerfreundliche, interaktive und hochperformante Plattform zur webbasierten Visualisierung. Dafür unterstützt ECharts eine Reihe von Optionen, um bereits vorgefertigte Diagrammtypen flexibel auf die eigenen Ansprüche zu konfigurieren. Die zugrundeliegende ZRender-Engine basiert auf HTML5 Canvas und sorgt für eine hohe Performanz in der Darstellung der Daten. Zusätzlich stehen viele Funktionalitäten zur Verfügung, um mit den Diagrammen zu interagieren. So gibt es unter anderem ein Tool zum Zoomen, Legenden für Linienkurven und Informationsanzeigen beim Hovern mit der Maus. Neben den bereitgestellten Komponenten, können auch eigene Komponenten implementiert werden, was in Kombination eine große Vielfalt an Analysemöglichkeiten bietet. Im TME wird die aktuellste Version 5 von Apache ECharts verwendet.

### 5.1.3. Schnittstelle Frontend

Damit zwei Systeme mit unterschiedlichen Technologien wie das Frontend und Backend miteinander interagieren können, ist eine entsprechende Programmierschnittstelle erforderlich. Eine solche Schnittstelle wird Application Programming Interface, kurz API, genannte und definiert präzise Regeln für eine einheitliche Kommunikation. Auf der Client-Seite muss eine API geschaffen werden, über die korrekte Requests an den Server formuliert und Responses empfangen werden können. Dazu soll JSON als Datenformat verwendet werden und die Datenübertragung mit Hilfe von AJAX erfolgen.

Traditionelle Web-Anwendungen waren hinsichtlich ihrer Interaktivität und Performanz typischen Desktop-Anwendungen unterlegen [32]. Das war hauptsächlich darauf zurückzuführen, dass wenn neue Inhalte oder Daten benötigt wurden, das gesamte HTML-Dokument vom Server erneut generiert und anschließend neu geladen werden musste. AJAX [32] steht für Asynchronous JavaScript and XML und umgeht dieses Problem durch die Verwendung eines neuen Konzepts, das Web-Anwendungen wie Desktop Anwendungen wirken lässt. Mit AJAX verläuft die Kommunikation mit dem Server asynchron und im Hintergrund. Für den Datenaustausch steht eine XMLHTTP-Schnittstelle zur Verfügung, über die gemäß dem Hypertext Transfer Protocol (HTTP) Daten über

das Web versendet oder empfangen werden können. Dabei wird in JavaScript ein XMLHTTP-Request-Objekt erstellt, dass Informationen über den zu sendenden Request an den Server enthält. Dieses Objekt wird mittels eines XMLHTTP Requests an die Server API gesendet und vom Server verarbeitet. Anschließend formuliert der Server einen entsprechenden Response und schickt sie mit einem XMLHTTP Response zurück. Die erhaltenen Daten werden dann in JavaScript verarbeitet und über das DOM in das HTML-Dokument eingebunden. Durch die Anpassung des DOM muss nicht die gesamte Webseite neu geladen werden, sondern sie verändert sich dynamisch. Dieses asynchrone Laden der Daten erspart dem Benutzer der Anwendung Wartezeit, da er weiterhin bis zum Vorliegen der Daten mit der Web-Anwendung interagieren kann. Zusätzlich verbessert AJAX die Performanz der Web-Anwendung, da nicht ständig die gesamte Webseite neu generiert, übertragen und geladen werden muss, sondern nur die Daten die auch benötigt werden.

Die Verwendung von jQuery erleichtert die Erzeugung eines Request über AJAX mit einer eigens dafür angefertigten Methode. Hierbei müssen lediglich die grundlegenden Parameter genannt werden und jQuery kümmert sich um die korrekte Formulierung des Requests.

Um Daten zwischen unterschiedlichen Plattformen zu senden, müssen sie in einem klar definierten Format vorliegen. Ein populäres und textbasiertes Datenformat zur Übertragung von Daten über das Web ist die JavaScript Object Notation (JSON) [33]. Da JSON auf der Syntax von JavaScript-Objekten basiert, lässt sich es sich leicht in JavaScript Code integrieren. Es lässt sich nicht nur aufgrund seiner kompakten Größe effizient von Maschinen auswerten, sondern die simple Struktur ist auch gut für den Menschen verständlich. Das Erstellen oder die Auswertung von Daten im JSON-Format wird dadurch erleichtert. Ursprünglich war AJAX für die Übertragung in XML ausgelegt, doch mittlerweile werden auch andere Datenformate wie JSON unterstützt [34, S. 1174f]. Damit sind JSON und AJAX kompatibel und es bedarf keiner weiteren Umwandlung der Daten.

### 5.1.4. Schnittstelle Backend

Beim Server muss eine Verbindungsstelle zum Frontend geschaffen werden, über die es möglich ist, Requests vom Client zu empfangen und eine passende Response zu versenden. Dazu wird mit Hilfe des Frameworks Jersey[5] eine API implementiert, die dem REST-Architekturstil entspricht.

REST steht für Representational State Transfer [35] und beschreibt eine Richtlinie für die Kommunikation zwischen Systemen über das Web. Dabei werden unter REST etablierte Technologien zu einer Architektur zusammengefasst, die deren Vorteile kombiniert und somit eine bestmögliche Grundlage für Web-Anwendungen bietet. REST konnte sich gegen andere vergleichbare Architekturstile, wie SOAP oder WSDL durchsetzen und ist heute eine dominierende Vorlage zur Entwicklung von Programmierschnittstellen über das Web. Eine API, die den Prinzipien von REST folgt, wird auch RESTful API genannt. Die fundamentalsten Anforderungen an eine solche RESTful API hat Alex Rodriguez in einem Paper über die Grundlagen von REST zusammengefasst [35]:

- Nutze die HTTP-Methoden explizit
- Sei zustandslos

[5] https://eclipse-ee4j.github.io/jersey/ (abgerufen am 01.03.2023)

- Schaffe URIs, die wie eine Verzeichnisstruktur aussehen
- Übertrage XML, JSON oder beides

Das erste Prinzip besagt, dass alle Aktionen einer RESTful API ausschließlich über die HTTP-Methoden GET, POST, PUT und DELETE zu realisieren sind [35]. Dabei sollen diese Methoden ihren Aufgaben entsprechend verwendet werden, wie es in der Definition RFC 2616 des Protokolls festgelegt ist. So sollen Ressourcen mit GET abgerufen und mit POST auf dem Server erzeugt werden. PUT soll zum Verändern der Ressourcen eingesetzt werden und DELETE zum Entfernen. Die korrekte Verwendung der HTTP-Methoden sorgt nicht nur für einen guten Stil, sondern vermeidet auch ungewollte Nebeneffekte. Daher ist es wichtig, dieses Prinzip zu befolgen.

Das zweite Prinzip fordert eine Zustandslosigkeit der Requests an den Server, um die Web-Anwendung performant zu halten [35]. Jeder Request soll unabhängig von anderen Requests sein und alle Informationen und Daten enthalten, die der Server zur Verarbeitung benötigt. Der Server soll zur Bearbeitung des Requests nicht zusätzlich noch einen Zustand oder Anwendungskontext abrufen müssen. Ein Beispiel zum Verständnis ist das Abfragen der nächsten Seite einer Suchmaschine. Ein zustandsbehafteter Request würde den Server ohne genaue Informationen einfach nach der nachfolgenden Seite fragen. Damit der Server diese Seite liefern kann, muss er die aktuelle Seite des Clients bereits kennen. Anschließend muss er, ausgehend vom gespeicherten Zustand, die nächste Seite berechnen und an den Client übermitteln. Dem entgegen steht eine zustandsloser Request, wobei der Client seinen aktuellen Zustand selbst verwaltet. Der Client sorgt sich eigenständig um die Berechnung der nächsten Seitenzahl und erfragt die dazugehörige Seite beim Server. Der Server muss nun lediglich die Seite liefern, die im Request gefordert ist und benötigt keine weiteren Informationen über den Zustand des Clients. Die Zustandslosigkeit sorgt für eine serverseitige Entlastung und teilt die Verantwortungen besser auf. Der Client soll sich selbst um seinen Anwendungszustand kümmern und der Server lediglich Responses auf vollständige Requests generieren.

Um das nächste Prinzip verstehen zu können, muss zunächst geklärt werden, was URIs sind. Über Uniform Resource Identifier (URI) [36] werden abstrakte oder physikalische Ressourcen nach einer einheitlichen Syntax eindeutig identifiziert. Dabei kann weiter in Uniform Resource Locator (URL) und Uniform Resource Name (URN) klassifiziert werden, wobei URL den genauen Ort der Ressource beschreibt und URN deren eindeutigen Bezeichner. Die Struktur von URI ist in ein Schema und einer Hierarchie unterteilt. Das Schema legt eine Spezifikation von Identifikatoren fest, was zum Beispiel das HTTP-Protokoll sein kann. Der hierarchische Teil beschreibt einen Pfad zur Ressource innerhalb des Schemas, wobei in jedem Schritt weiter spezifiziert wird. Nach REST soll die Struktur der URIs in einer API einem Verzeichnis ähneln [35]. Dabei sollen die Ressourcen eines Web-Server zunächst in einem allgemeinen Pfad miteinander in Beziehung stehen und davon ausgehende Teilpfade zu spezifischen Web-Server-Aufgaben führen. Diese Hierarchie macht eine RESTful API intuitiv nutzbar und leicht verständlich, da die Struktur eine selbsterklärende Dokumentation bietet.

