Internet and Security
Social Graph Explorer - Backend

Vera Benz, Yasmin Meier, Esther Mugdan

3. Juli 2021

Inhaltsverzeichnis

Einleitung

Das Ziel des Projektes ist es, für das bestehende BACnet einen sozialen Graphen bereitzustellen, welcher eine visuelle Darstellung des Netzwerkes ermöglicht. Dadurch soll ein Überblick der Verbindungen bzw. Follow-Beziehungen zwischen den Personen im Netzwerk gegeben werden. An einen Social Graph Explorer können unterschiedliche Anforderungen gestellt werden. Wesentlich ist die Eigenschaft, dass die bestehenden Verbindungen von einem gegebenen Zentrum aus korrekt angezeigt werden.

Die Umsetzung des Projektes wurde in zwei Gruppen aufgeteilt: Front- und BackEnd. Dieser Report befasst sich mit den Aufgaben des BackEnds. Diese bestehen darin, die Datenstrukturen bereitzustellen und die notwendigen Voraussetzungen zu schaffen. Um die Zusammenarbeit mit dem FrontEnd zu ermöglichen, dienen die Verwendung eines Json-Files und die Arbeit über die Datenstruktur des **Person**-Objekts als Schnittstellen. Im FrontEnd werden die bereitgestellten Daten verarbeitet und über eine interaktive Plattform visuell dargestellt. Der daraus entstehende Graph soll in variabler Anzahl Hops angezeigt werden können. Zusätzlich sollen Metriken für den Social Graph Explorer bereitgestellt werden. Der Influencer Status und der Aktivitätsgrad einer Person werden hauptsächlich im BackEnd berechnet. Das Visualisieren von Cliquen und mögliche Follow-Vorschläge werden im Verlaufe der Projektarbeit als zusätzliches Ziel des FrontEnds definiert, um die Aufgabenlast gleichmässig zu verteilen. Nebst den inhaltlichen Zielen soll eine gute und verständliche Dokumentation der Implementation erreicht werden, um den Einstieg in eine allfällige Erweiterung zu erleichtern. Ein Ziel, welches nicht erreicht werden konnte, war die Umsetzung des Social Graph Explorer mit der grossen Datengrundlage von Secure Scuttlebutt[1].

Hintergrund

Die bisher bestehenden Teile des BACnets wurden im Rahmen der letztjährigen Projekte erarbeitet. Da diese nicht vollständig integriert sind, dient ein kleines Netzwerk aus manuell erstellten Personen und entsprechenden Follow Beziehungen als Grundlage für die Umsetzung des Social Graph Explorers. Grundlegende Strukturen, wie der Feed und das Erstellen von privaten Schlüsseln, können von den Projekten aus dem Vorjahr übernommen und zur Generierung der projektinternen Struktur genutzt werden. Auf einem Feed werden Informationen zu allen Aktivitäten im BACnet gespeichert. Die entsprechenden Einträge (Events) werden verkettet, wobei ein neuer Event am Ende der bestehenden Kette angefügt wird. Das Löschen von Events ist nicht möglich. Durch die Synchronisierung von Feeds können Informationen über das Geschehen im Netzwerk unter Personen ausgetauscht werden. Als Orientierung für unser Projekt dient der Follow Graph von Secure Scuttlebutt. Der Follow Graph zeigt von einem gegebenen Feed aus, welchen anderen Feeds dieser folgt. Der Follow Graph kann in mehreren Ebenen (Hops) dargestellt werden. Je nachdem, wie viele Verkettungen bestehen, kann die Darstellung mit steigender Anzahl angezeigter Hops unübersichtlicher werden.

[1] http://scuttlebot.io/

Implementation

Um eine grobe Übersicht über die Implementation zu geben, werden die Schnittstellen und die wichtigsten Klassen im Folgenden kurz erklärt.

Schnittstellen

Als Schnittstelle zur FrontEnd-Gruppe wurde die Klasse **Person** und das Json-File `loadedData.json` definiert.

Im Json-File werden alle Daten für den aktuellen Graphen gespeichert. Gespeichert wird eine Liste von Knoten, gefolgt von einer Liste von Verbindungen im Graphen. Jeder Knoten stellt eine:n Nutzer:in dar. Hier werden alle Attribute (diese werden in **Person** genauer erklärt) zum:r jeweiligen Nutzer:in, sein:ihr Name, seine:ihre BACnet-Id und seine:ihre ID innerhalb des Graphen gespeichert. Zusätzlich wird noch das HopLayer bestimmt. Die Anzahl Hops ist ein minimaler Pfad zwischen der Hauptperson und dem:r Nutzer:in. In der Liste der Verbindungen wird eine Follow-Verbindung von einer Person zu einer anderen gespeichert. Dafür speichern wir für jede Verbindung die Graph-ID der folgenden Person und die Graph-ID der gefolgten Person. Das Json-File wird im Ordner der FrontEnd-Gruppe abgespeichert.

