

UNIVERSITÄT BASEL

INTERNET AND SECURITY

FS 2021

BACnet Social Graph - Frontend

Authors

Pascal KUNZ, Philipp HALLER, Sebastian SCHLACHTER

16. Juli 2021

Einführung

Im Rahmen des Projektes in der Vorlesung Internet and Security stellten wir uns die Aufgabe das Frontend des sozialen Graphen vom BACnet zu implementieren. Die Visualisierung des Netzwerks in Form eines gerichteten Graphen steht dabei im Zentrum, jedoch soll das Frontend den Nutzern auch weitere Funktionalitäten zur Verfügung stellen. Dazu zählen hauptsächlich das Folgen und Entfolgen von Personen, sowie das Updaten der eigenen Nutzerinformationen. Die Möglichkeit, das Look and Feel und die Werte der Graph-Attribute anpassen zu können, soll es den Nutzern ferner erlauben, das Aussehen des Frontends nach eigenem Geschmack zu personalisieren.

Hintergrund

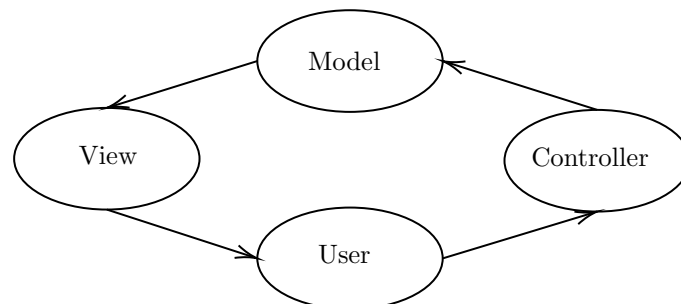
Django

Django ist ein leistungsfähiges Python-Webframework. Wir haben uns für dessen Nutzung entschieden, da es unter anderem den Fokus der Applikationsentwicklung auf folgende Eigenschaften setzt:

- **Vielseitigkeit:** Django bietet viele Hilfsmittel der Webentwicklung wie bspw. die einfache Verknüpfung von Datenbanken und die Bereitstellung von Content in verschiedensten Formaten (JSON, HTML, XML usw.). Einige der populärsten Websites wie YouTube, Instagram oder Spotify benutzen das Django-Webframework für ihre webbased Applikationen.
- **Plattformübergreif:** Die Serverplattform für eine Django-Website kann von jedem Betriebssystem bereitgestellt werden, auf welchem ein Python-Interpreter vorhanden ist.
- **Wartbarkeit:** Django verwendet das Softwaredesignpattern Model-View-Controller. Ein klares Design-Pattern gibt dem Programmierer einen Leitfaden für die Programmierung und führt zu einer hohen Verständlichkeit, Wartbarkeit wie auch Wiederverwendbarkeit des Codes.
- **Skalierbarkeit:** Das Design von Django erlaubt es dem User, eine Applikation in einem kurzen Zeitfenster zu erstellen. Es gibt dem Entwickler die Möglichkeit, viele Tools der Webentwicklung in die Applikation zu integrieren. So kann beispielsweise die standardmässige SQLite Datenbank durch Datenbanken, die auf anderen Server laufen, ersetzt werden. Des Weiteren bietet die Django Community wertvolle Tools für das Auffinden von Bottlenecks, Debugging und das Verbessern der Performance der Applikation.

Das Model-View-Controller-Schema

Das Designpattern, das Django verwendet, kann visuell folgendermassen dargestellt werden:



- **Views / Presentation Layer:** Die Presentations Layer stellt die Daten des Models graphisch dar. Dies bietet eine visuelle Referenz an den User.
- **User:** Der User nimmt graphische Inputs von der Presentations Layer wahr und nimmt per Eingabe Einfluss auf den Controller.
- **Controller:** Der Controller gibt Usereingaben weiter an das Model.
- **Model:** Das Model hat die Aufgabe, die Daten zu verwalten und auf Anfrage des Controllers zu aktualisieren. Es stellt die Daten für die Präsentationsschicht bereit.