Die Übertragung der Daten in XML, JSON oder beides stellt ein weiteres Grundprinzip von REST dar [35]. Die Daten die zwischen Server und Client ausgetauscht werden, sollen demnach in JSON oder XML vorliegen. Diese beiden Datenformate veranschaulichen Relationen zwischen den Daten und lassen sich durch einen Benutzer leicht lesen. Ebenfalls soll der Client von den HTTP Accept

Headern Gebrauch machen, indem er über einen MIME Type [37] angeben kann, welches Datenformat er vom Server erwartet. Die MIME-Typen sind standardisiert und bestehen aus einem Haupttyp und einem Untertyp. Der Haupttyp beschreibt die Daten grob, wie etwa text oder application und der Untertyp definiert ein spezifisches Format wie beispielsweise HTML oder JSON. Durch die Verwendung von MIME-Typen im Accept Header kann der Server die Daten im passenden Format versenden, wodurch Clients mit unterschiedlichen Technologien die Dienste des Web-Servers nutzen können.

Die Entwicklung einer REST-konformen API wird über JAX-RS [38] spezifiziert. JAX-RS steht für Jakarta-RESTful-Web-Service und stellt eine Reihe von Regeln auf, mit denen sich RESTful APIs in Java umsetzen lassen. Eine Referenzimplementierung zu JAX-RS stellt das Open Source Framework Jersey [38] dar. Jersey bietet ein Grundgerüst zur konkreten Umsetzung der in JAX-RS definierten Anforderungen und erleichtert somit die Entwicklung von RESTful APIs in Java. Über Annotationen bei Java Klassen und Methoden wird Jersey mitgeteilt, dass es sich um Jersey-Ressourcen innerhalb einer RESTful API handelt. Jersey kümmert sich dann um die Verwaltung dieser Ressourcen und stellt sie als Web-Dienste für einen Client bereit. Über die @Path Annotation wird Klassen und Methoden eine URI zugewiesen, die nötig ist, um eine Ressource eindeutig zu identifizieren. Mit @GET, @POST, @PUT und @DELETE können Ressourcen entsprechende HTTP-Methoden zugewiesen werden, damit Requests korrekt verarbeitet werden können. Des Weiteren gibt es die @Produces Annotation, mit der unterschiedliche Datenformate für den Response festgelegt werden können. Dies ist vor allem dann wichtig, wenn in einem HTTP Request über den Accept Header ein bestimmter MIME Type im Response gefordert wurde. Neben Annotationen ist POJO-Mapping eine weitere nützliche Funktionalität, die Jersey zur Verfügung stellt. Dabei werden JSON-Objekte aus einem HTTP Request in einfache Java-Klassen, also Plain Old Java Objects (POJO), überführt. Hierzu wird nach vordefinierten POJOs gesucht, deren Attribute sich denen des JSON-Objektes zuordnen lassen. Bei einer Übereinstimmung wird das JSON-Objekt automatisch in eine Java-Klasse konvertiert und muss somit nicht manuell eingelesen werden. Da der TME die vorhandene API des TGE-Webservice erweitert, wird die Jersey-Version 1.19 verwendet.

### 5.1.5. Anwendungslogik

Die Anwendungslogik wertet die vom Client empfangenen Daten aus und nutzt die von Gradoop bereitgestellten Methoden zur Berechnung der zeitlichen Graph Metrik. Anschließend wird ein passender Response für den Client generiert. Dazu wird die Programmiersprache Java[6] verwendet.

Java [39] wurde ursprünglich entwickelt, um eine plattformunabhängige Alternative zu C++ für die Entwicklung interaktiver Fernsehgeräte zu bieten. Es stellte sich heraus, dass Java auch über die Eigenschaften einer leistungsstarken serverseitigen Programmiersprache verfügt. Dies führte dazu, dass es bei der Entwicklung von Web-Anwendungen immer beliebter wurde und auch heutzutage unter Stand-alone-Anwendungen sehr populär ist. Java kombiniert bewährte Konzepte und vereinigt damit Vorteile verschiedener Programmiersprachen. So basiert Java zum Beispiel auf dem Klassenkonzept und verfolgt damit den objektorientierten Ansatz. Auch die automatische Garbage Collection, die mögliche Ausnahmbehandlung und das bewusste Verzichten auf Zeiger und einen

[6] https://www.oracle.com/de/java/ (abgerufen am 01.03.2023)

Präprozessor haben Java zum Erfolg verholfen. Des Weiteren wird Java Code zuerst von einem Java Compiler in Bytecode kompiliert und im Gegensatz zu anderen Sprachen anschließend noch in einer Laufzeitumgebung, der Java Virtual Machine, interpretiert. Durch dieses Vorgehen kann eine Plattformunabhängigkeit und ein hohes Maß an Sicherheit geboten werden. Mit der zusätzlichen Verwendung eines Just-In-Time Compilers kann auch die Ausführungsgeschwindigkeit auf ein Niveau gesteigert werden, was sich in vielen Fällen mit C vergleichen lässt. Daher bietet Java eine optimale Grundlage für die serverseitige Entwicklung und wird somit beim TME verwendet. Um mit Gradoop bestmöglich kompatibel zu sein, wird die Java-Version 1.8 genutzt.

Die Anwendung wird auf einem Grizzly HTTP-Server laufen, der von Jersey zur Verfügung gestellt wird. Dieser macht die RESTful API für das Web zugänglich und verwaltet die Netzwerkkommunikation. Das soll an dieser Stelle jedoch lediglich erwähnt und nicht weiter vertieft werden. Auch auf die vom TGE und Gradoop verwendeten Technologien wird nicht näher eingegangen, da diese im Rahmen der vorliegenden Arbeit ebenfalls nicht implementiert werden.

### 5.1.6. Tabellarische Übersicht

Die nachfolgende Tabelle 5.1 gibt einen zusammenfassenden Überblick über die Technologien, mit denen der Temporal Metric Explorer realisiert wird. Dabei ist jedoch anzumerken, dass ausschließlich jene Technologien aufgeführt sind, die tatsächlich zur Implementierung verwendet werden. Nicht selbst implementierte Technologien, denen beispielsweise Gradoop zugrunde liegt und die zur Funktionsweise des TME beitragen, wurden dabei außen vor gelassen.

| Softwarekomponente | Technologieauswahl | Version |
|---|---|---|
| Benutzeroberfläche | HTML | 5 |
| | CSS | 3 |
| | Bootstrap | 5.0.2 |
| Client-Logik | JavaScript | ES2022 |
| | Apache ECharts | 5.0.2 |
| | VirtualSelect | 1.0.3 |
| Client-Logik /Schnittstelle Frontend | jQuery | 1.11.3 |
| Schnittstelle Backend | Jersey | 1.19.3 |
| Anwendungslogik | Java SE Development Kit | 1.8 |

Tab. 5.1.: Übersicht der Technologieauswahl des TME

## 5.2. Implementierung

Nachdem es ein konkretes Konzept zur Umsetzung und eine dafür geeignete Technologieauswahl gab, konnte mit der Implementierung der Web-Anwendung begonnen werden. In diesem Abschnitt wird daher das Zusammenspiel der Technologien erklärt und näher auf einzelne Programmierabschnitte eingangen. Die gesamte Implementierung und Dokumentation ist der CD im Anhang zu entnehmen.

### 5.2.1. Vorgehensweise

Um das Schreiben von Quellcode zu erleichtern, wurde JetBrains IntelliJ[7] als Entwicklungsumgebung gewählt. IntelliJ [40] selbst wird auf der eigenen Webseite als marktführende Java IDE beworben, mit der die Entwicklung von Software effizienter gestaltet werden kann. Code-Vervollständigung, weitreichende Debug-Optionen oder automatisches Refactoring sollen hier nur beispielhaft für die Menge an Funktionalitäten genannt werden, über die IntelliJ den Entwicklungsprozess für den Programmierer erleichtert. Zusätzlich werden neben Java auch JavaScript, HTML und CSS unterstützt, wodurch sich IntelliJ als Entwicklungsumgebung für den Temporal Metric Explorer eignet.

Die Implementierung erfolgte in einem iterativen und zugleich inkrementellen Prozess, wodurch sich das Vorgehen der agilen Softwareentwicklung zuordnen lässt. Ausgehend von der Grundfunktion wurden in kleinen Schritten neue Funktionen hinzugefügt, während regelmäßig die gesamte Software in Betracht gezogen wurde. Somit konnten die einzelnen Softwarekomponenten im Hinblick auf die finale Anwendung stetig verbessert werden. Um bei dieser schrittweisen Anpassung des Codes Fehler zu vermeiden, wurde das GitLab[8] der Universität Leipzig als Versionsverwaltungssystem verwendet. In GitLab konnten Zwischenstände der Software in einer Historie gespeichert werden. Sobald Fehler aufgrund von Änderungen am Quellcode auftraten, konnten diese durch die Verwendung eines älteren Standes rückgängig gemacht werden.

### 5.2.2. Anwendungslogik

Um die Graph Metriken in einem Frontend visualisieren zu können, müssen die erforderlichen Daten zunächst vom Backend erzeugt werden. Daher wurde mit der Implementierung der Anwendungslogik begonnen, denn diese ist für die korrekte Berechnung der zeitlichen Graph Metriken zuständig.