Die Klasse **Person** dient zur Behandlung von neuen Veränderungen, wie zum Beispiel einem neuen Folgen oder einer Änderung im Bereich der Attribute. Diese Klasse wird weiter unten genauer erklärt.

Main

Die **Main**-Klasse bildet die Grundlage unserer ganzen Implementation. Der **Main**-Klasse wird ein Name übergeben. Dieser Name ist der Name der Hauptperson. Aus ihrer Sicht wird der Graph dargestellt. Wir nehmen an, dass man sich am Anfang im BACnet anmelden muss und daher der Name der Hauptperson schon bekannt ist.

Als erstes ruft sie die Klasse `generateDirectories` auf. Hier wird der Ordner `data` erstellt, in dem anschliessend für jede:n Benutzer:in ein eigener Ordner eingefügt wird, in dem sein Feed-File und sein Key-File gespeichert werden. In unserer Implementation sind das manuell erstellte Benutzer:innen. Nach der Integrierung im BACnet kann diese Zeile im `main` weggelassen werden, da die bekannten Benutzer:innenordner durch das BACnet automatisch abgespeichert werden.

Anschliessend wird in der `main`-Methode über alle bekannten Benutzer:innenordner in `data` iteriert. Für jede:n Benutzer:in wird ein **Feed**-Objekt und ein **Person**-Objekt erstellt. **Feed** und **Person** werden später erklärt.

Zum Schluss wird das Json-File für die FrontEnd-Gruppe aktualisiert.

Feed

In dieser Klasse werden die Einträge aus dem Feed ausgelesen und neue Einträge für den Feed mit der entsprechenden Formatierung in den Feed eingetragen.

Person

Diese Klasse ist ein Teil der Schnittstelle für das FrontEnd. Für jede:n Nutzer:in wird ein **Person**-Objekt erstellt. Hier werden seine:ihre Attribute, seine:ihre Follow-Liste, sowie seine:ihre BACnet-ID, sein:ihr Name, sein:ihr Profilbild und sein:ihr **Feed** abgespeichert. Folgende Attribute wurden von uns implementiert:

1. Geschlecht
2. Geburtstag
3. Land
4. Stadt
5. Sprache
6. Status
7. Aktivitätslevel
8. Influenzerstatus

Die Attribute 1-6 können über die GUI der FrontEnd-Gruppe von dem:der Nutzer:in angegeben und verändert werden. Sie werden daraufhin auf dem Feed abgespeichert. Dasselbe gilt für das Profilbild des:r Nutzers:in. Das Aktivitätslevel und der Influenzerstatus wird in **Person** für den:die Nutzer:in berechnet. Beim Aktivitätslevel werden erst alle Einträge im Feed gezählt. Je nach Anzahl der Einträge wird der:die Nutzer:In in eine Kategorie eingeteilt:

- weniger als 10 Aktivitäten: Kategorie 0
- zwischen 10 und 25 Aktivitäten: Kategorie 1
- zwischen 25 und 45 Aktivitäten: Kategorie 2
- zwischen 45 und 70 Aktivitäten: Kategorie 3
- zwischen 70 und 100 Aktivitäten: Kategorie 4
- mehr als 100 Aktivitäten: Kategorie 5.

Wer mehr als 3 Follower hat, erhält den Influenzerstatus. Durch ein Entfolgen eines:r Freundes:in kann man diesen Status auch wieder verlieren. Somit ist er dynamisch. Die Zahl 3 wurde von uns aus Testzwecken gewählt. Sie kann für grössere Graphen angepasst werden. Neben dem Verändern der Attribute ist es für den:die Nutzer:in möglich, einem:r neuen Freund:in zu folgen oder einem:r bestehenden Freund:in zu entfolgen. Dies kann über die Methoden `follow` und `unfollow` ausgeführt werden. Auch diese Events werden entsprechend in den Feed eingetragen und die Follow-Liste wird aktualisiert. Nach jeder Veränderung, sei es in der Follow-Liste oder ein Attribut, wird das Json-File für die FrontEnd-Gruppe aktualisiert.

Resultate

Unser Ziel konnte erfolgreich umgesetzt werden. Über das zur Verfügung gestellte Json-File kann die FrontEnd-Gruppe den Graph mit den richtigen Informationen darstellen. Das Json-File steht der FrontEnd-Gruppe immer auf dem aktuellsten Stand zur Verfügung. Veränderungen, die über die GUI des FrontEnds während des laufenden Programms getätigt werden, werden über die zweite Schnittstelle, die Klasse **Person**, direkt im Json-File angepasst.