D3

Neben Django verwenden wir als zweites grosses Tool D3.js, eine JavaScript-Bibliothek zur Erzeugung dynamischer und interaktiver Datenvisualisierungen. D3 erlaubt es datengesteuerte Transformationen auf ein Dokument anzuwenden und hat dabei das Ziel, den Kern des Problems, die effiziente Manipulation von Dokumenten auf Basis von Daten, zu lösen. D3 ist kein monolithisches Framework, das versucht, alle erdenklichen Funktionalitäten zur Verfügung zu stellen. Durch das Ausschöpfen der vollen Möglichkeiten von Webstandards wie HTML, SVG und CSS kann eine grosse Flexibilität garantiert und proprietäre Darstellungen vermieden werden. Zudem ist D3 mit minimalem Overhead extrem schnell und unterstützt große Datensätze und dynamische Verhaltensweisen für Interaktionen und Animationen.

AJAX-Calls

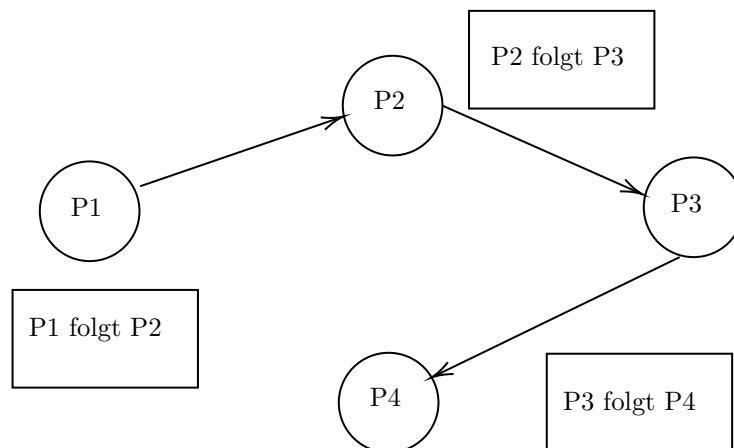
AJAX ist eine Abkürzung für Asynchronous JavaScript and XML. Es wird verwendet, um mit einem Server asynchron zu kommunizieren, was bedeutet, dass die Kommunikation ohne Reload der Seite von statten geht. Dabei können entweder JSON, XML, HTML oder text files übertragen werden. In unserer Implementation verwenden wir AJAX-Calls beispielsweise, um die Präsentationslayer nach Usereingabe zu rerendern, ohne dass die Seite neu geladen werden muss.

Levenshtein / Editier-Distanz

Gegeben zwei Strings berechnet die Levenshtein-Distanz die Anzahl von Lösch-, Einfüge- oder Ersetzoperationen um ein String in einen anderen zu verwandeln. Diese Metrik wird vor allem dann verwendet, um die webbasierte Suche in Verbindung mit Datenbanken zu verbessern. Angenommen in einer Datenbank befindet sich ein Objekt mit dem Namen "Hans". Beim Suchen nach diesem Namen vertippt sich User A und gibt anstelle von "Hans" "Fans" ein. Die Levenshtein-Distanz beträgt also 1. Für das Suchergebnis werden jetzt nur noch die Objekte der Datenbank mit der niedrigsten Levenshtein-Distanz zum eingegeben fehlerhaften Suchstring angezeigt. Dies wurde auch bei uns im Follow-GUI bei der Suche nach Städten und Namen so umgesetzt.

Sozialer Graph / Hoplayer

Das Konzept des sozialen Graphen und der Hoplayer lässt sich graphisch am einfachsten erklären. Der komplette soziale Graph eines Netzwerks ist folgendermassen aufgebaut, wobei P1 der Rootuser ist:



Eine gerichtete Verbindung zwischen zwei Knotenpunkten bedeutet, dass eine Person der anderen folgt. Anhand dieses Netzwerks lassen sich die sozialen Untergraphen im Bezug auf die Hoplayer, also die maximale Anzahl Kanten zwischen Rootuser und einem weiteren Teilnehmer des Netzwerks, nun erklären:

- Szenario Hoplayer 1: $P1 \rightarrow P2$
- Szenario Hoplayer 2: $P1 \rightarrow P2 \rightarrow P3$
- Szenario Hoplayer 3: $P1 \rightarrow P2 \rightarrow P3 \rightarrow P4$

In unserer Implementation überlassen wir dem User bis zu Hoplayer 5 die Wahl, für welche Schicht ein sozialer Untergraph erstellt werden soll.

Setup / Install

Um die nötigen Dependencies zu installieren, haben wir ein Skript programmiert. Es befindet sich im Ordner 'FrontEnd'.