Da unterschiedliche Metriken jeweils eine andere Berechnung erfordern, wurde zunächst ein zentraler Ausgangspunkt geschaffen. Innerhalb des TGE-Webservice wurde somit die Java-Methode `getMetric` implementiert, die den temporalen Graph erzeugt, von dem die Metrik bestimmt werden soll. Da der Benutzer des TME Voreinstellungen treffen kann, wird der Graph auf diese angepasst. Als Aufrufparameter für die Methode wird ein Konfigurationsobjekt benötigt, das unter anderem den Namen für einen Datensatz, ein Zeitintervall und eine Menge von Knoten enthält. Zu Beginn

[7] https://www.jetbrains.com/idea/ (abgerufen am 01.03.2023)
[8] https://git.informatik.uni-leipzig.de/ (abgerufen am 01.03.2023)

wird eine von Gradoop zur Verfügung gestellte `TemporalGraph`-Instanz erzeugt. Hierbei muss zuvor, auf Grundlage des in der Konfiguration gewählten Graphennamens, im lokalen Dateisystem nach vorhandenen Daten gesucht werden.

```
1  if(request.getFilters().length > 0){
2        temporalGraph = temporalGraph.edgeInducedSubgraph(new EdgeFilter<>(request.
             getFilters()));
3     }
```

Abb. 5.3.: Graphreduzierung nach Knotenfilterung

Abbildung 5.3 zeigt, wie im TME Graphen reduziert werden. Sollte der Request Filterknoten enthalten, dann wird ein kanteninduzierter Teilgraph erzeugt. Der Teilgraph bildet eine Teilmenge des ursprünglichen Graphen, bestehend aus den zu filternden Knoten, den dazu inzidenten Kanten sowie allen Knoten, die an den inzidenten Kanten anliegen. Durch diese Reduzierungen des ursprünglichen Graphen wird die Berechnungsdauer der zeitlichen Graph Metrik für die Filterknoten auf ein Minimum verringert (NFA2). Denn Operationen müssen nicht mehr auf dem gesamten Graphen ausgeführt werden, sondern nur noch auf einem für die Metrik relevanten Ausschnitt. Um überflüssige Berechnungen frühzeitig zu vermeiden, erfolgt die Reduzierung direkt nach der Instantiierung des Graphen. Schließlich wird der Graph noch auf das konfigurierte Zeitintervall beschränkt und anschließend an eine Methode zur Berechnung der Metrik weitergeleitet. Durch einen zentralen Aufruf der unterschiedlichen Metrik-Operatoren, lassen sich unkompliziert neue Metriken hinzufügen. Die Struktur der Anwendung muss dabei nicht verändert werden, was eine spätere Erweiterung um Funktionalitäten erleichtert (NFA1).

Bereits implementiert wurden die Grad Metriken Eingangs-, Ausgangs- und einfacher Knotengrad, sowie Minimum-, Maximum- und Durchschnittsknotengrad. Deren Berechnung erfolgt in der Methode `metricDegree`. Hierbei wird zunächst ein Objekt der Klasse `TemporalVertexDegree` erzeugt. Dieses Objekt stellt einen Operator dar, der auf die Graph-Instanz angewendet wird, um die Gradmetrikevolution über Gradoop zu berechnen (FA4, NFA7). Die Ergebnisse der Berechnung werden in einer weiteren Methode zusammengetragen und im Nachgang in ein JSON-Objekt überführt. Dadurch stehen nun alle Daten, die für die Visualisierung benötigt werden, im richtigen Format bereit.

Neben der Metrikberechnung gibt es die Methoden `getGraphs` und `getVertices`. Erstere liefert die Namen der im Dateisystem vorhandenen Graph-Datensätze und `getVertices` gibt alle Knoten und die dazugehörigen Labels eines Graphen zurück. Deren Verwendung wird später in Abschnitt 5.2.5 näher erläutert.

Die genannten Methoden werden durch mehrere Hilfsfunktionen unterstützt, die jeweils notwendige Teilschritte für die Verarbeitung der Daten durchführen. Generell wurde bei der Implementierung darauf geachtet, Komponenten auszulagern. Dadurch werden Ressourcen geteilt und Funktionalitäten können wiederverwendet werden (NFA1, NFA8).

### 5.2.3. Schnittstelle Backend

Um die errechneten Daten für das Frontend zugänglich zu machen, wurden die erstellten Java-Methoden in Jersey-Ressourcen umgewandelt. Somit konnte eine RESTful API geschaffen werden, die HTTP Requests entgegennimmt und entsprechende Daten zurückliefert. Anhand der folgenden Abbildung 5.4 wird gezeigt, wie dies umgesetzt wurde.

```
1  @POST
2  @Path("/metric")
3  @Produces("application/json;charset=utf-8")
4  public Response getMetric(MetricRequest request) throws Exception {
5      ...
6      return response;
7  }
```

Abb. 5.4.: Annotation der `getMetric` Java-Methode als Jersey-Ressource

Die Annotationen an der `getMetric` Methode geben an, wie Jersey die Klasse verwalten soll. Über `@POST` wird mitgeteilt, dass hier ein POST HTTP Request erwartet wird. Denn für die Ausführung der Methode werden Daten in Form eines `MetricRequest`-Objektes benötigt. Die Annotation `@Path` definiert den genauen Pfad, über den die Ressource erreicht werden kann. Dabei ist zu beachten, dass es sich um einen relativen Pfad handelt. Da sich `getMetric` jedoch in der Klasse `RequestHandler` befindet, gibt es neben dem Ursprungspfad keine weiteren übergeordneten Verzeichnisse. Der absolute Pfad ist durch `http://localhost:2342/metric/` gegeben, wobei `http://localhost` die HTTP-Web-Instanz des Grizzly-Servers darstellt und `:2342` den Port definiert, über den Verbindungen mit dem Netzwerk hergestellt werden. Das Unterverzeichnis `/metric/` gibt schlussendlich die Jersey-Ressource an, an die Daten gesendet werden können. Über die `@Produces`-Annotation wird festgelegt, dass das `Response`-Objekt, also die zu sendende Serverantwort `response`, im JSON-Format vorliegt und den Zeichensatz UTF-8 verwendet.

Um die Daten des Requests in einer Java-Klasse abzubilden, wurde eine POJO-Klasse namens `MetricRequest` erstellt. Diese Klasse enthält die Attribute, die für die Konfiguration benötigt werden. Sollte ein Request ein Objekt enthalten, dass die gleichen Attribute enthält, kann Jersey dieses Objekt der Klasse `MetricRequest` zuordnen. Dadurch wird das Einlesen der Requests erleichtert und zugleich genau definiert, welche Daten vom Client erwartet werden.

### 5.2.4. Schnittstelle Frontend

Damit der Client mit dem Server kommunizieren kann, wurden über jQuery geeignete Requests formuliert. Der Abbildung 5.5 ist zu entnehmen, wie der Request für die Knotenmenge eines bestimmten Graphen implementiert wurde. Innerhalb der JavaScript-Methode `loadVertices` wird ein AJAX-Aufruf mittels jQuery gestartet. Dabei müssen verschiedene Parameter gesetzt werden, um eine korrekte Kommunikation mit dem Server zu gewährleisten. Über die `url` wird der absolute Pfad zur Jersey-Ressource angegeben. Hierbei wird festgelegt, dass die Java-Methode mit dem Pfad `/vertices/` und dem Parameter `graphName` aufgerufen werden soll. Über `dataType` wird angegeben, dass die Daten des Response im JSON-Format erwartet werden. Mit dem `type` Parameter wird zusätzlich die HTTP-Methode definiert. Es handelt sich hierbei um HTTP GET, weshalb neben Metainformationen keine weiteren Daten an den Server gesendet werden. Des Weiteren wird über `success` festgelegt, was nach einem erfolgreichen Response geschieht und über `error` wird der Fehlerfall behandelt. Da es sich um eine asynchrone Kommunikation handelt, werden diese beiden Funktionen erst ausgeführt, sobald ein Response vorliegt.

```
1  function loadVertices(graphName) {
2      ...
3      $.ajax({
4          url: "http://localhost:2342/vertices/" + graphName,
5          dataType: "json",
6          type: "get",
7
8          success: function(response) {
9          ...
10         }
11         error: {
12         ...
13         }
14     });
15 }
```

Abb. 5.5.: JQuery Request für die Knotenmenge eines Graphen

### 5.2.5. Client-Logik

Bevor eine Metrik visualisiert werden kann, muss eine Graph-Konfiguration gewählt werden. Um dem Benutzer die Eingabe zu erleichtern (NFA3), werden bereits vor der Benutzereingabe zwei asynchrone Requests an den Server gesendet, die jeweils Informationen über die vorhandenen Daten liefern. Zuerst werden über die Jersey-Ressource `getGraphs` die Graph-Datensätze erfragt, die beim Server im Dateisystem vorliegen. Nachdem ein existierender Datensatz gewählt wurde, werden die in diesem Graphen vorhandenen Knoten samt Label in einem Dropdown zur Wahl gestellt. Dies ermöglicht einen asynchronen Request an die `getvertices` Ressource und das anschließende Laden der erhaltenen Daten in eine VirtualSelect-Instanz. Der Benutzer kann nun eine Konfiguration wählen. Anschließend wird mittels JavaScript die Benutzereingabe eingelesen und zu einem Objekt

zusammengefasst. Dieses Objekt wird mit jQuery an die `getMetric` Webserver-Methode geliefert, wo die Berechnung der Metrik beginnt.

Während auf den Response vom Server gewartet wird, wird durch JavaScript das DOM verändert, indem in der Seitenleiste neue HTML-Elemente hinzugefügt werden. Diese Elemente enthalten Informationen über die geforderte Metrik und erlauben über einen Bezeichner die Zuordnung von Graphen zu Diagrammen. Die Diagramme werden als ECharts-Instanz erzeugt und an ein weiteres HTML-Element gebunden.

```
1  "data":[
2  {"vertex":"5ea8252a9e45472ee6b","from":1532801049746,"to":1532801683565,"value":4},
3  {"vertex":"5ea8252a9e45472ee6b","from":1532801683565,"to":1532801726065,"value":3},
4  {"vertex":"5ea8252a9e45472ee6b","from":1532801726065,"to":1532801971953,"value":2},
5  ...
6  ]
```

Abb. 5.6.: Ausschnitt einer Server-Response auf die Berechnung einer Metrik

Nachdem die Berechnung abgeschlossen ist, wird der Response mit den enthaltenen Daten von jQuery abgefangen und in JavaScript ausgelesen. Abbildung 5.6 zeigt, wie Metriken innerhalb des `data` Attributs eines Server Response strukturiert sind. Im gezeigten Ausschnitts ist zu erkennen, wie sich der Metrikwert des Knotens `5ea8252a9e45472ee6b` im Zeitintervall `1532801049746` bis `1532801971953` von `4` auf `2` senkt. Die Zeitpunkte sind dabei in Unixzeit mit der Genauigkeit auf Millisekunden angegeben. Das `data` Attribut in dem Response ist neben Metainformation jenes Attribut, das die Ergebnisse der Metrikberechnung speichert. Dabei besteht die Ergebnismenge aus einzelnen Metrikobjekten, die einem Knoten `vertex` einen Metrikwert `value` in einem bestimmten Zeitintervall `from` bis `to` zuordnen. Diese Objekte werden im Browser Cache zwischengespeichert. Dadurch wird die Performanz verbessert (NFA2), indem die Metriken bei Bedarf nicht neu geladen werden müssen. Sollte eine neue Metrik beim Server erfragt werden, dann wird zuvor geschaut, ob die Ergebnisse nicht bereits im Cache vorhanden sind. Wenn die benötigte Metrik für die Visualisierung bereits vorliegt, dann referenziert das neue Diagramm auf die bereits vorhanden Daten. Durch die Vermeidung von redundanten Kopien können zusätzlich Hardwareressourcen gespart werden (NFA8).