Diskussion

Die aktuelle Implementation des Social Graph Explorers wird über eine festgelegte Hauptperson (via Konsolen-Input), den:die Nutzer:in, gesteuert. Das Ziel wäre es, dass diese Person nicht manuell angegeben werden muss, sondern bei der Integration in BACnet der:die Nutzer:in jeweils dynamisch erkannt wird.

Die zu Beginn des Projektes festgelegten Ziele bezüglich BACnet konnten gut umgesetzt werden. So ist es nun möglich, unterschiedliche Attribute für das eigene Profil festzulegen. Ausserdem lassen sich Attribute wie zum Beispiel der Aktivitäts-status automatisch berechnen. Der Kern des Projektes, die Übersicht über die Follow-Beziehungen innerhalb des Netzes und deren graphische Darstellung (siehe FrontEnd), konnte ebenfalls erfolgreich in BACnet integriert und mit eigens erstellten Daten getestet werden.

Ein Nachteil der aktuellen Implementierung ist die Effizienz. Diese ist eher gering, da für jede Veränderung, wie zum Beispiel die Veränderung eines Attributs oder das Folgen und Entfolgen, das gesamte Json-File neu geschrieben wird. Da die Implementierung bisher nur mit kleineren Datensätzen getestet wurde und das Json-File entsprechend klein gehalten wurde, wirkte sich dieses Problem nicht auf die Performance aus. Bei Json-Files mit einer grossen Anzahl an Einträgen könnte das wiederholte Aufsetzen des Files jedoch zu Verzögerungen bei der Aktualisierung führen.

Leider war es uns nicht möglich, den Social Graph Explorer auch mit grösseren Datenbanken, wie der von Secure Scuttlebutt, zu testen. Wir haben Accounts bei Secure Scuttlebutt angelegt und mit deren Hilfe einem Hub folgen können. Das Ziel wäre gewesen, über diesen Hub den Zugriff auf eine grosse Menge an Accounts und entsprechend auch deren Feeds zu erlangen. Diese Daten hätten dann mittels des Social Graph Explorers ausgelesen werden sollen und die sozialen Verknüpfungen zwischen den Feeds grafisch dargestellt werden sollen. Leider konnten wir dieses Vorhaben nicht umsetzen. Für den Zugriff auf ebendiese Daten wären weitere Installationen und Importe nötig gewesen. Diese gestalteten sich etwas schwierig und trotz mehrerer unterschiedlicher Lösungsversuche liess sich das Problem nicht lösen, zumal uns die entsprechende Hilfestellung nicht zur Verfügung gestellt wurde.

Fazit

Der Einstieg in das Projekt fiel uns eher schwer, da das Einlesen in die bereits vorhandenen Projekte einige Zeit beanspruchte. Zudem fehlte uns die Übersicht, wie die einzelnen Funktionalitäten von BACnet zusammenhängen. Trotz der Anlaufschwierigkeiten blieb die Kommunikation innerhalb des Teams immer sehr gut. Wir haben uns regelmässig zu Meetings getroffen, Ideen ausgetauscht und die Implementation gemeinsam ausgearbeitet.

Schwieriger gestaltete sich die Zusammenarbeit mit der FrontEnd Gruppe. Das lag vor allem daran, dass die Schnittstelle zunächst nicht klar definiert werden konnte und dementsprechend beide Gruppen im Vorhinein nicht voraussagen konnten, was von den jeweils anderen benötigt wird. Das führte unter anderem dazu, dass einige Funktionalitäten auf beiden Seiten implementiert wurden. Schlussendlich gelang es uns, die beiden Teile des Projekts miteinander zu verknüpfen und ein erfolgreiches Endprodukt zu gestalten.

Leider gestaltet sich die Kommunikation mit den Projektverantwortlichen etwas aufwändig. Anfangs stellten vor allem die Assistenten ihre Unterstützung zur Verfügung. Gegen Ende des Projekts fehlte uns trotz mehrfachen Nachfragens die Hilfe bei der Gewinnung und Auslesung der Secure Scuttlebutt Daten. Das resultierte darin, dass wir diesen Teil des Projekts nicht umsetzen konnten.

Wir sind stolz, dass wir trotz einiger Schwierigkeiten ein erfolgreiches Projekt abschliessen konnten und die Implementation im Rahmen von BACnet sowie zusätzliche Features, wie Aktivitätslevel oder das Influenzer-Kennzeichen, bereitstellen konnten.