Folgender Befehl installiert alle Dependencies:

```
python setup.py install
```

Um das FrontEnd zum Laufen zu bringen, sind nun folgende weitere Schritte notwendig:

1. Man lässt das main-Programm vom Backend laufen. So werden die PCAP-Files und das JSON-File erstellt.
2. Man manövriert zum Ordner 'FrontEnd', in welchem sich das File manage.py befindet.
3. Man führt folgenden Befehl aus: **python manage.py makemigrations** So wird sichergestellt, dass Veränderungen, die am Model vorgenommen wurden, für das Migrieren in die Datenbank bereit gemacht werden.
4. Anschliessend: **python manage.py migrate** So werden Veränderungen an der Datenbank angewandt.
5. Der Server kann nun gestartet werden mit: **python manage.py runserver**. Der Server läuft anschliessend auf dem Localhost.

Implementierung

Kommunikation mit dem Backend:

Die Kommunikation vom Front- zum Backend wird direkt über Funktionsaufrufe geführt, wobei die Informationen als Parameter übergeben werden. So können beispielsweise Follow-Events oder Profile-Updates übergeben und in den Feed des Nutzers geschrieben werden. Alle nötigen Funktionen im Backend werden dabei im Thread des Frontendes ausgeführt. Dadurch können zwar Ladezeiten für das Userinterface entstehen, die Zusammenarbeit und eine schnelle Entwicklung wird mit diesem Ansatz aber vereinfacht.

In die andere Richtung, also vom Back- zum Frontend, werden Informationen hingegen per JSON-Datei übergeben. Diese besteht aus zwei Teilen: Einer Liste von Nodes und einer Liste von Links. Die Nodes entsprechen dabei einer Hashmap, welche alle Attributwerte definiert, die zu einer Node bzw. deren Nutzerprofil gehören. Auch die Links entsprechen Hashmaps, welche jeweils ein Attribut für deren Start und Ziel haben. So beschreiben sie die Verbindungen innerhalb des Netzwerkgraphs. Die JSON-Datei wird neu geschrieben, sobald sich im Backend etwas ändert, also immer dann, wenn etwas in den Feed geschrieben wird. Das Frontend liest dann die JSON-Datei erneut aus und updatet den Graphen, seine Datenbank und damit auch die Profile entsprechend.

Home

Das Home-Userinterface ist die erste Seite, die beim Zugriff auf das Frontend sichtbar ist, es ist in Abbildung 1 zu sehen. Es soll den User einerseits willkommen heissen und andererseits interessante Metriken wie die Anzahl Follows und Followers, den Aktivitäts- und Influencerstatus graphisch ansprechend darstellen. Des Weiteren lassen sich folgende Eigenschaften des sozialen Graphen von hier anpassen: Der Radius der Knotenpunkte, die Länge der Links, die Schriftgrösse, die Farben für die unterschiedlichen Geschlechter (Abbildung 2 zeigt dieses Einstellungs-Menü).

Graph:

Zur Implementierung des Graphen verwenden wir das Force-Layout von D3, mit welchem D3 einen physik-basierten Simulator zur Positionierung von Elementen zur Verfügung stellt. Dieser erlaubt es, die einzelnen Elemente, sogenannte Nodes, über Linien, sogenannte Links, basierend auf den zugrunde liegenden Daten zu verbinden. Durch das Hinzufügen von Pfeilen am Ende der einzelnen Links ergibt sich ein interaktiver und dynamischer gerichteter Graph. Da das Force-Layout unter anderem das JSON-Format als Dateninput akzeptiert, können die vom Backend empfangenen Informationen ohne grosse Umwege visuell ansprechend dargestellt werden. Über die Home-Page hat man zudem -wie bereits angesprochen- Zugang zu den Graph-Settings. Die Werte dieser Settings werden in einem separaten JSON-File gespeichert und bleiben somit auch bei einem Browser- oder Serverrestart bestehen. Betreffend der Grösse der Nodes gilt es ferner zu erwähnen, dass diese nicht nur vom gesetzten Radius abhängt, sondern auch die Anzahl der Followers in Betracht gezogen werden. Der Graph wird in Abbildung 3 gezeigt.

Profilseiten:

Jeder Nutzer verfügt über eine eigene Profile-Page, auf welcher er verschiedene Informationen über sich anzeigen lassen kann. In der Datenbank gibt es für jedes Profil ein Objekt, welches alle Attribute enthält, die den Nutzer beschreiben (Bsp.: Geschlecht, Geburtsdatum und Wohnort). Jedes Profilobjekt hat dabei einen Primary-Key, eine einzigartige Nummer, welcher der Node-ID im Graphen entspricht. Diese Nummer wird auch genutzt, um die verschiedenen Profile-Page-URLs zu beschreiben. Django erstellt für jedes Profil dynamisch eine HTML-Datei anhand des von uns geschriebenen Templates, welches mit Informationen aus dem entsprechenden Datenbankobjekt ergänzt wird. Das Userinterface der Profileseiten ist in Abbildung 4 zu sehen.