Da es sich bei den einzelnen Metrikobjekten jedoch um Zeitintervalle handelt und für ECharts konkrete Datenpunkte benötigt werden, müssen die Daten für die Visualisierung vorverarbeitet werden. Dafür wird eine Methode verwendet, die virtuelle Datenpunkte auf Grundlage der tatsächlichen Werte erzeugt. Hierbei wird in kontinuierlichen Zeitschritten iteriert und innerhalb der Ergebnismenge geprüft, welche Metrikobjekte mit ihren Zeitintervallen den aktuellen Zeitpunkt beinhalten. Bei allen validen Objekten wird ein neuer Datenpunkt erstellt, der aus dem dazugehörigen Knoten, dem Metrikwert und dem momentanen Zeitpunkt besteht. Damit diese Berechnung möglichst effizient läuft, wird nur innerhalb eines Zeitintervalls iteriert, das durch die erste und letzte Metrikänderung beschränkt ist (NFA2). Da alle Zeitpunkte außerhalb dieses Intervalls keine Änderung aufweisen, sind sie für die Analyse weniger relevant und deren Vernachlässigung spart Speicherplatz (NFA8). Die Granularität dieser virtuellen Datenpunkte wird durch den Zeitabstand zwischen den Iterationen bestimmt. Der Zeitabstand wird in Abhängigkeit von dem vorliegenden

Zeitintervall prozentual bestimmt und sorgt somit mit hoher Wahrscheinlichkeit für einen angemessenen Detailgrad.

Die virtuellen Daten werden nun in eine zuvor konfigurierte ECharts-Instanz geladen und im entsprechenden Diagramm visualisiert. Als Diagramm-Konfiguration wurde unter anderem der Moving Average implementiert, mit dem sich Schwankungen in den Daten glätten lassen. Dafür wird kontinuierlich ein Fenster über die im Diagramm abgebildeten Datenpunkte geschoben und jeweils ein Mittelwert gebildet. Das Zeitfenster definiert dabei die Anzahl an Datenpunkten, die aggregiert werden sollen. Diese neu entstandenen Datenpunkte, bestehend aus gemittelten Metrikwerten und dem Mittelwert der Zeitpunkte, werden nun als neuer Datensatz für die ECharts-Instanz gesetzt.

Um auf Interaktionen mit der Benutzeroberfläche zu reagieren, wurden über JavaScript Event Handler an HTML-Elemente gekoppelt. Als Beispiel sei hier der Event Handler auf dem Button mit Mülleimer Icon genannt. Durch den Event Handler wird ein Klicken erkannt und anschließend eine Methode aufgerufen, die für das Löschen der Diagramme zuständig ist. Dabei wird die zum Graphen dazugehörige ECharts-Instanz gelöscht und es werden alle Diagrammreferenzen auf die Daten entfernt.

### 5.2.6. Benutzeroberfläche

Die grundlegende Struktur der graphischen Benutzeroberfläche wurde im HTML-Dokument in eine Navigationsleiste, Seitenleiste und Darstellungsfläche unterteilt. Des Weiteren wurden hier der Aufbau eines Graph und Diagramm-Konfigurationsfensters sowie die Anordnung der visuellen und interaktiven Elemente innerhalb der Benutzeroberfläche festgelegt. Über HTML wurden auch alle Abhängigkeiten, die der Client für die Darstellung oder Verarbeitung der Webseite benötigt, referenziert. Darunter fallen neben den JavaScript- und CSS-Dateien auch die Frontend-Bibliotheken wie jQuery oder Frameworks wie Bootstrap.

```
1  <div class="row form-group">
2      <label for="graphSelector" class="col-6 form-label">Choose a graph</label>
3      <div class="col-6">
4          <select class="form-select selector" id="graphSelector"></select>
5      </div>
6  </div>
```

Abb. 5.7.: Bootstrap-Komponente aus dem Graph-Konfigurationsfenster

In Abbildung 5.7 wird anhand eines Beispiels vermittelt, wie Bootstrap, JavaScript und CSS mit den Elementen des HTML-Dokuments agieren. Durch die Verwendung von einheitlichen Komponenten konnte gewährleistet werden, dass die Webseite auch in unterschiedlichen Browsern konsistent erscheint (NFA9). Dazu werden HTML-Elemente über Klassennamen vorgefertigten Bootstrap-Komponenten zugeteilt. Das `select` HTML-Element wird in der Abbildung durch die Klasse `form-select` über Bootstrap neu definiert. Auch die Verwendung des Grid-Systems ist in der Abbildung zu sehen. In diesem Beispiel bildet ein `div` Element eine neue Reihe `row`, die sich in zwei Spalten unterteilt. Die Spalten bilden dabei die Elemente `label` und `div`, denen jeweils die Klasse `col-6`

zugeordnet wurde. Da Reihen im Bootstrap-Grid-System aus zwölf Einheiten bestehen, wird den Spalten jeweils genau die Hälfte der Breite zugeteilt. Durch das `id` Attribut des `select` Elements mit dem Wert `graphSelector` kann das Element mittels JavaScript leicht im DOM adressiert werden. In diesem Fall handelt es sich um das Dropdown-Menü, mit dem der Graphen Datensatz für die Konfiguration gewählt werden kann. Über JavaScript kann somit unkompliziert das Dropdown mit Optionen gefüllt und anschließend der gewählte Wert ausgelesen werden. Durch CSS wurden optische Anpassungen vorgenommen, um das Erscheinungsbild der Benutzeroberfläche nach eigenen Vorstellungen zu entwerfen (NFA9). Über verschiedenste Selektoren ließen sich mit CSS einzelne HTML-Elemente oder auch ganze Gruppen individuell gestalten. Über den Klassenselektor wurden beispielsweise alle Dropdowns einheitlich verändert. Deshalb trägt das `select` Element aus der Abbildung auch die Klasse `selector`. Das Einklappen der Seitenleiste oder die dynamische Anpassung mehrerer Diagramme an die Darstellungsfläche seien hier als weitere Beispiele für die Verwendung von CSS genannt.

# 6. Evaluierung des Temporal Metric Explorer

In diesem Kapitel wird die Funktionsweise des entworfenen Temporal Metric Explorers untersucht. Dabei werden zunächst die Funktionalitäten unter Berücksichtigung der in Abschnitt 4.1 festgelegten Anforderungen getestet. Anschließend wird anhand verschiedener Datensätze geprüft, ob der TME als Analyseanwendung seinen Zweck erfüllen kann.

## 6.1. Funktionalitäten

Die Verwendung des Temporal Graph Explorer in Kombination mit dem Temporal Metric Explorer schafft eine ideale Grundlage für die Analyse temporaler Graphen. Die Tatsache, dass sich deren Funktionalitäten ergänzen, lässt sich schnell erkennen, denn bereits auf der Startseite beider Explorer lassen sich Hyperlinks zur jeweils anderen Anwendung finden (FA1). So ist dem Nutzer die Möglichkeit gegeben, ohne Umwege die Anwendung zu wechseln. Nach einer Graphenvisualisierung im TGE kann der Nutzer einen für ihn interessanten Knoten wählen und erhält Metainformationen zu diesem. Zudem hat er nun direkt die Option, sich die Graph Metrik des Knotens im TME anzeigen zu lassen (FA1). Um die Konfiguration muss sich der Nutzer hierbei nicht kümmern, denn diese wird automatisch entsprechend der TGE-Datengrundlage angepasst. Über den Hyperlink in der Kopfzeile gelangt der Nutzer auch ohne direkte Berechnung eines Knotens zur Startseite des TME. In der Seitenleiste des TME befindet sich ein *add Graph* Button durch dessen Klicken die Graph Einstellungen geöffnet werden (FA2).

Hier kann der Benutzer mehrere Einstellungen für eine gezielte Analyse vornehmen (NFA5). Zu Beginn muss er einen Graph wählen (FA3). Dazu werden ihm bereits alle temporalen Graph-Datensätze vorgeschlagen, die sich auf seinem lokalen Dateisystem befinden. Anschließend hat er die Wahl zwischen mehreren Metriken. Bereits implementiert ist der einfache Knotengrad sowie der Eingangs- und Ausgangsknotengrad (FA3). Zudem kann der Nutzer ein zu betrachtendes Zeitfenster konfigurieren (FA3). Dabei wird zwischen *all*, $asOf$, $fromTo$ und *between* unterschieden. Bei *all* wird die Metrik für den ganzen Graph-Datensatz berechnet, während bei $asOf$ nur die Daten zu einem gewählten Zeitpunkt $t_1$ in Betracht gezogen werden. $FromTo$ definiert ein abgeschlossenes Intervall mit $t_1$ und einem zweiten Zeitpunkt $t_2$, wohingegen *between* ein offenes Intervall zwischen $t_1$ und $t_2$ festlegt. Sollte für das gewählte Zeitfenster die Angabe von $t_1$ oder $t_2$ benötigt werden, wird der Benutzer in den darunterliegenden Feldern aufgefordert, die Daten anzugeben. Andernfalls sind diese Felder ausgegraut, um Uneindeutigkeiten zu vermeiden (NFA3, NFA6). Wenn der Nutzer bereits weiß, welchen Bereich des Graphen er analysieren möchte, kann er im letzten Feld nach Knoten filtern (FA3). Dafür werden ihm alle verfügbaren Knoten samt deren Label angezeigt. Dem Benutzer steht im Dropdown auch eine Suchfunktion zur Verfügung (NFA3). Bei der Berechnung der Metrik werden dann lediglich die ausgewählten Knoten berücksichtigt, wodurch die Wartezeit erheblich verkürzt wird (NFA2). Durch Klicken auf *Execute* beginnt die Berechnung der Metrik.

Die Seitenleiste gibt nun einen Überblick über die Metrik sowie den Datensatz der verwendet wurde (FA2). Durch eine Ladeanimation im Diagramm wird der Benutzer darüber in Kenntnis gesetzt, dass noch kein Ergebnis vom Server vorliegt. Ebenfalls ist der Button *add Graph* ausgegraut, wodurch darauf hingewiesen wird, dass während einer Berechnung keine neuen Graph Metriken angefordert werden können (NFA6). Sobald die Berechnung abgeschlossen ist, wird die entsprechende Metrik im Diagramm visualisiert. Es ist jedoch auch möglich, dass im Datensatz keine Daten zur Auswertung der gewünschten Metrik vorliegen. In diesem Fall wird der Benutzer über ein Pop-up-Fenster darüber informiert (NFA3). Im Diagramm werden initial bevorzugt jene Knoten dargestellt, die vorher manuell gefiltert wurden. Wenn kein Filter angegeben war, wird die Metrik eines zufälligen Knoten präsentiert.

Das dargestellte Stufendiagramm visualisiert die zuvor konfigurierte zeitliche Graph Metrik eines Knotens (FA5), wobei die Abszisse den Zeitverlauf abbildet und die Ordinate den entsprechenden Wert der Metrik. Am rechten Rand des Diagramms finden sich die skalaren Werte des *Minimum-*, *Maximum-* und *Durchschnittsknotengrad* (FA5). Diese Werte erlauben einen globalen Vergleich mit anderen Knoten und es lässt sich so die Bedeutsamkeit des aktuellen Knotens in Bezug auf den ganzen Graphen einordnen. Dem Benutzer werden neben den Metriken noch weitere Möglichkeiten zur Analyse derselben geboten (NFA5). So kann er unter anderem mit dem Mauszeiger über auffällige Abweichungen im Metrikverlauf schweben. Dabei rastet ein vertikaler Achsenzeiger an jenen Zeitpunkten ein, an denen es eine Wertänderung der Metrik gab. Ein danebenstehendes Informationsfeld gibt Aufschluss über den markierten Knoten, sowie dessen Wert der Metrik zum gewählten Zeitpunkt (FA5). Die Kombination des Achseneinrastens mit dem Informationsfeld erlaubt eine zeitgenaue Analyse der Änderungen im Verlauf. Ein anpassbares Zeitfenster unterhalb des Diagramms definiert den zu betrachtenden Zeitraum der Abbildung. Es kann verkleinert, vergrößert und verschoben werden, indem auf dem Zeitfenster gezogen oder alternativ innerhalb des Diagramms gescrollt wird (FA6). Somit lässt sich die Abbildung auf einen gewünschten Ausschnitt anpassen, um interessante Verläufe zu vergrößern. Des Weiteren ist der Nutzer in der Lage, über die drei Balken oben links im Fenster die Seitenleiste einzuklappen (FA5). Damit wird die Größe des Diagramms erweitert und es bleibt mehr Platz für die Visualisierung.

Weitere nützliche Funktionalitäten finden sich in der rechten oberen Ecke des Diagramms. Hier gibt es die Optionen *Data View*, *Save as Image*, *Chart Configuration* und *Moving Average*. Der *Data View* bietet eine Liste in Textform, in der alle Daten enthalten sind, die auch für die Visualisierung genutzt werden (FA6). Somit kann der Nutzer unproblematisch die Knoten-ID und deren Metrik, definiert durch Zeitintervalle von Werten, kopieren. Durch die *Save as Image-* Funktion kann die aktuelle Abbildung als PNG heruntergeladen werden (FA6). Mit der *Chart Configuration* bekommt der Nutzer weitere Einstellungsmöglichkeiten. Durch Klicken öffnet sich ein Fenster, in dem ausgewählt werden kann, welcher Knoten visualisiert werden soll (FA6). Dabei ist zu beachten, dass hier nur Knoten aufgelistet werden, die bereits Teil des ursprünglichen Request waren und für die somit Werte einer Metrik vorliegen (NFA3). Des Weiteren besteht hier die Option, noch einen weiteren Knoten auszuwählen, wodurch sich zwei Knoten innerhalb einer Metrik zeitgleich analysieren lassen (FA5). Dies ist vor allem dann hilfreich, wenn Vergleiche zwischen zwei Knoten gezogen werden sollen. Die zwei Knoten werden in unterschiedlichen Farben abgebildet, wodurch sie sich einfach voneinander unterscheiden lassen. Am oberen Rand des Diagramms finden sich die zu

den Knoten gehörigen Labels. Diese Labels tragen jeweils ein kleines Icon. Die Icons sind entsprechend der Knotenfarben im Diagramm gefärbt, um sie den Knoten eindeutig zuordnen zu können (NFA3). Sollte der Benutzer auf ein Label klicken, wird der zugehörige Knoten temporär aus der Visualisierung entfernt (FA6). Dadurch lässt sich schnell zwischen den Knoten wechseln, ohne eine erneute Diagramm-Konfiguration vornehmen zu müssen (NFA2, NFA8). Wenn die dargestellten Daten große Schwankungen aufweisen, ist es schwierig, genaue Entwicklungen zu erkennen. Dabei schafft der *Moving Average* Abhilfe, indem er den Diagrammverlauf glättet (FA6). Durch Klicken auf das Filter-Icon öffnet sich dem Nutzer das Einstellungsfenster. Hier kann er die Fenstergröße des Moving Average anhand eines Schiebereglers konfigurieren und durch *Execute* die Glättung ausführen.

Es kann sein, dass mehrere Metriken miteinander in Korrelation stehen. Dazu unterstützt der Temporal Metric Explorer die zeitgleiche Visualisierung von bis zu vier Metriken (FA5). So kann beispielsweise der Ausgangsknotengrad mit dem Eingangsknotengrad verglichen und analysiert werden, ob Abhängigkeiten untereinander bestehen. Um mehrere Metriken darzustellen, muss der Benutzer lediglich auf *add Graph* in der Seitenleiste klicken und erhält anschließend die Graph-Konfiguration für die weitere Metrik. Bereits berechnete Metriken werden zwischengespeichert, weshalb erneute Requests an den Server nur notwendig sind, wenn noch keine Daten für die Metrik vorliegen (NFA2). Dies verhindert Wartezeiten auf bereits erhaltene Berechnungen. Eine automatische Anpassung der Diagrammgrößen bei mehreren Visualisierungen sorgt für eine optimale Nutzung der verfügbaren Bildschirmfläche (NFA9). Die Seitenleisteninformationen tragen als Überschrift Graphennamen, wodurch sich Metriken und Datensätze deren jeweiligen Diagramm zuordnen lassen (FA2). Das Löschen von Diagrammen ist ebenfalls problemlos über die Seitenleiste möglich. Hierzu muss der Benutzer auf das Papierkorb-Icon neben den Graphennamen klicken und das zugehörige Diagramm wird entfernt (FA2).

Der Funktionsumfang des TME bietet eine gute Grundlage für eine interaktive Exploration und zeitliche Analyse von Graph Metriken. Durch die Auflistung der Funktionen konnte gezeigt werden, dass der TME die in Abschnitt 4.1 gestellten Anforderungen erfüllt.

## 6.2. Chess Stack Exchange

Chess Stack Exchange[9] ist das Schach-Äquivalent zu Stack Overflow. In dem Online-Forum können die Benutzer ihr Wissen rund um das Thema Schach austauschen. Dabei lassen sich Diskussionen mit anderen Nutzern führen, in denen unter anderem Fragen, Kommentare und Antworten verfasst werden können. Der für dieses Beispiel verwendete Datensatz umfasst die einzelnen User und deren Beiträge sowie Relationen.