Profil Updates:

Der Nutzer kann alle Informationen, die in seinem Profil angezeigt werden, verändern oder löschen (verstecken). Dazu drückt er in seinem Profil auf den Update-Button. Auf der Update-Seite kann er nun in den einzelnen Feldern die jeweiligen Attribute bearbeiten und die Änderungen mit „Save“ speichern. Im Hintergrund füllt er dabei ein HTML-Formular aus, dessen Daten werden dann durch das Drücken von „Save“, welches als Submit-Button dient, per POST an Django übergeben und können anschliessend in der entsprechenden Methode weiterverarbeitet werden. Wurden tatsächliche Änderungen vorgenommen, werden diese -wie zuvor beschrieben- an das Backend weitergeleitet. Das Userinterface der Profil-Update-Seite ist in Abbildung 5 zu sehen.

Follow / Unfollow GUI

Der graphische Aufbau des Follow- und Unfollow-GUI ist in Abbildung 6 ersichtlich. Hier werden nun dessen Funktionalitäten erklärt: Die Suche lässt sich anhand des Filter-Menüs auf der linken Seite nach Geschlecht, Name, Stadt, Influencer, Hoplayer und Alter einschränken, woraufhin die Follow-Recommendations auf der rechten Seite entsprechend angepasst werden. Ebenfalls kann der Modus von Follow zu Unfollow per Toggle-Switch gewechselt werden. Per Knopfdruck auf 'Follow!' oder 'Unfollow!' (je nach Modus) kann einer Person gefolgt oder entfolgt werden. Mit jeder Interaktion des Users innerhalb des Filtertableaus werden per AJAX-Call die Filterparameter via dem Controller an das Model weitergeleitet, woraufhin mit SQL-Queries ein gefiltertes, neues Queryset erstellt wird, was bewirkt, dass die Präsentationsschicht mit den aktualisierten Daten neu gerendert wird.

Look and Feel

Die Möglichkeit, zwischen einem 'Light'- und einem 'Dark'-Mode hin und her zu wechseln, gehört bei heutigen Applikationen fast schon zum Standard und sollte daher auch beim BACnet Social-Graph-Frontend nicht fehlen. In der rechten oberen Ecke des Interfaces findet man den entsprechenden Switch, der beim Betätigen die vom CSS Stylesheet verwendete Farbpalette entsprechend ändert. Die Auswahl wird zudem über Local-Storage permanent lokal auf dem Computer des Nutzers gespeichert, wodurch sie auch nach einem Neustart des Browser oder des Servers beibehalten wird.

Resultate

Während des Projektes konnten wir wie geplant ein webbasiertes Userinterface entwickeln, welches dem Nutzer die Möglichkeit bietet, Informationen aus Feeds anzuzeigen und in seinen eigenen Feed zu schreiben. Optisch haben wir dabei versucht, eine möglichst stilvolle Einfachheit zu erreichen. Dies ist uns durch die Nutzung einheitlicher Formen, Schriften und Farbpaletten unserer Meinung nach gut gelungen. Doch auch funktional kann das Frontend überzeugen, so lässt sich das Netzwerk aus umliegenden (sich folgenden) Personen anschaulich darstellen und interpretieren. Mit Nutzer definierten Farben lassen sich zudem direkt im Graphen weitere Informationen wie das Geschlecht anzeigen und per Schieberegler kann die Sichtweite zwischen 1 und 5 Hops gewählt werden. Wir haben es uns auch zum Ziel gemacht, den Follow- und Unfollow-Prozess für den Nutzer zu vereinfachen. Die Follow-Page ermöglicht dies, Nutzer können per Mausklick anderen Nutzern folgen oder aufhören, dies zu tun. Die Suche nach anderen Personen konnten wir zudem mit einer umfassenden Filtermaske vereinfachen, die sich sehr spezifisch konfigurieren lässt. Ist man schliesslich mit den gewünschten Personen vernetzt, kann man sich Informationen von diesen auf ihren Profilseiten anschauen und sein eigenes Profil direkt im Webinterface verwalten, sogar das Hochladen eines Profilbildes ist möglich. Im Grossen und Ganzen konnten wir die meisten unserer Ideen wie geplant umsetzen und sind mit unserem Userinterface sehr zufrieden. Wir würden den aktuellen Implementationsstand jedoch noch nicht als fertiges Produkt bezeichnen, denn es gibt durchaus noch ein paar Knackpunkte und Probleme zu lösen, doch mehr dazu im nächsten Abschnitt.

Lessons Learned

Während des Projektes konnten wir viele neue Dinge in Erfahrung bringen und kennenlernen. So konnten wir praktische Erfahrungen im Umgang mit Django sammeln, aber auch unsere HTML- und CSS-Kenntnisse auffrischen und ergänzen. Zusätzlich haben Einzelne von uns sich jeweils stark mit den Visualisierungsmöglichkeiten, die D3 bietet, auseinandergesetzt können oder einen ersten Einblick in das Arbeiten mit Datenbanken erhalten.

Das Arbeiten als Gruppe hat sehr gut funktioniert, in wöchentlichen Meetings konnten geplante Ideen und Aufgaben entwickelt, besprochen und verteilt werden, dabei konnte jeweils auf die Interessen und Kenntnisse der Gruppenmitglieder eingegangen werden und bei Problemen konnte man sich schnell gegenseitig helfen und unterstützen. Während der engen Zusammenarbeit mit der Backend-Gruppe haben wir gelernt, dass es oft schwierig sein kann, eine strikte Teilung in Front- und Backend vorzunehmen. Für uns hat sich deshalb ein eher fließender Übergang der beiden Aufgaben- und Verantwortungsbereiche ergeben, mit denen wir in den zwei Dreiergruppen arbeiten konnten. Dabei haben wir auch gelernt, dass eine klare, deutliche und häufige Kommunikation absolut essentiell ist, um ein funktionierendes Gesamtsystem zu erreichen, denn oft hatten Front- und Backend sehr unterschiedliche Perspektiven auf ein Problem und dessen Lösung.

Damit in beiden Gruppen kontinuierlich gearbeitet werden konnte, mussten zudem gewisse Entscheidungen bereits sehr früh im Verlaufe des Projektes getroffen werden.

Aufgrund dessen und auch der noch eher geringen Erfahrung mit webbasierten Userinterfaces gibt es deshalb noch ein paar Punkte, die bemängelt und für einen Einsatz mit wirklichen Usern noch verbessert werden sollten. Dies betrifft unserer Meinung nach einerseits den Informationsaustausch mit dem Backend, welcher effizienter gestaltet werden muss, denn aktuell werden zu oft redundante Informationen an das Frontend gegeben. Die Performance könnte wahrscheinlich auch verbessert werden, wenn für Front- und Backend nicht der gleiche Thread genutzt wird. Andere Optimierungen könnten aber auch direkt auf Seiten des Frontends gemacht werden, beispielsweise beim Speichern von Nutzerdaten. So speichert aktuell das Frontend alle Userinformationen dreifach, einmal für die Darstellung des Graphen, für die Darstellung der Profile und für das Erzeugen der Follow-Recommendations. Wir sind uns dieser Problematiken allerdings bewusst und gehen davon aus, dass mit mehr Zeit, einem grösseren Neudesign und vor allem all den Erfahrungen, die wir während dieses Projektes sammeln konnten, Performance- und Effizienzprobleme durchaus gelöst werden könnten. Dann sollte auch einer Nutzung in einem grösseren Netzwerk nichts mehr im Wege stehen und die Funktionalitäten könnten weiter ausgeweitet werden. In diesem Sinne sind wir froh, dass wir einen Grundstein für ein webbasiertes Userinterface zum BACnet legen konnten und sind gespannt auf die Arbeiten und Ideen zukünftiger Projekte.