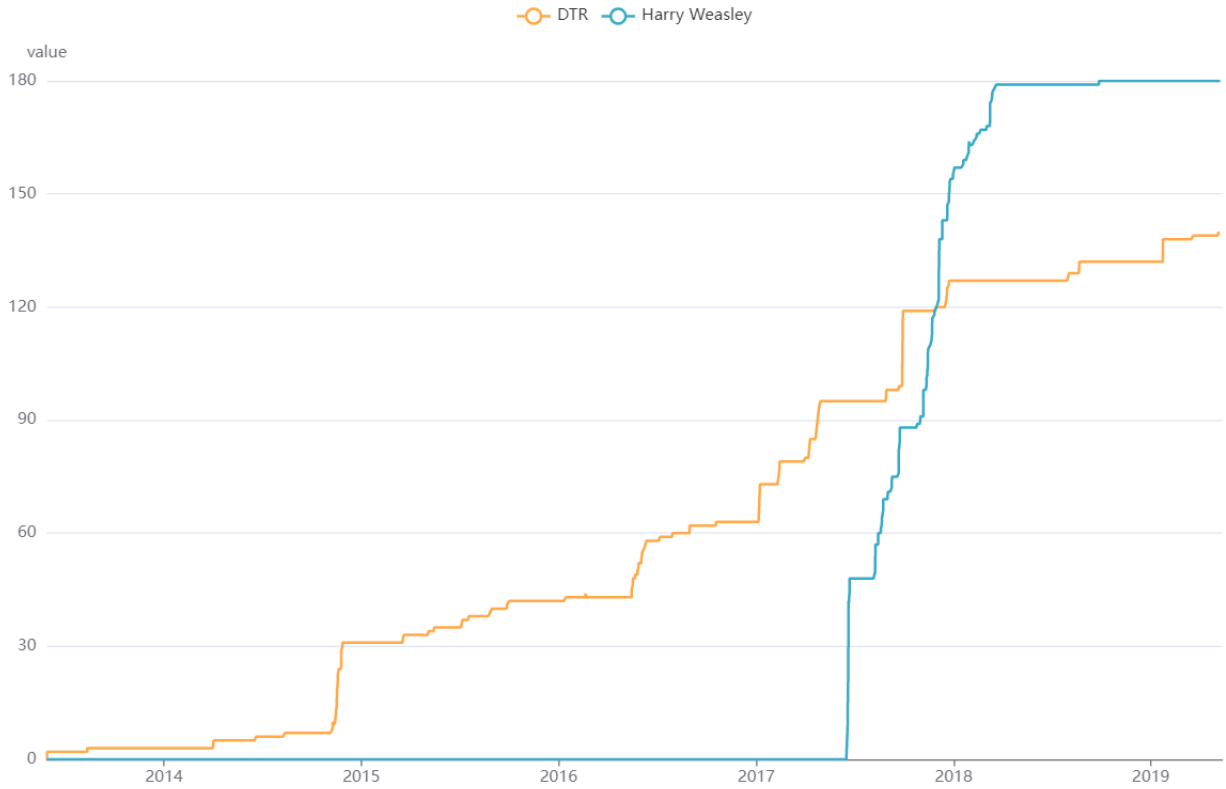



Abb. 6.1.: Knotengradevolution von zwei Usern im Chess-Stack-Exchange-Datensatz

In Abbildung 6.1 sind die Knotengrade zweier Benutzer als Zeitserien visualisiert. Zu sehen ist deren Interaktion mit dem Forum im Zeitraum von 2014 bis 2019. Der User *DTR* ist bereits seit Anfang 2014 Mitglied im Forum. Bei ihm ist durch die kleinen Anstiege in der Evolution zu erkennen, dass er meist ziemlich wenig, dafür aber kontinuierlich am Forum teilnimmt. Da er sich online scheinbar oft mit Schach beschäftigt, kann vermutet werden, dass er auch im echten Leben regelmäßig Schach spielt. Anders ist dies bei dem User *Harry Weasley*, denn seine Entwicklung verläuft zwar rasant, aber dafür nur kurz. Ab Mitte 2017 war *Harry Weasley* so aktiv im Forum, dass er bereits nach etwa vier Monaten mehr mit der Plattform interagierte als der jahrelange User *DTR*. Doch da diese Teilnahme nur etwa ein halbes Jahr hielt, ließe sich möglicherweise schlussfolgern, dass Schach spielen für ihn nur eine vorübergehende Beschäftigung war.

[9] https://chess.stackexchange.com (abgerufen am 12.03.2023)

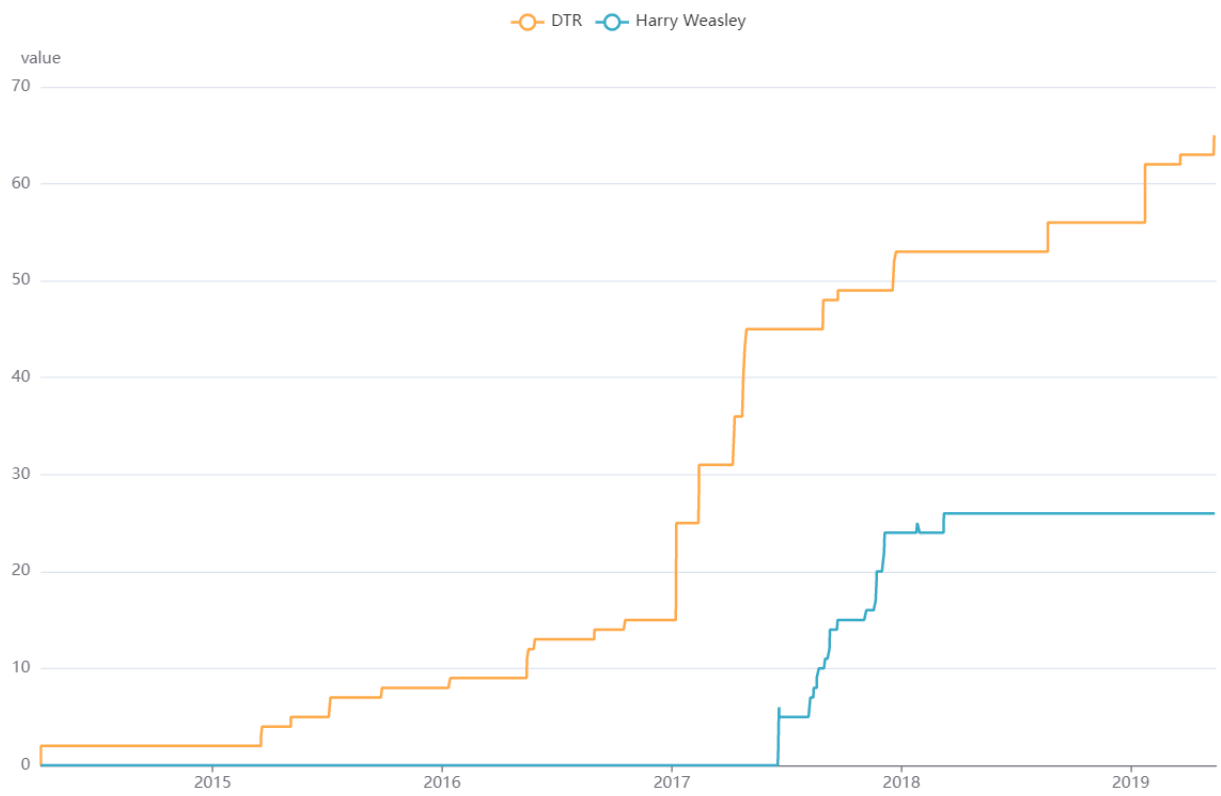



Abb. 6.2.: Eingangsknotengradevolution von zwei Usern im Chess-Stack-Exchange-Datensatz

Obwohl der Knotengrad des Users *Harry Weasley* jenen von *DTR* schnell überstieg, konnte festgestellt werden, dass der Eingangsknotengrad wenig dazu beitrug. Abbildung 6.2 zeigt die Eingangsknotengrade von den beiden Usern im Zeitraum von 2014 bis 2019. Es lässt sich erkennen, dass die Werte der Zeitserie von *Harry Weasley* stets unter denen von *DTR* lagen. Bei einem Vergleich der Maximalwerte der Metriken von *Harry Weasley*, lässt sich erkennen, dass der Eingangsknotengrad mit 26 nur einen kleinen Teil von den insgesamt 180 ausmacht. Es fanden somit im Vergleich nur wenige Interaktionen auf Beiträge von *DTR* statt, wohingegen er viel mit anderen interagierte. Dies könnte bedeuten, dass er entweder wenig Fragen gestellt und somit Diskussionen eröffnet hat oder die gestellten Fragen leicht zu beantworten waren, wodurch keine weiteren Diskussionen ausgelöst wurden. Bei *Harry Weasley* scheint es ausgeglichener zu sein. Denn einem Vergleich seiner Eingangsknotengradevolution mit der des Knotengrades ist zu erkennen, dass der Eingangsknotengrad in etwa die Hälfte des Gesamtwertes ausmacht. Daher lässt sich annehmen, dass dieser User etwa gleich viele Fragen und Antworten verfasst hat und somit ebenso oft an fremden Diskussionen teilnimmt wie andere an seinen.

## 6.3. Citi Bike

Citi Bike[10] ist ein Anbieter für den Fahrradverleih in New York City, der seine generierten Daten für die Öffentlichkeit zugänglich macht. Der für dieses Beispiel verwendete Datensatz bildet ein Netzwerk aus Verleih-Stationen, die über Fahrtwege miteinander verbunden sind. Ein Fahrtweg beschreibt dabei eine zeitabhängige Ausleihe von einer Station zur nächsten.