Appendix

Abbildungen:

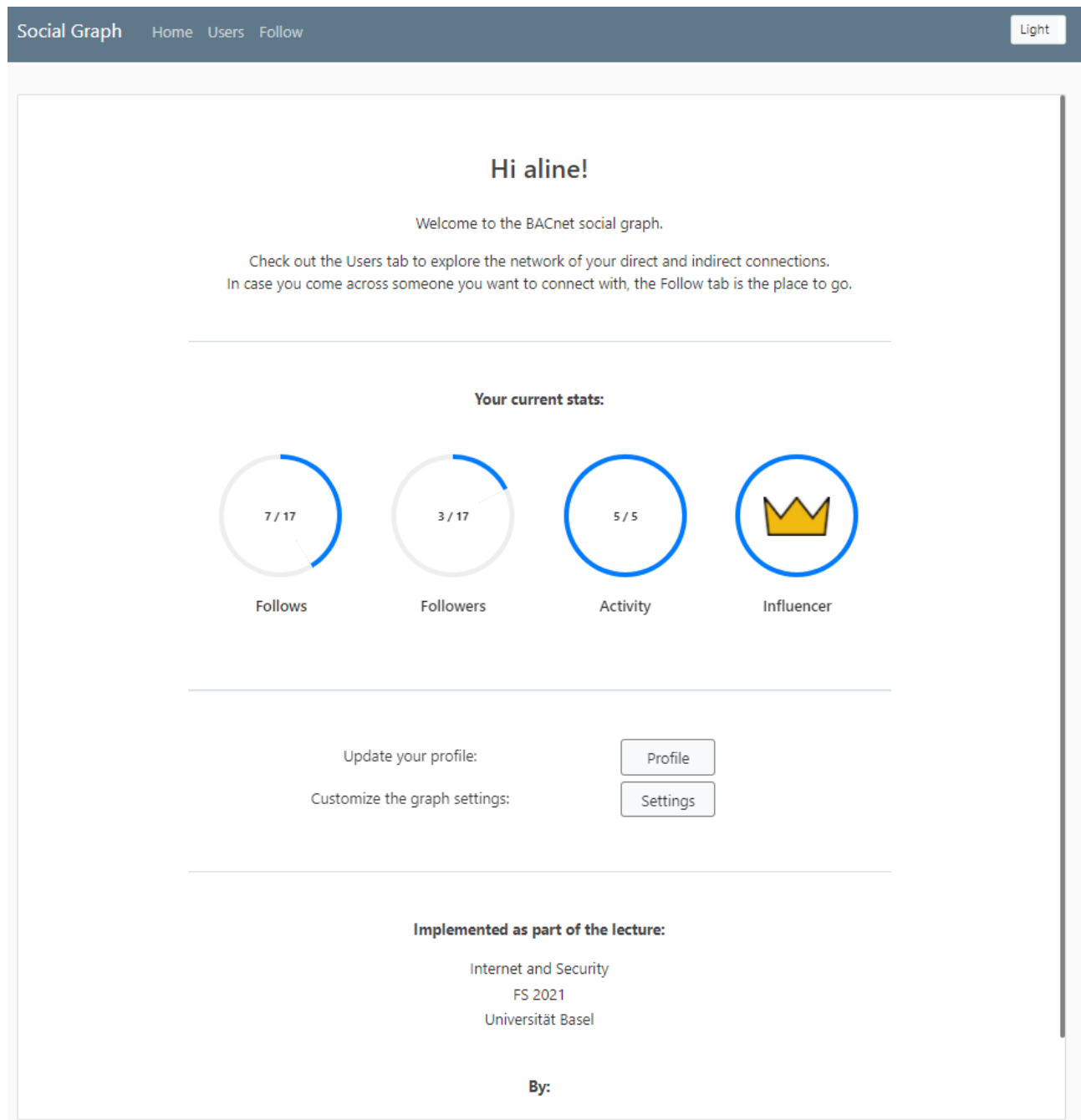


Abbildung 1: Home GUI

Graph Settings

Node radius	<input type="text" value="15"/>
Link length	<input type="text" value="200"/>
Text font-size	<input type="text" value="12"/>
Node (male)	<input type="color" value="#007bff"/>
Node (female)	<input type="color" value="#e83e7c"/>
Node (other)	<input type="color" value="#6c757d"/>

ResetSave changesClose

Abbildung 2: Graph Settings

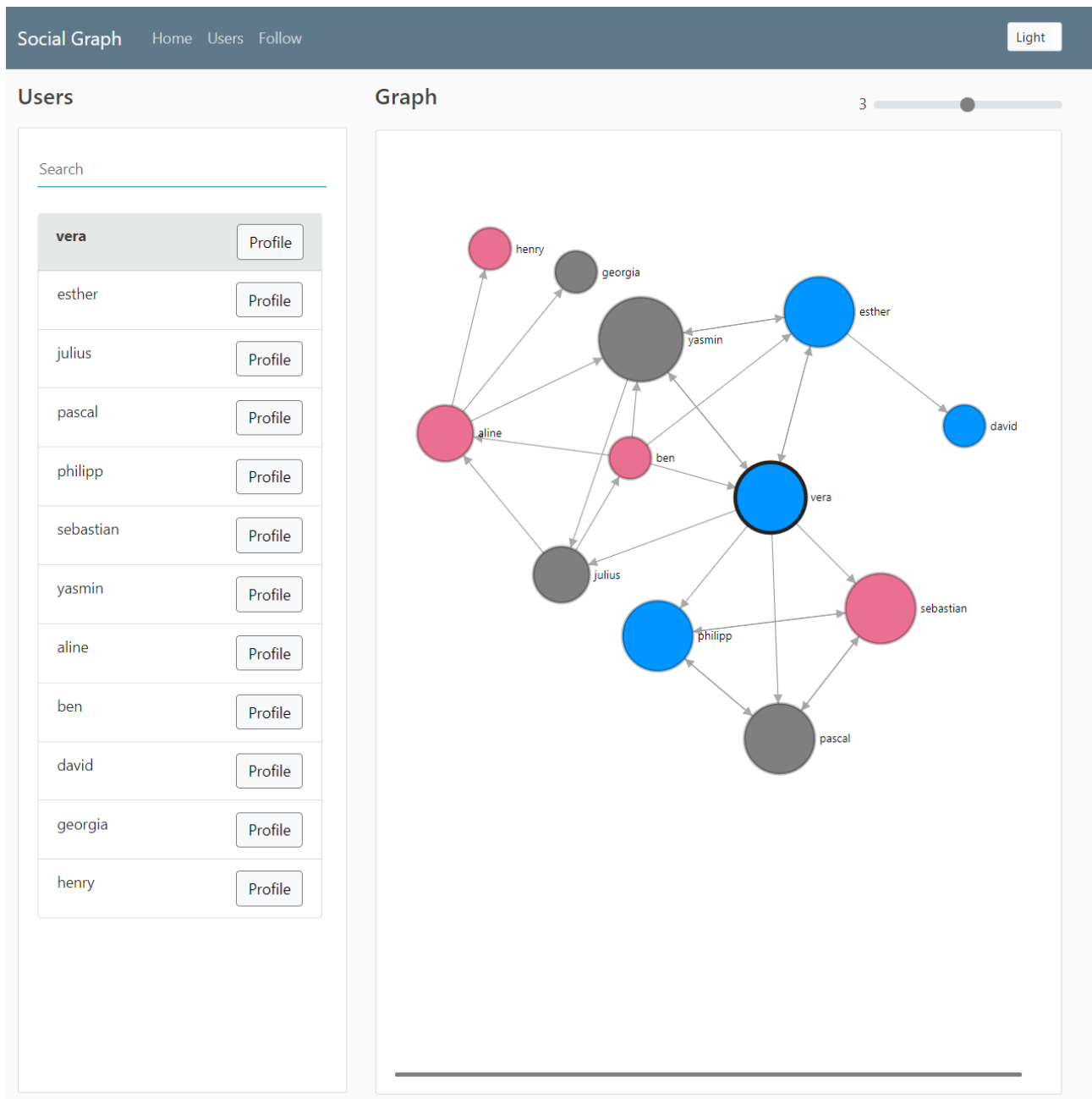




Abbildung 3: Graph

Social Graph
Home
Users
Follow
Light



vera
ID:a03997520a256f67
Status: I love Internet and Security!

Update




Details:


Gender:	female
Birthday:	June 1, 1999
Country:	Schweiz
Town:	Bern
Language:	Deutsch

Follows:


esther




julius



pascal



philipp



Status:

July 7, 2021, 10:30 a.m.
I love Internet and Security!


July 7, 2021, 10:30 a.m.
<https://www.youtube.com/watch?v=a3Z7zEc7AXQ>

Abbildung 4: Profilseite

Social Graph
Home
Users
Follow
Light

Profile Editor

Status
I love Internet and Security!

Birthday
01.06.1999


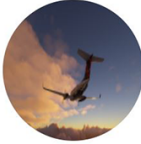
Gender
Female
Male
Other
del

Country
Schweiz

Town
type...