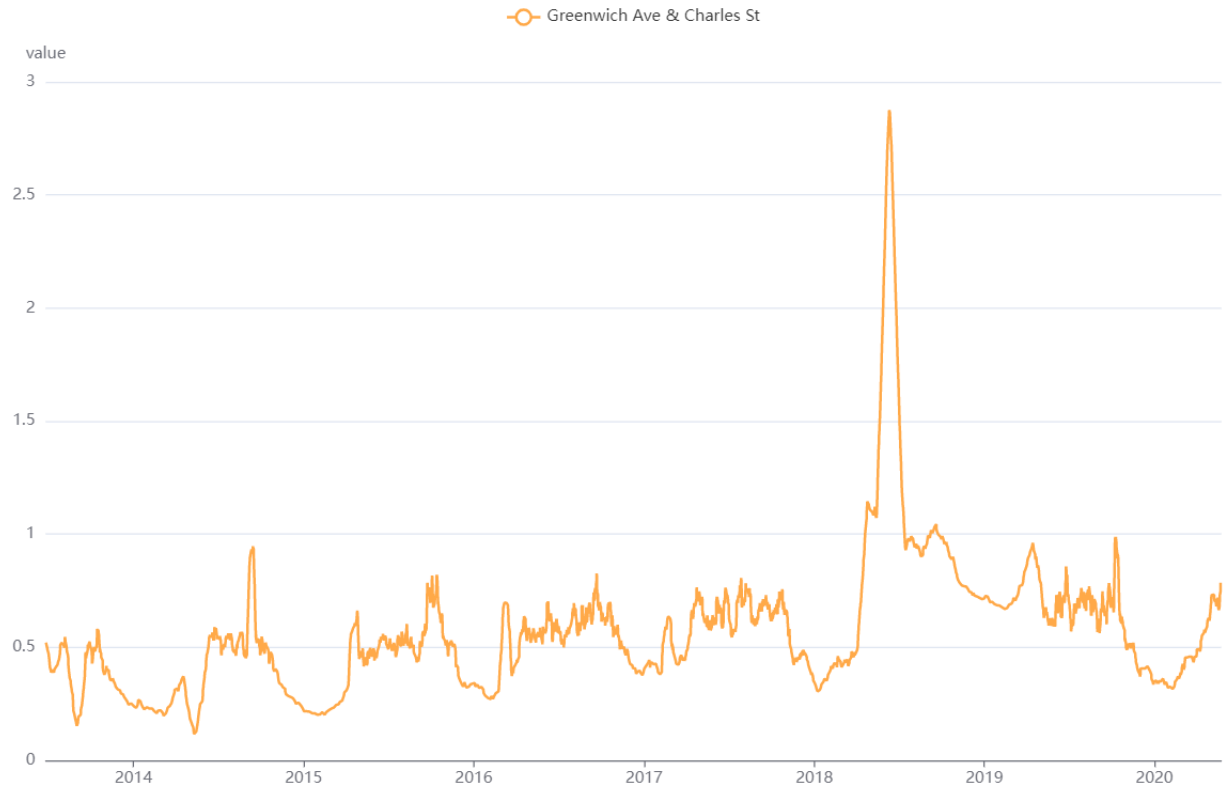



Abb. 6.3.: Knotengradevolution von einer Station des Citi-Bike-Datensatzes

Abbildung 6.3 zeigt die Knotengradevolution der Station *Greenwich Ave & Charles St* im Zeitraum von Anfang 2014 bis Ende 2020. Zu sehen ist, wie häufig Fahrten diese Station als Ziel oder Ausgangspunkt haben. Im gezeigten Diagramm wird der Moving Average verwendet. Durch diese Aggregierung der Daten verringert sich zwar die Genauigkeit der Werte, doch ohne diesen würde die Entwicklung des Knotengrades zu sprunghaft wirken. Dies liegt an der kurzen Dauer von Fahrten und der hohen Frequenz an Ausleihen. In der Evolution lassen sich jeweils zu den Jahreswechseln Einbrüche erkennen. Hier ließe sich darauf schließen, dass aufgrund der Kälte im Winter und Frühling weniger Personen mit dem Fahrrad unterwegs sind. Im Gegensatz dazu wird das Angebot der Ausleihe von Fahrrädern in wärmeren Monaten aktiver genutzt. Möglicherweise befinden sich zu diesen Jahreszeiten auch mehr Touristen in New York City, wodurch mehr Personen auf ein Fahrrad angewiesen sind. Neben dem periodischen Muster ist Mitte 2018 eine starke Abweichung zu erkennen. Dieser Ausschlag lässt vermuten, dass es in dieser Zeit ein Ereignis gab, das mehr Menschen als üblich dazu bewegte, diese Station zu nutzten.

[10] https://citibikenyc.com/system-data (abgerufen am 12.03.2023)

Nachdem die Stationen im näheren Umfeld untersucht wurden, konnte eine Erklärung für den Anstieg an Ausleihen gefunden werden. Die Station *Greenwich Ave & 8 Ave* liegt auf derselben Straße und dessen Knotengradentwicklung verzeichnete zur gleichen Zeit einen starken Einbruch.

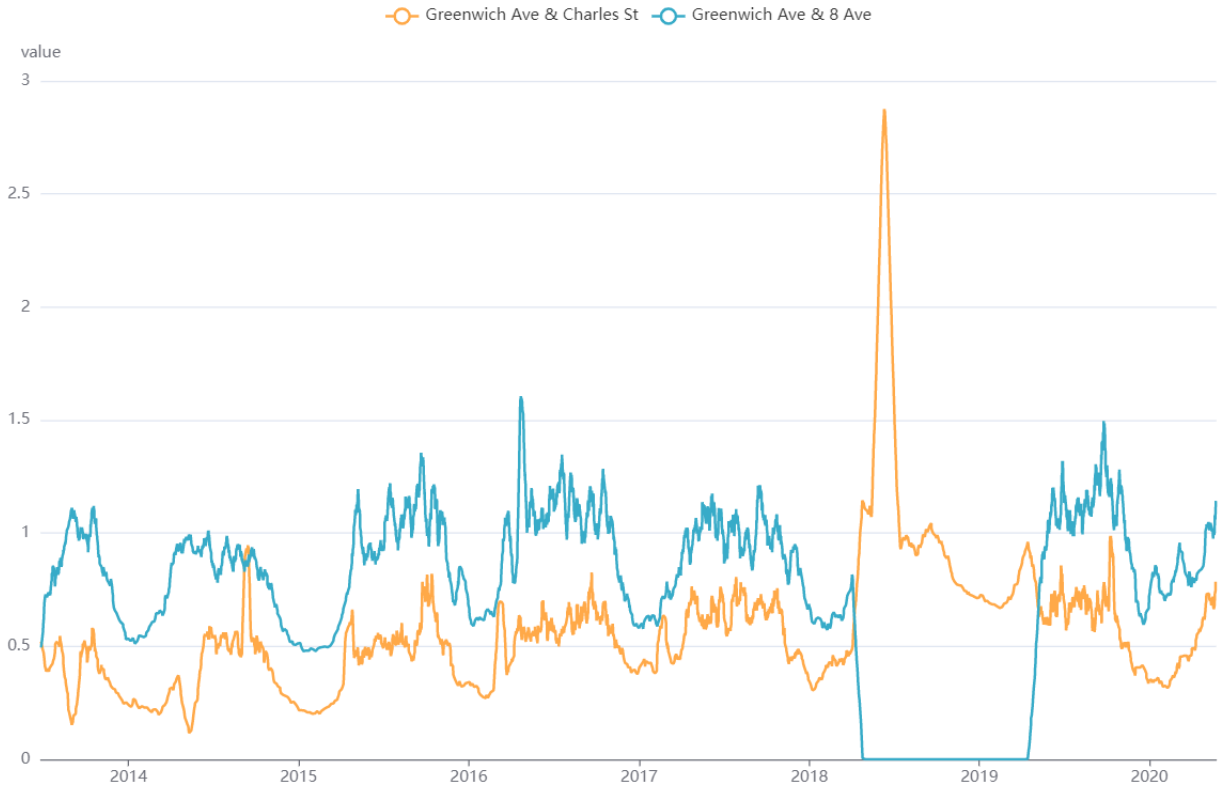



Abb. 6.4.: Knotengradevolution von zwei Stationen des Citi-Bike-Datensatzes

In Abbildung 6.4 wurden beide Stationen in einem Diagramm abgebildet, wodurch sich deren zeitliche Metrik besser vergleichen lässt. Dabei ist eindeutig zu erkennen, dass der Anstieg bei der Station *Greenwich Ave & Charles St* mit dem Einbruch korreliert. Da die Station *Greenwich Ave & 8 Ave* etwa ein Jahr lange gar keine Ausleihen oder Abgaben von Fahrrädern aufwies, ließe sich vermuten, dass beispielsweise Bauarbeiten eine zwischenzeitliche Schließung erforderten. Die Personen, die sonst diese Station nutzten, wichen somit auf die nächstgelegene Station *Greenwich Ave & Charles St* aus. Jedoch ist zu sehen, dass der Andrang an Personen nicht über das ganze Jahr anhielt. Eine mögliche Erklärung hierfür wäre, dass aufgrund des Bedarfs eine weitere Station als Ersatz aufgestellt wurde.

# 7. Zusammenfassung und Ausblick

Zu Beginn dieser Arbeit erfolgte eine Einführung in die Thematik. Dabei wurde gezeigt, dass die Visualisierung zeitlicher Graph Metriken neue Perspektiven in der Analyse von Netzwerken ermöglicht. Durch das Abbilden von zeitabhängigen Entwicklungen lassen sich neben Korrelationen und Trends auch Zeitpunkte identifizieren, die für aufbauende Untersuchungen interessant sein könnten. Nachdem ein Verständnis für grundlegende Begriffe in diesem Gebiet aufgebaut wurde, konnte anhand verwandter Arbeiten ein Einblick in die aktuelle Forschungslage vermittelt werden. Dabei ist zu erkennen, dass es zwar Ansätze und theoretische Modelle zur Analyse temporaler Graphen gibt, diese jedoch selten in der Praxis umgesetzt werden. Zudem ist die Visualisierung der Evolution von Graph Metriken weiterhin unterrepräsentiert. Die meisten Arbeiten in diesem Gebiet legen ihren Fokus eher auf die Darstellung des gesamten temporalen Graphen. Demnach herrscht momentan Bedarf an praktischen Anwendungen, die die Visualisierung zeitlicher Graph Metriken in den Mittelpunkt der Analyse rücken. Mit der Entwicklung des Temporal Metric Explorer wurde ein Schritt in diese Richtung unternommen. Um eine erfolgreiche Implementierung gewährleisten zu können, wurde zunächst eine Konzeption aufgestellt. In dieser wurden funktionale und nichtfunktionale Anforderungen an den Prototyp erhoben und eine Systemarchitektur entworfen. Damit das Konzept umgesetzt werden konnte, wurden geeignete Technologien gewählt und deren jeweiligen Vorteile erläutert. Anschließend wurden Teile der Implementierung vorgestellt und es konnte gezeigt werden, wie die einzelnen Technologien im TME miteinander interagieren. Nachdem die einzelnen Schritte zur Realisierung des Prototypen erklärt wurden, erfolgte eine Evaluierung der Anwendung. Dabei konnte gezeigt werden, dass der TME die Anforderungen erfüllt. Anhand der Auswertung zweier Beispieldatensätze wurde ein Einblick in die Möglichkeiten zur Analyse vermittelt.