Languages
Deutsch

Save
Cancel

Profile Picture


Datei auswählen
Desktop S....58.44.png

Abbildung 5: Profile Update

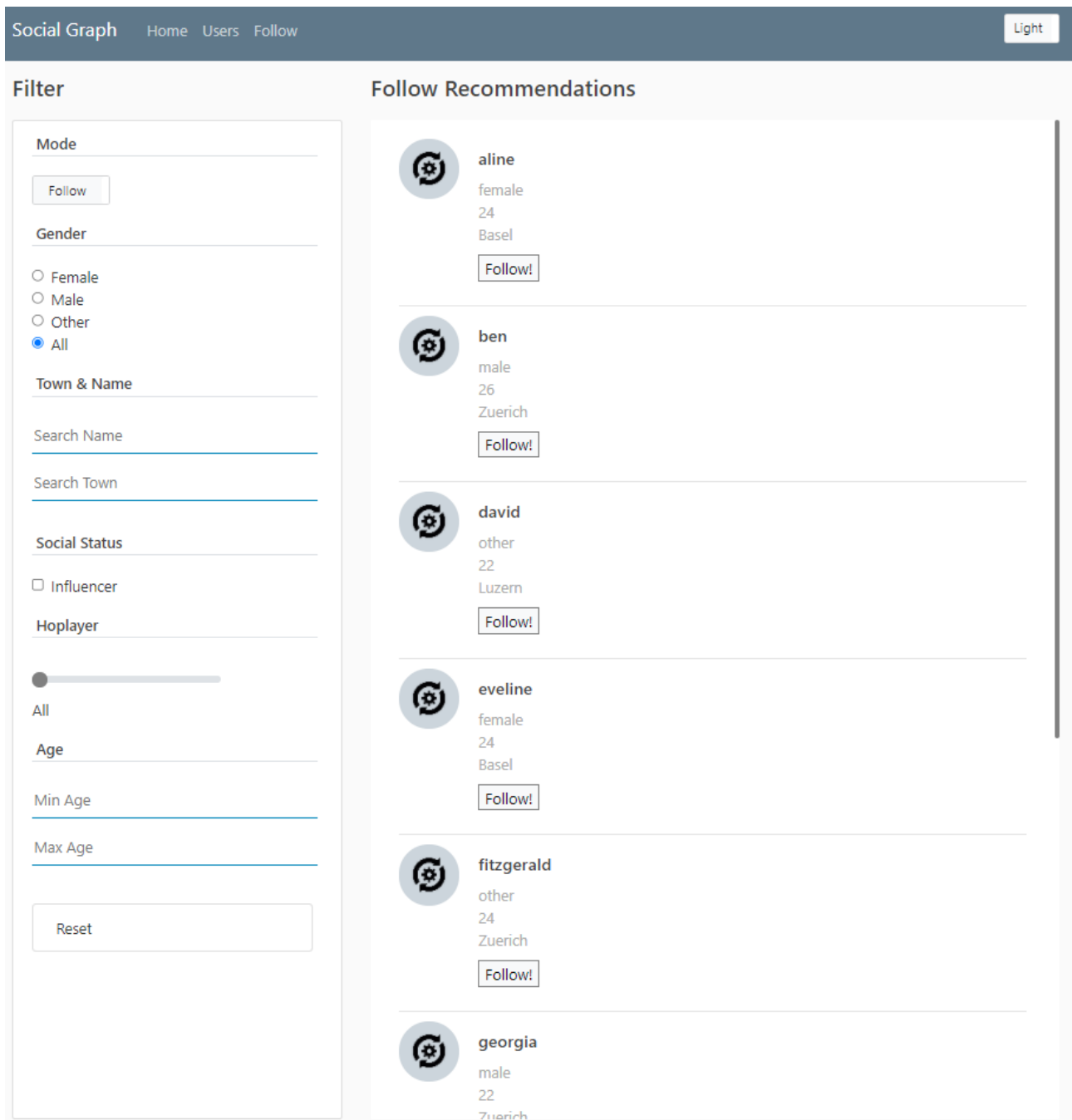


Abbildung 6: Follow / Unfollow GUI

Referenzen

- **Model-View-Controller:** https://de.wikipedia.org/wiki/Model_View_Controller
- **Django Introduction:** <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>
- **Famous Django Based Applications:** <https://www.geeksforgeeks.org/top-10-django-apps-and-why-companies-are-using-it/>
- **Django Tutorial:** <https://www.youtube.com/playlist?list=PL-osiE80TeTtoQC-KZ03TU5fNfx2UY6U4p>
- **D3:** <https://d3js.org/>
- **D3:** <https://www.d3-graph-gallery.com/network.html>
- **Local Storage:** <https://www.tutorialrepublic.com/html-tutorial/html5-web-storage.php>