Der Temporal Metric Explorer kann als Grundlage für weitere Arbeiten verwendet werden. Da bei der Implementierung darauf geachtet wurde, die einzelnen Komponenten erweiterbar zu gestalten, kann der TME unkompliziert um neue Funktionalitäten ergänzt werden. Allem voran können neben dem Knotengrad weitere zeitliche Graph Metriken hinzugefügt werden, sobald diese über Gradoop verfügbar sind. Der momentan implementierte Gradoop-Operator zur Berechnung des Minimum-, Maximum- und Durchschnittsknotengrades erfordert eine sequentielle Verarbeitung jedes Knotens und eine anschließende Aggregation. Daher stellt dies aktuell einen Engpass der Laufzeit dar. Eine parallele Verarbeitung der Knoten könnte insbesondere bei großen Datensätzen die Ausführungsgeschwindigkeit verbessern. Sollte es möglich sein, den Gradoop-Operator anzupassen, könnte auch der TME dahingehend optimiert werden. Auch wenn Gradoop zum jetzigen Zeitpunkt für die Batch-Verarbeitung von Daten ausgelegt ist, könnte in Zukunft eine Anpassung zur Stream-Verarbeitung erfolgen. Die Verwendung von Apache ECharts als Visualisierungsbibliothek unterstützt die Verarbeitung von Streaming-Daten, weshalb auch hier eine gute Grundlage für zukünftige Arbeiten vorliegt. Da der TME als Web-Anwendung entwickelt wurde, ist es denkbar, eine spätere Version zu hosten und über das Web zugänglich zu machen. Da hierbei parallele Zugriffe entstehen können, müsste eine Verwaltung von Client Sessions implementiert werden und eine Ergänzung um Sicherheitsmechaniken erfolgen.

Abschließend lässt sich sagen, dass mit dem Temporal Metric Explorer das Ziel der Entwicklung einer interaktiven Anwendung zur Exploration zeitlicher Graph Metriken erreicht wurde.

# Literatur


[1] Valentina Kuskova. *The Future Belongs to Network Analysis*. abgerufen am 15.03.2023. URL: https://www.hse.ru/en/news/admission/213168174.html.

[2] Statista. *Social Media - Anzahl der Nutzer weltweit bis 2022*. abgerufen am 01.03.2023. URL: https://de.statista.com/statistik/daten/studie/739881/umfrage/monatlich-aktive-social-media-nutzer-weltweit/.

[3] Reinhard Diestel. *Graph Theory*. 5th Edition. abgerufen am 15.03.2023. Springer Berlin, 2017. ISBN: 978-3-662-53622-3.

[4] Béla Bollobás. *Modern Graph Theory*. abgerufen am 13.02.2023. Springer New York, 1998. ISBN: 978-1-4612-0619-4.

[5] David Lisowski. *Modularity-basiertes Clustern von dynamischen Graphen im Offline-Fall*. abgerufen am 06.01.2023. 2011. URL: https://i11www.iti.kit.edu/_media/teaching/theses/da-lisowski.pdf.

[6] Christopher Rost u. a. „Evolution of Degree Metrics in Large Temporal Graphs“. In: *Datenbanksysteme für Business, Technologie und Web (BTW 2023), 20. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme"(DBIS), 06.-10, März 2023, Dresden, Germany, Proceedings*. Hrsg. von Birgitta König-Ries u. a. Bd. P-331. LNI. abgerufen am 15.03.2023. Gesellschaft für Informatik e.V., 2023. DOI: 10.18420/BTW2023-23.

[7] Christopher Rost u. a. „Distributed temporal graph analytics with GRADOOP“. In: *VLDB J.* 31 (2022). abgerufen am 15.03.2023. DOI: 10.1007/s00778-021-00667-4.

[8] J.L. Gross, J. Yellen und Zhang. *Handbook of Graph Theory*. 2nd Edition. abgerufen am 13.02.2023. Chapman und Hall/CRC, 2013. ISBN: 9780429192753. DOI: https://doi.org/10.1201/b16132.

[9] Linton C. Freeman. *Centrality in Social Networks Conceptual Clarification*. abgerufen am 06.01.2023. 1979. URL: https://www.bebr.ufl.edu/sites/default/files/Centrality%5C%20in%5C%20Social%5C%20Networks.pdf.

[10] Francis Bloch, Matthew O. Jackson und Pietro Tebaldi. *Centrality Measures in Networks*. abgerufen am 06.01.2023. 2016. URL: https://arxiv.org/abs/1608.05845.

[11] I. Herman, G. Melancon und M.S. Marshall. „Graph visualization and navigation in information visualization: A survey“. In: *IEEE Transactions on Visualization and Computer Graphics* 6.1 (2000). abgerufen am 13.02.2023. DOI: 10.1109/2945.841119.

[12] Stef van den Elzen u. a. „Dynamic Network Visualization withExtended Massive Sequence Views“. In: *IEEE Transactions on Visualization and Computer Graphics* 20.8 (2014). abgerufen am 13.02.2023. DOI: 10.1109/TVCG.2013.263.

[13] Lothar Krempel. „Network visualization“. In: *The SAGE handbook of social network analysis* (2011).

[14] Mathias Pohl, Florian Reitz und Peter Birke. „As Time Goes by: Integrated Visualization and Analysis of Dynamic Networks“. In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. abgerufen am 06.01.2023. Association for Computing Machinery, 2008. ISBN: 9781605581415. DOI: 10.1145/1385569.1385636.

[15] Christopher Rost u. a. „Exploration and Analysis of Temporal Property Graphs“. In: *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*. Hrsg. von Yannis Velegrakis u. a. abgerufen am 15.03.2023. OpenProceedings.org, 2021. DOI: 10.5441/002/edbt.2021.83.

[16] Christopher Rost u. a. „Evolution Analysis of Large Graphs with Gradoop“. In: *Machine Learning and Knowledge Discovery in Databases - International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019, Proceedings, Part I*. Hrsg. von Peggy Cellier und Kurt Driessens. Bd. 1167. Communications in Computer and Information Science. abgerufen am 15.03.2023. Springer, 2019. DOI: 10.1007/978-3-030-43823-4_33.

[17] Christopher Rost, Andreas Thor und Erhard Rahm. „Analyzing Temporal Graphs with Gradoop“. In: *Datenbank-Spektrum* 19 (2019). abgerufen am 15.03.2023. DOI: 10.1007/s13222-019-00325-8.

[18] Konstantin Shvachko u. a. „The Hadoop Distributed File System“. In: *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*. abgerufen am 13.02.2023. 2010. DOI: 10.1109/MSST.2010.5496972.

[19] Paris Carbone u. a. „Apache Flink™: Stream and Batch Processing in a Single Engine“. In: *IEEE Data Eng. Bull.* 38 (2015).

[20] Lawrence Chung u. a. *Non-functional requirements in software engineering*. Springer Science & Business Media, 2012.

[21] Lawrence Chung und Julio Cesar Sampaio do Prado Leite. „On Non-Functional Requirements in Software Engineering“. In: *Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos*. Hrsg. von Alexander T. Borgida u. a. abgerufen am 13.02.2023. Springer Berlin Heidelberg, 2009. ISBN: 978-3-642-02463-4. DOI: 10.1007/978-3-642-02463-4_19.

[22] Pawan Vora. *Web application design patterns*. Morgan Kaufmann, 2009.

[23] O Vogel I Arnold A Chughtai, E Ihler T Kehrer U Mehlig und U Zdun. „Software-Architektur“. In: (2009).

[24] Jörg Krause. „HTML: Hypertext Markup Language“. In: *Introducing Web Development*. abgerufen am 13.02.2023. Apress, 2016. ISBN: 978-1-4842-2499-1. DOI: 10.1007/978-1-4842-2499-1_3.

[25] Ian Hickson und David Hyatt. „Html5“. In: *W3C Working Draft WD-html5-20110525* (2011).

[26] Hans Van Vliet, Hans Van Vliet und JC Van Vliet. *Software engineering: principles and practice*. John Wiley & Sons Hoboken, NJ, 2008.

[27] *Bootstrap v5 Documentation*. abgerufen am 06.01.2023. URL: https://getbootstrap.com/docs/5.0.

[28] Allen Wirfs-Brock und Brendan Eich. „JavaScript: the first 20 years“. In: *Proceedings of the ACM on Programming Languages* 4 (2020).

[29] David Flanagan. *JavaScript: the definitive guide*. Bd. 1018. O'reilly, 2006.

[30] Bear Bibeault, Aurelio De Rosa und Yehuda Katz. *jQuery in Action*. Simon und Schuster, 2015.

[31] Deqing Lia u. a. „ECharts: A Declarative Framework for Rapid Construction of Web-based Visualization“. In: (2018).

[32] L.D. Paulson. „Building rich web applications with Ajax“. In: *Computer* 38 (2005). abgerufen am 13.02.2023. DOI: 10.1109/MC.2005.330.

[33] Felipe Pezoa u. a. „Foundations of JSON schema“. In: *Proceedings of the 25th international conference on World Wide Web*. 2016.

[34] Boci Lin u. a. „Comparison between JSON and XML in Applications Based on AJAX“. In: *2012 International Conference on Computer Science and Service System*. abgerufen am 13.02.2023. 2012. DOI: 10.1109/CSSS.2012.297.

[35] Alex Rodriguez. „Restful web services: The basics“. In: *IBM developerWorks* 33 (2008).

[36] Tim Berners-Lee, Roy Fielding und Larry Masinter. *Uniform resource identifier (URI): Generic syntax*. Techn. Ber. 2005.

[37] Ned Freed und Nathaniel Borenstein. *Multipurpose internet mail extensions (MIME) part two: Media types*. Techn. Ber. 1996.

[38] Jose Sandoval. *RESTful Java Web Services*. Bd. 1. Packt Publishing, 2009.

[39] Christian Ullenboom. *Java ist auch eine Insel*. Bd. 1475. Galileo Press, 2004.

[40] JetBrains. *IntelliJ IDEA – the Leading Java and Kotlin IDE*. abgerufen am 25.02.2023. URL: https://www.jetbrains.com/idea/features/.

# Selbstständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit mit dem Thema:

*„Visualisierung zeitlicher Graph Metriken“*

selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann. Ich versichere, dass das elektronische Exemplar mit den gedruckten Exemplaren übereinstimmt.

Leipzig, den 23.03.2023

_____

Julian Nico Pielmaier